N° d'ordre 2023-HDR-008 Année 2023

HABILITATION À DIRIGER DES RECHERCHES

présentée devant

l'Institut National des Sciences Appliquées de Lyon
et l'Université Claude Bernard Lyon I

Privacy Preservation in Trust-Deficient Decentralized Systems

SPÉCIALITÉ : Informatique

par
Omar HASAN

Soutenue le 16 juin 2023 devant la commission d'examen

Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS) – UMR 5205

Abstract

Decentralized distributed systems can provide certain advantages over their centralized counterparts. These include improved fault tolerance and attack resistance due to the elimination of a single point of failure, better censorship resistance and openness because of the absence of a central authority, as well as higher autonomy of the nodes due to self-governance of resources. Distributed ledger and blockchain technologies have played a key role in the increasing adoption of decentralized systems. One of the notable contributions of blockchains to decentralized systems is that they are able to ensure properties such as integrity, immutability, and verifiability even in trust-deficient environments where nodes lack trust in each other. However, ensuring the privacy of users still remains a challenge in decentralized systems where the application requires users to contribute confidential or identifiable information. Privacy is particularly challenging to achieve in trust-deficient environments due to the nodes not being able to rely on fellow nodes for privacy guarantees. My work addresses **privacy preservation in trust-deficient decentralized systems**. In this habilitation thesis, we include some selected contributions from our work in three broad areas: privacy-preserving decentralized reputation systems, privacy-preserving message routing in decentralized networks, and privacy preservation in decentralized financial networks.

The reputation of a user in a distributed system may be computed as an aggregate of the feedback provided by fellow users. **Privacy-preserving reputation systems** enable users to provide feedback in a private and thus uninhibited manner. We present several contributions in this area, which include a decentralized privacy-preserving reputation protocol based on the computation of mean that is secure under the malicious adversarial model. The protocol offers significant improvement in performance as compared to earlier protocols in the literature. Additionally, we define an attack called Reputation-based Re-identification (RR attack), which can link successive contributions provided by a participant in participatory sensing applications and subsequently re-identify them. We then propose PrivaSense, a privacy-preserving reputation system that defends against this attack. Experiments on a real dataset demonstrate that PrivaSense is successful in decreasing the number of contributions linked to their providers. Furthermore, we propose a voting protocol (as a generalized instance of a privacy-preserving reputation protocol) that ensures transparency, confidentiality, and integrity in a decentralized trust-deficient setup. The persistence and immutability of the protocol's communication allow verifiability of the outcome by the voters themselves. Our contribution on secure voting is further extended by our work on collusion-resistant worker set selection. We propose protocols that select a subset of workers (who process data during the protocol execution) from the set of participants such that the risk of collusion between workers is minimized.

Mobile Delay Tolerant Networks (MDTNs) are composed of mobile devices that communicate in a decentralized manner without the help of fixed infrastructure. A prediction-based routing protocol for MDTNs functions by forwarding a message from one intermediate node to another if the latter has higher probability of encountering the destination node. However, this process compromises the privacy of the nodes by revealing their mobility patterns. We propose the **Privacy-Preserving Probabilistic Prediction-based Routing** (4PR) protocol that forwards messages by comparing information about communities of nodes instead of individual nodes in order to protect their privacy. The protocol computes a probability function in a decentralized privacy-preserving manner. Simulations on a community-based mobility model demonstrate that our protocol is able to preserve privacy while offering performance comparable to protocols that do not protect privacy.

Many Decentralized Finance (DeFi) **Peer-to-Peer (P2P) lending platforms** offer users to obtain a loan by committing a collateral or by calculating a credit score. However, the requirements of collateral and credit history are quite burdensome for certain groups of users. We propose to use a social trustworthiness score drawn from users' social interactions as an alternative risk mitigator for lending instead of collateral. Privacy considerations are taken into account in order to protect the borrower's privacy despite the use of social interaction data. Another application that we address is **Supplier Impersonation Fraud (SIF) detection** in the Business-to-Business (B2B) context. This type of fraud occurs when a company supplying goods and services to another company is impersonated by a fraudster in order to trigger a payment to an illegitimate bank account. We introduce GraphSIF, a SIF detection system that aims to infer knowledge from the relational properties created by the transactions of a company. GraphSIF analyzes data that has been heavily anonymized in order to preserve the privacy of the participating companies. The classification of a targeted transaction is performed by first clustering the graphs, and then comparing the similarity between the targeted transaction's graph with the other graphs of its cluster. The model shows good efficiency in terms of computational time needed to create the behavior sequence and to classify the transactions.

Acknowledgments

First and foremost, I wish to thank the Rapporteurs (Reviewers), Nicolas Anciaux, François Charoy, and François Taïani, who generously accepted to review this work and provided valuable expert feedback.

I would like to express my profound thanks to the Examinateurs (Examiners), Elisa Bertino, Lionel Brunie, Stelvio Cimato, Christophe Garcia, Parisa Ghodous, and Andrew Martin, as well as the Rapporteurs, for graciously participating in the jury of the defense and for the enriching discussions and advice.

I thank all PhD and Master's students whom I had the opportunity of supervising and co-supervising. A large part of the work included in this habilitation thesis was made possible due to their hard work and contributions.

It has been a privilege working alongside Lionel Brunie on the scientific problems and the co-supervision of PhD students. I thank Lionel for his excellent advice and for creating a conducive research environment at the DRIM team and the IRIXYS center (as their founder).

My deepest gratitude goes to all research collaborators, co-supervisors, and co-authors. Their input and contributions were vital to the progress made in this work. In particular, I thank Elisa Bertino for her valuable advice and support throughout the years.

I would like to appreciate all my colleagues from the DRIM team, LIRIS lab, IRIXYS center, department of computer science (IF), INSA Lyon, and University of Lyon.

Lastly, but very importantly, I would like to thank my family, in particular my parents.

Contents

List of Figures

List of Tables

Part I

Introduction and Background

Chapter 1

Introduction

Decentralized systems have gained tremendous interest in recent years. In contrast to their centralized counterparts, decentralized systems exclude any central controlling entity or authority. This architectural choice allows decentralized systems to inherently favor fault tolerance and attack resistance due to the elimination of a single point of failure, offer better censorship resistance and openness because of the absence of a central authority, as well as support higher autonomy of the nodes due to self-governance of resources.

Decentralized systems, particularly in the form of Peer-to-Peer (P2P) systems, have been popular since over two decades. File sharing (e.g., using the BitTorrent protocol) was an early significant application of P2P systems. However, with the advent of distributed ledger and blockchain technologies in the previous decade, decentralized systems now find application in a variety of areas. These applications range from providing basic infrastructure, such as decentralized storage (e.g., Sia, Storj), to offering sophisticated decentralized financial services, such as decentralized lending (e.g., Aave, Compound, MakerDao), as well as cryptocurrencies (e.g., Bitcoin, Ether). At the time of this writing, the global cryptocurrency and decentralized application market capitalization stands at 1.15 Trillion Euros [21].

The reasons for the interest and popularity of decentralized systems are multifold. The monetary incentives on offer for the miners, validators, and investors are obvious draws. However, there are also some reasons behind the increasing adoption of decentralized systems that are more profound in nature. The case of the Web3 movement is an example. The web at its inception was largely decentralized with servers (owned by companies as well as individuals) offering original content. However, the web eventually shifted to a mostly centralized architecture where content is either owned, published, or algorithmically selected and ranked by a few major central entities. Web3 is a vision for the future of the web where individuals exercise higher control over the ownership, publication and accessibility of content.

Distributed ledger and blockchain technologies made new and innovative decentralized systems possible mainly because these technologies enable building consensus in large open networks in the presence of high ratios of self-serving or malicious nodes. Blockchain-based decentralized systems take advantage of consensus mechanisms to provide many desirable properties, such as immutability, transparency, verifiability, etc. However, blockchain-based decentralized systems are not without their problems. Some challenges that they currently face are limited scalability and efficiency, high energy consumption, difficult to achieve interoperability, and lack of privacy preservation.

Trust is an element that is necessary for the correct functionality and security of many distributed systems. Informally, trust implies a belief by the trusting node that the node being trusted will behave in an expected manner. As an example, a user needs to trust a cloud provider to maintain the promised quality of service. Moreover, the user trusts the cloud provider to uphold the security of their processes and data. Another example in a centralized system is a user on a creative content sharing platform trusting the central authority to list and promote their content in an unbiased manner. In a decentralized system, such as a vehicular network, an example would be a vehicle needing to trust the legitimacy of road conditions shared by a peer vehicle.

Unfortunately, trust in distributed systems can be misplaced and trust can be breached, which can harm the interests of the trusting nodes. Nodes can turn out to be partly honest, self-serving, or outright malicious. A cloud provider could promise high quality of service and security yet provide poor quality of service and insufficient security. A centralized content platform could bias search results in favor of selected creators. A vehicle could withhold information or even falsify it. A previously unknown node poses particularly high risk for being untrustworthy. However, even a well-known and trustworthy node can switch behavior and become malicious. Due to this uncertainty, trust can be deficient or lacking in distributed systems.

Trust needs to be established by nodes in order to securely participate in such distributed systems. In the case of centralized systems of global scale, users may trust the central entity by choice because of its recognizability and its cultivated image. On the other hand, users may trust the central entity simply out of necessity because equivalent alternatives are not available. In decentralized systems, globally well-recognized entities are generally absent. In these systems, one of the main approaches of establishing trust is through reputation.

Trustless systems propose an alternative to the need of trusting nodes. These systems replace subjective trust by a process of proofs and their verification. Bitcoin and other blockchain-based systems are examples of trustless systems. They assume trust to be completely nonexistent in the network, that is, no node trusts any other node. Bitcoin's Proof of Work (PoW) mechanism allows one node to prove the discovery of the solution to a cryptographic hash puzzle and all other nodes to efficiently verify the claim. This mechanism allows the nodes of the network to agree on the next node to append to the blockchain without knowing or trusting that node or even each other.

Systems that rely on trusted central entities and trustless systems that assume total absence of trust are on the two opposite ends of the trustfulness spectrum. However, we observe that trust deficiency in a network can range between these two limits. As we discuss later in Section 2.2.1, a node in a decentralized distributed system could be willing to trust a Trusted Third Party (TTP), some arbitrary fellow nodes, some chosen partially trusted nodes, or no nodes at all.

Even in blockchain-based systems, there are varying degrees of trustfulness that is assumed. In Bitcoin, as discussed above, trust is considered to be entirely absent. Whereas, in Proof of Authority (PoA) based systems such as VeChain, a small number of nodes are trusted as validators. The security of the blockchain system relies on the honesty of this subset of nodes. PoA based blockchains are favored by permissioned or consortium blockchains where trust is not completely deficient.

In the current digital era, an enormous amount of personal data is being collected by central entities (such as governments, institutions, and corporations). This data includes biographical data, professional activities data, medical data, mobility and travel data, communication and social interaction data, utility usage data, financial data including daily expenses, opinion and political views, images and videos, and even physical activities data (such as sleep cycles, heart rate, number of steps taken, etc. through fitness trackers).

The collection and analysis of such diverse and large amounts of data has allowed the central entities to offer helpful and innovative services. As an example, analysis of personal data has even resulted in life saving diagnosis of rare diseases [286]. However, the utility of these services often comes at the expense of the **privacy** of the users. Central entities may learn information about individuals that may be considered sensitive.

Sharing personal data is essential for the functionality of many distributed systems whether they be centralized or decentralized. Some decentralized systems that we consider in this work include reputation systems, voting systems, message routing in mobile delay tolerant networks, and peer-to-peer lending. In each of these systems, users must share their personal data in order to benefit from their functionality in the long term. In reputation systems, the users are required to express their personal opinion in order to establish the reputation of entities that would eventually help the community. In voting systems, users submit their personal choice in order to reach majority consensus. In certain social-interaction-based message routing protocols, users are required to share their mobility and social interaction patterns in order to help efficient delivery of messages for the users of the network. In peer-to-peer lending, the borrower must demonstrate to the lender their creditworthiness by divulging their credit history or other personal information.

Our broad goal is to enable users to avail useful services provided by decentralized systems while preserving their privacy. Looking back at the systems mentioned above, we could ask the following questions: Could we compute reputation without the users revealing their feedback? Could we determine the legitimate majority choice in a decentralized manner without anybody learning the votes? Could a message be routed taking advantage of the mobility patterns of users without them divulging their behaviors? Could a lender establish confidence in the intention and the ability of a borrower to reimburse a loan without acquiring their personal information? These are some of the themes that we address in our work.

We are particularly interested in privacy-preserving decentralized protocols. In these protocols, there are n nodes that need to compute a function such that each user holds a confidential input. This is a reflection of the problem that is studied by the field of secure multi-party computation. Privacy in such a decentralized system can be trivial to achieve when trust is not deficient. Let's assume that the n nodes all trust a single entity (a trusted third party, or TTP) to uphold their privacy, and to perform computations correctly and fairly. This TTP does not need to be centralized. It can be chosen by the nodes for a given instance of the protocol. In this case, all n nodes can submit their inputs to the TTP, who then computes the function and discloses the outcome. None of the n nodes learn the confidential inputs of the fellow nodes (except what can be inferred from the output).

However, in a system where trust is deficient, achieving privacy becomes a challenge since the above solution is no longer viable. In trust deficient systems, we need to rely on a combination of partial trust (depending on the level of trust deficiency) and cryptographic building blocks, such as verifiable secret sharing, homomorphic encryption, zero-knowledge proofs, etc. (discussed in Section 2.3.2). In trustless systems where trust is considered to be absent, we need to turn to proofs and verification. However, there do not exist generic efficient solutions for privacy preservation in trust-deficient decentralized systems. We need to develop protocols that optimize resource utilization while guaranteeing correctness and privacy.

1.1 Contributions

In this habilitation thesis, we include some selected contributions from our work published in the last 10 years, i.e., from the year 2013 to the year 2022. These selected contributions can be grouped under three main categories that we summarize below.

1.1.1 Privacy-Preserving Reputation Systems

The purpose of a reputation system is to hold the users of a distributed application accountable for their behavior. The reputation of a user is computed as an aggregate of the feedback provided by fellow users in the system. Truthful feedback is clearly a prerequisite for computing a reputation score that accurately represents the behavior of a user. However, it has been observed that users can hesitate in providing truthful feedback because, for example, of fear of retaliation. Privacy-preserving reputation systems enable users to provide feedback in a private and thus uninhibited manner.

- We present a comprehensive **survey of privacy-preserving reputation systems with emphasis on blockchain-based decentralized systems**. Blockchain-based privacy-preserving reputation systems have properties, such as trustlessness, transparency, and immutability, which prior systems do not have. We propose analysis frameworks that we use to review and compare the existing systems. Our analysis provides several insights and directions for future research. These include leveraging blockchain to its full potential in order to develop truly trustless systems, to achieve some important security properties, and to include defenses against common attacks that have so far not been addressed by most current systems.

- We develop a **decentralized privacy-preserving reputation protocol for the malicious adversarial model**. The malicious users in this model actively attempt to learn the private feedback values of honest users as well as to disrupt the protocol. Our protocol does not require centralized entities, trusted third parties, or specialized platforms, such as anonymous networks and trusted hardware. Moreover, our protocol is efficient. It requires an exchange of $O(n + log\ N)$ messages, where n and N are the number of users in the protocol and the environment respectively. This is a significant improvement because comparable protocols in the literature require as much as $O(n^3 + N)$ messages using similar building blocks.

- **Privacy-preserving reputation systems used in participatory sensing applications** have two seemingly conflicting objectives of monitoring the behavior of the participants over subsequent interactions and yet unlinking those subsequent interactions in order to preserve participants' privacy. We define an attack called Reputation-based Re-identification (RR attack) that exploits this conflict in order to detect the succession of contributions provided by the same participant and to re-identify their original identity. We show that using this attack, more than 35% of contributions can be associated to their successive contributions in each campaign. We then propose PrivaSense as a new privacy-preserving reputation system that integrates both reputation and privacy such that their objectives are simultaneously achieved. Experiments conducted using a real dataset show that PrivaSense decreases the number of contributions linked to their original providers by up to 80%.

- **Secure electronic voting** is a vision for the future of elections and referendums. However, this vision brings transparency and confidentiality requirements that render the design of such solutions challenging. Specifically, the counting must be implemented in a reproducible way and the ballots of individual voters must remain concealed. We propose a voting protocol (as a generalized instance of a privacy-preserving reputation protocol) that ensures **transparency, confidentiality, and integrity in a decentralized trust-deficient setup**. The persistence and immutability of the protocol's communication allow verifiability of the outcome by the voters themselves. The protocol is built by combining Secure Multi-Party Computation (SMPC) and decentralized distributed ledger technology.

- Our contribution on secure voting is further extended in our work on **collusion-resistant worker set selection for transparent and verifiable voting**. Collusion between workers can be particularly harmful to the security of the protocol. We propose protocols that select a subset of workers (who process data during the protocol execution) from the set of participants such that the risk of the workers colluding together is minimized. Our first solution is a decentralized protocol that randomly selects workers in a verifiable manner without any trusted entities. The second solution is an algorithm that uses a social graph of participants and community detection to select workers that are socially distant in order to reduce the risk of collusion.

1.1.2 Privacy-Preserving Message Routing

The pervasiveness of mobile devices with networking capabilities led to the emergence of Mobile Delay Tolerant Networks (MDTNs). These networks are often decentralized in nature and participating nodes communicate without the help of fixed infrastructure. The characteristics of MDTNs, which include frequent and long-term partitions, make message routing a major challenge.

- Message routing in mobile delay tolerant networks inherently relies on the cooperation between nodes. In most existing routing protocols, the participation of nodes in the routing process is taken as granted. However, in reality, nodes can be unwilling to participate. We **study the impact of the unwillingness of nodes to participate in existing routing protocols** through a set of experiments. Results show that in the presence of even a small proportion of nodes that do not forward messages, performance is heavily degraded. We then analyze two major reasons of the unwillingness of nodes to participate, i.e., their rational behavior (also called selfishness) and their **wariness of disclosing private mobility information**. We conduct experiments to compare the performance of existing strategies for preventing different types of selfish behavior. For protocols that preserve the privacy of users, we classify the existing approaches and provide an analytical comparison of their security guarantees.

- A prediction-based routing protocol for MDTNs functions by forwarding a message from one intermediate node to another if the latter has higher probability of encountering the destination node. However, this process compromises the privacy of the nodes by revealing their mobility patterns. We propose the **Privacy-Preserving Probabilistic Prediction-based Routing (4PR) protocol** that forwards messages by comparing information about communities of nodes instead of individual nodes. The privacy of individual nodes is preserved since only aggregate information about a community as a whole is disclosed. The 4PR protocol is an improvement over our prior protocol called 3PR. The 4PR protocol provides better privacy-preservation as well as higher efficiency.

1.1.3 Privacy Preservation in Financial Networks

There are many Decentralized Finance (DeFi) Peer-to-Peer (P2P) lending platforms that offer users to obtain a loan by committing a collateral or by calculating a credit score, which is based on factors such as the users' credit history. However, the requirements of collateral and credit history are quite burdensome for certain groups of users. Although these platforms are innovative in the sense that they are decentralized, they end up relying on the same loan-securing approaches as traditional banks and lending institutions. Therefore, a portion of the population is deprived from loan opportunities on these DeFi P2P lending platforms as well.

- We propose to use personal social media data as an alternative risk mitigator for lending. Users' professional behavior and reliability may be gleaned from such data allowing us to predict their trustworthiness in the lending context. We develop an Ethereum blockchain-enabled **decentralized lending platform** that calculates a "social score" based on the social media data of a user. Privacy considerations are taken into account in order to **protect the borrower's privacy** despite the use of personal and social data.

Another application that we address is Supplier Impersonation Fraud (SIF) detection in the Business-to-Business (B2B) context. This type of fraud occurs when a company supplying goods and services to another company is impersonated by a fraudster in order to trigger a payment to an illegitimate bank account. Information pertaining to successful fraud events against a company is sensitive and therefore there is little exchange between businesses of pertaining data. Absence of useful data poses an obstacle in the development of innovative data-driven fraud detection systems.

- We introduce **GraphSIF, a supplier impersonation fraud detection system** that aims to infer knowledge from the relational properties created by the transactions of a company. GraphSIF analyzes data that has been heavily anonymized in order to preserve the privacy of the participating companies. First, a sequence of graphs modeling the links between companies and accounts is created by aggregating the transactions. It is called the behavior graph. A targeted transaction's legitimacy is asserted by analyzing the patterns formed when it is added to the most recent graph of this sequence. Due to the potentially high number of patterns that can be found in the behavior sequence, a Self-Organizing Map is used to regroup graphs with similar patterns. The classification of a targeted transaction is performed by first clustering the graphs, and then comparing the similarity between the targeted transaction's graph with the other graphs of its cluster. The model shows good efficiency in terms of computational time needed to create the behavior sequence and to classify the transactions.

1.2 Outline

Chapter 2, the next chapter in this part, discusses some general concepts as the background for the rest of the work. The parts II, III, and IV of this habilitation thesis correspond respectively to each of the categories of contributions discussed in the previous section. Part V comprises of Chapter 12 and 13 that summarize the contributions and present directions for future research respectively. The outline of the next three parts is given below. Each chapter in these three parts is an adapted and abridged version of the published article that is indicated in the outline.

	PART II – Privacy-Preserving Reputation Systems
Chapter 3	**Privacy Preserving Reputation Systems based on Blockchain and other Cryptographic Building Blocks: A Survey.** O. Hasan, L. Brunie, E. Bertino. *ACM Computing Surveys.* 2023. Vol. 55, no. 2, article 32, pp 1-37. Published online: January 2022.
Chapter 4	**A Decentralized Privacy Preserving Reputation Protocol for the Malicious Adversarial Model.** O. Hasan, L. Brunie, E. Bertino, and N. Shang. *IEEE Transactions on Information Forensics and Security.* 2013. Vol. 8, no. 6, pp 949-962.
Chapter 5	**PrivaSense: Privacy-Preserving and Reputation-Aware Mobile Participatory Sensing.** H. Mousa, S. B. Mokhtar, O. Hasan, L. Brunie, O. Younes, and M. Hadhoud. *In Proceedings of the 14th Intl. Conference on Mobile and Ubiquitous Systems.* November 2017. Pp. 38-47.
Chapter 6	**A Transparent Referendum Protocol with Immutable Proceedings and Verifiable Outcome for Trustless Networks.** M. Schiedermeier, O. Hasan, L. Brunie, T. Mayer, and H. Kosch. *In Proceedings of the 8th Intl. Conference on Complex Networks and their Applications.* December 2019. Pp. 647-658.
Chapter 7	**Collusion-Resistant Worker Set Selection for Transparent and Verifiable Voting.** M. Bettinger, L. Barbero, O. Hasan. *SN Computer Science (Springer Nature).* 2022. Vol. 3, no. 5, article 334.

PART III – Privacy-Preserving Message Routing	
Chapter 8	**An Investigation on the Unwillingness of Nodes to Participate in Mobile Delay Tolerant Network Routing.** J. Miao, O. Hasan, S. B. Mokhtar, L. Brunie, and K. Yim. *International Journal of Information Management (Elsevier).* 2013. Vol. 33, no. 2, pp. 252-262.
Chapter 9	**4PR: Privacy Preserving Routing in Mobile Delay Tolerant Networks.** J. Miao, O. Hasan, S. B. Mokhtar, L. Brunie, and A. Hasan. *Computer Networks (Elsevier).* 2016. Vol. 111, pp. 17-28.

PART IV – Privacy Preservation in Financial Networks	
Chapter 10	**Privacy Considerations for a Decentralized Finance (DeFi) Loans Platform.** J. Hartmann and O. Hasan. *Cluster Computing (Springer).* 2022. https://doi.org/10.1007/s10586-022-03772-3.
Chapter 11	**GraphSIF: Analyzing Flow of Payments in a Business-to-Business Network to Detect Supplier Impersonation.** R. Canillas, O. Hasan, L. Sarrat, and L. Brunie. *Applied Network Science (Springer).* 2020. Vol. 5, article 40.

Chapter 2

Background

2.1 Decentralized Systems

We begin by briefly describing the general concept of distributed systems, which encompass decentralized systems.

2.1.1 Distributed Systems

As defined by Steen and Tanenbaum [272], a distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system. An autonomous computing element, which may also be referred to as a node, is an instance of a hardware device or a software process that exists and acts independently from others. The nodes of a distributed system collaborate with each other in order to provide some expected functionality to the user. From the user's point of view, the distributed system acts as a single coherent entity, even though it is composed of a collection of independent nodes.

Distributed systems can be categorized according to their system architecture, which includes centralized, federated, and decentralized architectures. The system architecture of a distributed system guides how its software components are distributed and organized over nodes. It also dictates the authority that the various components exercise over other components. The centralized architecture assumes the presence of a central entity that provides services to nodes that consume those services. On the other hand, the decentralized architecture considers all nodes to assume the role of providing services to each other. In the next sections, we take a more in depth look at distributed systems based on the decentralized architecture.

2.1.2 An Overview of Decentralized Systems

The organization of nodes in a decentralized system may be structured, unstructured, or hybrid. In a structured system, the nodes are organized in a pre-defined topology. This means that a given node (often also referred to as a peer in the decentralized architecture) has specific neighbors that it can interact with as part of the system. The topology may be in the form of a structure such as a ring or a tree. The purpose of the structure is to allow efficient access to nodes for a target use case. In contrast, an unstructured system allows nodes to interact with any other nodes as well as to change neighborhoods and form new neighbors. This aspect of unstructured systems provides complete flexibility to the nodes, however, the absence of structure also means lack of efficiency in communication in some use cases. Hybrid systems borrow aspects of both types of system, for example by imposing structure on a small set of nodes or assigning them specialized roles.

The decentralized architecture inherently favors certain properties that may be desirable in distributed systems. A few of these properties are listed below.

- **Fault tolerance.** In centralized systems, their exists a central entity or server, which is fundamental to the correct functioning of the entire system. The failure of the central entity can lead to the failure of the system as a whole. In contrast, decentralized systems do not consist of such centralized entities that have the potential to become single points of failure.
- **Attack resistance.** Central servers in centralized systems are easily identifiable high-value targets for attackers. Decentralized systems are generally more attack resistant since there may be hundreds or thousands of nodes that may need to be breached in order to compromise the security of the entire system.

- **Censorship resistance.** Centralized networks have choke points that make it possible to control the flow of information and censor information that is unfavorable to the central authority in control. Information is able to flow more freely in decentralized systems due to the absence of such choke points. Information may pass from node to node without the intervention of a central authority.
- **Higher autonomy.** The nodes in a decentralized system control their own resources and therefore they have higher autonomy than in centralized systems where resources are centrally managed.
- **Decentralized governance.** Decentralized systems have mechanisms for decentralized governance, which may lead to more democratic policy and decision making for the system. This is often not the case in centralized systems where the central authority can dictate policies and decisions to the member nodes.

Although decentralized systems offer many advantages as listed above, their design and implementation entails several challenges as well. We point out some of them below.

- **Lack of a single point of reference.** Central systems have a significant advantage that a central entity is present that can provide a single point of reference. This implies that coordination, synchronization, and ordering of tasks becomes relatively straight forward. Decentralized systems lack a single central entity, which makes organization more challenging.
- **Trust deficiency.** In centralized systems, the central entity is often well-known throughout the system and thus trusted to a certain degree to perform some critical functions correctly. In decentralized systems, their may exist a large number of nodes whose reputation is not known and consequently there is lack of trust in the services that they may provide. This trust deficiency can impact the willingness of the users to participate in the system. We note that the trust placed in a central entity can also be considered a liability since a breach of that trust can have a significant negative impact on the security of the users.
- **Concerns about privacy.** The central entity in a central system is also often trusted to protect the privacy of users according to their privacy policy. A direct consequence of the lack of trust in decentralized systems is that nodes are not trusted in the context of privacy protection either. However, we note that lack of privacy can be a major problem in centralized systems as well where an unscrupulous central entity could exploit the private information of its users.
- **Higher cost.** The cost of operations in terms of resources consumed is generally higher in decentralized systems. The operation may require processing by many distant nodes and results may need to be propagated to a large network of nodes. A given operation in a centralized system may require a few messages, a small amount of bandwidth, and a few processing cycles on the central server. In order to accomplish the same operation, the scale of the resources required can be much larger in a decentralized system.

2.1.3 Applications of Decentralized Systems

We now take a brief look at some categories of applications of decentralized systems. Our selected contributions that are presented in later chapters address applications from these categories.

- **Web3.** The World Wide Web (which we now call 'Web 1.0') was initially conceived as a means of sharing information using decentralized protocols. This version of the web was composed of static websites created by companies as well as users that were mostly read-only. The second and the current version of the web (now called 'Web 2.0') was mainly a consequence of the emergence of social networks in the early to mid 2000s, which are platforms that enable visitors to generate and share content as well. However, this model resulted in the centralization of information on a few platforms. The entities that own these platforms also own the user information. They need to be fully trusted for maintaining the information as well as for ethical concerns such as those related to monetization, privacy, etc. Web3 (or 'Web 3.0') is a vision for the re-decentralization of the web such that the users generate as well as own their information and make it available for use according to their own policies. Web3 is composed of decentralized protocols primarily based on the distributed ledger or blockchain technology that enable this vision. Some examples of these protocols are Bitcoin for payments, Ethereum for smart contracts, Inter-Planetary File System (IPFS) for decentralized storage, Non-Fungible Tokens (NFTs) for ownership of digital assets, etc. We discuss Web3 in further detail in Section 2.1.4.

- **Decentralized reputation systems.** Decentralized reputation systems enable determining the trustworthiness of users in decentralized environments where there is no pre-established trust in users. Reputation of a target user is computed by aggregating the subjective feedback provided by other users, referred to as source users. These are users who have previously interacted with the target user and have consequently gained personal experience regarding her actions in the context of the given application. Reputation systems thus assist in holding users accountable for their actions despite the initial absence of trust in the users.

- **Mobile delay tolerant networks.** Mobile Delay Tolerant Networks (MDTNs) are constructed by the intermittent connection of co-located mobile devices. The short range networking interfaces (e.g., Bluetooth) of these devices enable neighboring devices to interact through short-range communications in a decentralized manner without the help of fixed infrastructure. However, routing messages between two nodes that are not within communication range is a challenge in MDTNs since an end-to-end routing path cannot be guaranteed.

- **Mobile participatory sensing.** Mobile participatory sensing is an application where users sense various environmental conditions with their mobile devices and then aggregate the sensing data for statistical analysis. The data may be aggregated by a central entity or it may aggregated by the mobile devices themselves in a decentralized manner. A related application area is participatory sensing in Vehicular Adhoc Networks (VANETs), where vehicles collect information about road conditions.

2.1.4 Web3

A distributed ledger or blockchain is a data structure that is composed of a set of blocks linked by cryptographic hash pointers. The blocks are chronologically ordered. Each block comprises of the record of a set of transactions or operations that have recently taken place between the users.

As introduced in the previous section, Web3 is composed of decentralized protocols primarily based on the distributed ledger technology and aims to offer a decentralized version of the web. According to Jay Graber [125] and as discussed by Korpal and Scott [175], the difference between the subsequent versions of the web can be summarized as below.

- Web 1.0 – Host-generated content, host-generated authority
- Web 2.0 – User-generated content, host-generated authority
- Web 3.0 – User-generated content, user-generated authority

We look at two fundamental building blocks of Web3: 1) decentralized payments based on cryptocurrencies, and 2) decentralized applications based on smart contracts.

Decentralized Payments – Cryptocurrencies

Cryptocurrencies enable native decentralized payments in Web3 instead of relying on the traditional mostly centralized financial infrastructure. A large number of cryptocurrencies have been introduced since the advent of Bitcoin, which was the first successful decentralized cryptocurrency. Many subsequent cryptocurrencies have proposed improvements to the original designs of Bitcoin to achieve better scalability, interoperability, energy efficiency, privacy, etc. However, Bitcoin still remains the largest cryptocurrency in terms of market capitalization [21]. Bitcoin was the first cryptocurrency to utilize many of the fundamental ideas together upon which many of the later cryptocurrencies have been founded. In this section, we give an overview of some aspects of how Bitcoin achieves decentralization.

Bitcoin succeeds by solving several important challenges for a digital currency in a decentralized manner. This is done through the use of crytpographic and other technical building blocks as well as innovative incentive engineering. Some of these challenges addressed include:

- Maintaining an immutable and incontestable ledger of the ownership of coins.
- Allowing authentication of valid ownership of coins.

- Preventing a user from double spending a coin.
- Enabling efficient consensus on a single valid state of the ledger.
- Incentivizing nodes to volunteer computing resources for the upkeep of the ledger.
- Decentralized identity management.
- Preventing Sybil attacks.
- Creation and distribution of new coins.

A block that comprises of newly carried out transactions is appended at regular intervals to the Bitcoin ledger. Several nodes in the system may propose their own versions of the block to be appended next. The nodes in the Bitcoin system aim to reach consensus on a single state of the ledger where a unique block gets appended. The block must be valid and consequently must contain only valid transactions.

The above problem is an instance of the classic problem of distributed consensus. In this problem, n nodes in a distributed system each have an input value. Some of the nodes are considered to be malicious (or faulty or 'Byzantine'). A distributed consensus protocol must terminate with all non-malicious, i.e. honest nodes, in agreement on a single value, which has been proposed by an honest node. The absence of a common global clock poses several hurdles in this problem. An impossibility result by Lamport et al. [179] shows the impossibility of reaching consensus in the representative Byzantine Generals Problem when one third or a higher portion of the generals (or nodes) are malicious. In fact, another impossibility result by Fischer et al. [112] shows that consensus is impossible even if only one node is malicious or may simply crash.

As discussed by Narayanan et al. [222], Bitcoin overcomes the above impossibility results as it operates in a model that is different from the one in which the impossibility results were proven. Specifically, Bitcoin introduces financial rewards for the nodes to behave honestly. These incentives are practical since the notion of currency is built into the protocol, which is not the case in the model of the impossibility results. Moreover, Bitcoin accepts eventual probabilistic consensus instead of deterministic consensus. This implies that the nodes in the Bitcoin system may not be in agreement on the state of the ledger at a given time, however, in practice the majority is expected to eventually reach consensus on the state of the ledger up to a certain block.

Bitcoin utilizes the Proof of Work (PoW) algorithm that requires a hash puzzle to be solved by the node that wishes to propose the next block. The solution of the puzzle lies in finding a suitable nonce value. The hash of this nonce concatenated with the hash of the previous block and all its transactions must be a number that falls within a required target space. The solution to such a puzzle is non-trivial due to primarily two reasons: Firstly, since the hash function SHA-256 is a one-way hash function, there is no efficient way of deducing the solution. Secondly, the target space is very small as compared to the complete output space of the hash function. The node that solves the puzzle for the current block, gets to propose the block. Any node can efficiently verify the solution to be correct due to the low asymptotic complexity of hashing the solution string. Honest nodes accept the new block into their local copies of the ledger if all the transactions are correctly formed and do not contain cases of double or un-authorized spending. This mechanism allows 'implicit' consensus between the nodes since no explicit coordination is required. The node whose valid block is accepted into the Bitcoin blockchain gets rewarded with an incentive in new Bitcoins, therefore it is in the interest of nodes to propose blocks that are valid.

A user in Bitcoin is able to create their unique identity themselves without the involvement of a central authority. The user first generates a pair of public and private keys. A hashed derivative of the user's public key is considered as the address of that user. In order to authorize spending transactions from a given address, the user must digitally sign them using the underlying public key's corresponding private key that is known only to the user who is the legitimate owner of the coins. The private keys are drawn from the large interval of 256 bit integers, therefore, the probability of collisions between multiple users is considered insignificant. A user may generate as many identities as they wish. However, owning multiple identities does not endow a user with any disproportional advantage over other users and does not favor Sybil attacks. The reason is due to the utilization of the Proof of Work (PoW) algorithm discussed before, which is not impacted by the number of identities owned by a node. A user owning one or multiple identities can only gain an advantage to solve the hash puzzle and receive the Bitcoin reward by increasing their computational processing power.

Decentralized Applications – Smart Contracts

Bitcoin proposes a native scripting language that is used mainly for processing transactions to be recorded in the next block. The scripting language supports cryptographic operations such as hashing, checking that a public key hashes to a given address, checking the validity of digital signatures, etc. The Bitcoin scripting language is stack-based, which means that each instruction specified in a given script gets executed only once in the exact order in which the instructions are listed. This implies that it is not possible to implement loops and that the language is not Turing-complete. This is intended in the design of Bitcoin so that users are unable to submit scripts that take arbitrarily long to process or that can even potentially contain infinite loops. A node who is compiling the next block executes the script submitted for the processing of an associated transaction. Arbitrary scripts would have the potential to unfairly consume valuable resources and slow down the system. The consequence of only offering a primitive scripting language is that Bitcoin does not support the execution of general purpose programs. It is limited to only processing financial transactions and some applications that are able to work within the constraints imposed by the language.

Subsequent alternative blockchains, notably Ethereum, proposed the notion of smart contracts. The goal of smart contracts is to remove the limitation discussed above and support general purpose computing on a blockchain. A smart contract is a program that is written in a Turing-complete programming language and that can execute on the blockchain platform. This implies that any application can be built using smart contracts. The term smart contract is used because the program can be considered as a contract between the users. The terms and outcomes of the program are enforced by the blockchain platform. For example, users may submit coins to a smart contract and agree upon the terms of the distribution of those coins at a later moment under certain conditions. The smart contract would hold the coins and precisely follow the initially laid out terms and conditions without any intervention or cheating on behalf of the participating users.

Ethereum implements a decentralized embedded canonical computer that is called the Ethereum Virtual Machine (EVM). All nodes maintain a local copy of the state of the machine and update state changes at regular intervals. Computation on the EVM can be requested by any participating node by calling a function of a deployed smart contract. Requests for computation are called transactions in Ethereum. All nodes of the Etherem network verify the transaction and in case of a valid request, they execute the computation. The resulting state change in the EVM is propagated throughout the network. The state of the EVM is stored on the Ethereum blockchain. The nodes reach consensus on a single valid state of the blockchain.

Similar to Bitcoin, the identities are managed in a decentralized manner. Transactions that spend the native cryptocurrency, called Ether, must be signed by the user in order to authenticate ownership. Economic incentives are provided to the nodes that commit computational resources for the verification and execution of transactions. A transaction request i.e. a call to a function of a smart contract must be accompanied by a fee in Ether proportionate to the amount of computation that is required by the function. A reward or a bounty is awarded to the node who performs the processing required for verifying and executing the transaction and appending it to the blockchain. This concept of fees prevents malicious users from congesting the system by requesting arbitrarily lengthy computation. This countermeasure allows Ethereum to be able to support Turing-complete computation.

A decentralized application (dapp) is an application that provides a front end user interface with its backend logic implemented on smart contracts. This is in contrast to centralized applications in which the backend logic is hosted on centralized servers. Dapps inherit the benefits of decentralization and smart contracts. They offer high availability, censorship resistance, data integrity and immutability, transparency and verifiability of all computation, without reliance on any trusted third party.

The current version (called the London upgrade) of Ethereum employs a Proof of Stake (PoS) algorithm for the consensus mechanism instead of the PoW algorithm that Bitcoin uses. Nodes who wish to participate in verifying and creating new blocks are required to stake capital in Ether into a smart contract, which acts as collateral. These nodes are called validating nodes or validators. The validators receive rewards in Ether if they provide computational resources for validation purposes as well as behave honestly. On the other hand, they are penalized and their stake is destroyed if they do not make resources available when needed or behave dishonestly. The PoS algorithm is comparatively energy efficient.

2.2 Trust

In order to use the functionality of a distributed system, the users may be required to trust certain entities, such as a central authority, or some fellow users in the system. The trust implies a belief by the trusting user that the trusted entity or the trusted fellow users will behave in an expected manner.

2.2.1 Degrees of Trustfulness (or Inversely Trust Deficiency)

In our work on privacy-preserving reputation systems [136], we identified four different models of trust that users are required to engage in. These models are listed below in descending order according to the degree of trustfulness that is required. Inversely, the models vary in the level of trust deficiency that is assumed in the system. The first model is fully trustful, that is, it requires users to completely trust another entity. Whereas, the last model is trustless or in other words fully trust deficient, that is, no trust is assumed to exist between fellow entities.

- **Trusted Third Party.** A Trusted Third Party (TTP) for a set of users is an entity whom every user in the set trusts completely for certain actions. In this model, all users of the system must trust the designated TTP entities in the system. A user in a system who needs to be fully trusted is also considered as a TTP.

- **Trust on arbitrary k fellow users.** A user in the system is required to place her trust in k different fellow users for the security guarantees, where $k \leq n$, and n is the total number of users participating in the protocol. These k users are selected by the system without taking the user's preferences into account. Thus from the perspective of the user, the set of trusted users is selected arbitrarily. Generally, only a partial level of trust is required in each of the trusted users in this model.

- **Trust on chosen k fellow users.** In this trust model, a user in the system also places her trust in k distinct fellow users. However, these fellow users are chosen by the user herself. The user may select the trusted users based on the level of their subjective trustworthiness in order to maximize the security or privacy guarantees. This model requires that a user is able to determine the trustworthiness of fellow users and choose accordingly from a pool of available users.

  In the context of our work on privacy-preserving reputation systems [139], we note that there is a difference between choosing fellow users for establishing security guarantees versus choosing feedback providers for personalizing the reputation score of the target entity. In the first case, a user chooses fellow users who specifically influence the security and privacy guarantees that she would receive in the reputation system. In the latter case, there is no impact intended on the security guarantees. The "Trust on chosen k fellow users" model addresses choosing k users specifically for the purposes of security in the reputation system.

  As an example, consider the systems by Hasan et al. [139] and Gudes et al. [128]. In the system by Hasan et al., the selection of k fellow users is made for preserving privacy. The trust model of this system can thus be classified under the chosen k users category. In contrast, in the system by Gudes et al., even though a user selects a subset of fellow users, the system's trust model cannot be classified as the chosen k users model. The reason being that the selection of users in this latter system is made purely for personalizing the reputation score.

- **Trustless.** In the trustless model, the users in a system do not need to trust any entities or any fellow users. Thus, this model does not expect users to have pre-existing trust toward fellow users or entities in the system. The users need to rely solely on the underlying algorithms and protocols of the system and their verification in order to receive the security guarantees.

  However, we note that even though the users do not need to directly trust any entities or users in this model, there may exist a requirement of trustworthiness for the overall correct and secure working of the system. Trustless systems are based primarily on the blockchain technology. As an example, the Bitcoin blockchain requires that a majority of all participants in the system act honestly in order to ensure integrity.

  The trustless model may be considered a special case of the "Trust on arbitrary k fellow users" model, where k is at least greater than half of the total number of all participants in the entire system (not just a protocol instance). A blockchain system functions by building consensus among peers. In the case of Bitcoin, if a majority of peers are dishonest, consensus cannot be achieved and the entire system malfunctions. Thus, the breach of the trustworthiness requirement in such systems does not simply threaten the security of a given user but the integrity of the entire system. It is therefore in the collective interest of all honest users in the system to prevent any breach of trustworthiness.

2.2.2 Defining Trust

There has been extensive research on the concept of trust in many different domains, such as sociology, philosophy, social psychology, economics, and computer science. Therefore, a number of definitions of trust have been proposed in the literature with different perspectives. According to Marsh [205, page 20], a common element that links all studies on trust is the assumption of the presence of a society. In this section, drawn from our prior work in the area of trust management [133, chapter 2], we present some of the influential definitions of trust that appear in the literature.

One of the earlier notable definitions of trust is formulated by social psychologist Morton Deutsch [92]. The definition states that when:

> 1. "the individual is confronted with an ambiguous path, a path that can lead to an event perceived to be beneficial ($Va+$) or to an event perceived to be harmful ($Va-$);
> 2. he perceives that the occurrence of $Va+$ or $Va-$ is contingent on the behavior of another person; and
> 3. he perceives the strength of $Va-$ to be greater than the strength of $Va+$.
>
> If he chooses to take an ambiguous path with such properties, I shall say he makes a trusting choice; if he chooses not to take the path, he makes a distrustful choice."

We interpret Deutsch as follows: Trust and distrust are discrete, distinct choices that arise when an individual must rely on another person to gain a benefit. However, there is also a possibility that relying on that person may actually lead to harm instead of benefit. The individual trusts that person if he chooses to rely on him, and distrusts him if he chooses otherwise.

In [116], sociologist Diego Gambetta proposes the following definition of trust:

> Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action.

This is one of the seminal definitions that describe trust as a quantifiable construct. Gambetta observes that trust is an agent's degree of belief (the level of subjective probability) that another entity will perform an expected action. An additional important aspect of this definition is the recognition that trust is contextual.

McKnight et al. [207], a prominent work in the field of trust in information systems, regards trust as a multidimensional concept composed of two constructs: trusting *intention*, and trusting *beliefs*. Trusting intention means that one is willing to depend on the other person in a given situation, and trusting beliefs imply that one believes the other person to be *competent*, *benevolent*, and having *integrity*. Competence is the ability of the trustee to do what the truster needs, benevolence is the trustee's caring and motivation to act in the truster's interests, and integrity is the trustee's honesty and promise keeping.

Grandison and Sloman [126] discuss trust from a computer science perspective. They consider trust to be a composition of many different attributes such as reliability, dependability, honesty, truthfulness, security, competence, and timeliness. Their definition of trust is as follows:

> The firm belief in the competence of an entity to act dependably, securely and reliably within a specified context (assuming dependability covers reliability and timeliness).

Jøsang et al.'s survey on trust and reputation systems [160] differentiates between two types of trust: reliability trust, and decision trust. Reliability trust, which is inspired by Gambetta's definition, is stated as:

> Trust (reliability trust) is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends.

This definition stresses on the *dependence* on a trustee, and the *reliability* (probability) of the trustee. However, Jøsang et al. concur with Falcone and Castelfranchi [108], and McKnight and Chervany [206] in recognizing that having high reliability trust in a person is not necessarily enough to decide to enter into a situation of dependence on that person. For example, if the stakes are too high, then even a small probability of failure may lead an individual to decide to not depend on a potential trustee. Thus, trust could be viewed as the decision to depend or not depend on another entity. Jøsang et al. summarize *decision trust* as:

> Trust (decision trust) is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.

2.3 Privacy

In this work, we are mainly interested in protocols where a number of nodes wish to collaboratively compute some function on inputs that they provide. This scenario gives a general description of many protocols in distributed systems. For example, in an election protocol, individual voters submit votes as their inputs and seek to compute a function that produces the winner of the election as its output.

2.3.1 Privacy-Preserving Protocols

A protocol can be said to be privacy-preserving if the participating nodes do not learn any information from the execution of the protocol other than the output itself and what can be inferred from the output. One of the main goals of privacy preservation in protocols is to protect the confidentiality of the sensitive information of participating nodes. In the example of an election protocol given above, the protocol could be considered as privacy-preserving if its execution does not reveal confidential information such as the individual votes of the nodes.

In addition to not revealing unnecessary information, we expect a privacy-preserving protocol to be overall secure. The security of a protocol goes beyond protecting confidentiality. Secure Multi-Party Computation (SMPC) is an area of cryptography that studies the security of privacy-preserving protocols. In SMPC, we assume that the protocol operates under a given adversarial model where a malicious entity has corrupted a set of the participating nodes (or parties) and controls them in order to learn private information or even to prevent correct execution of the protocol. We summarize below a set of properties that a secure protocol must achieve as discussed by Lindell and Pinkas [190]. However, we note that the list of properties does not constitute a complete definition of security. We will look at a more precise formulation of security later in this section with the discussion of the ideal/real simulation paradigm.

- **Privacy.** As discussed earlier, the protocol should not divulge information about the inputs of the participating parties other than what can be derived from the output itself. We take the example of an auction protocol where the execution of the protocol is expected to disclose the bid of the the highest bidder. In this case, it is straightforward to derive that the non-winning bids are lower than the winning one. The disclosure of this information is inherent in the protocol, however, in order to be considered as privacy-preserving, the protocol should not reveal any further information.

- **Correctness.** The protocol should guarantee that the output computed is correct and that each participating party receives that correct output. In the example of the auction protocol, this property implies that only the party who has submitted the legitimate highest bid can be declared the winner. No other party with a lower bid is able to cheat and become the winner.

- **Independence of inputs.** A corrupted party is unable to choose an input such that it is derived from the inputs of honest parties. In the example above, the bids in the auction are supposed to be secret and independent of each other. Deriving a bid that is higher than others gives the malicious party unfair advantage. We note that the property of privacy does not guarantee the property of independence of inputs. For example, it is possible in some homomorphic cryptosystems to derive an encryption of $x + 1$ given an encryption of x without learning the value of x.

- **Guaranteed output delivery.** The protocol should guarantee that the adversary will not be able to prevent the computation of the output of the protocol. The adversary should not be able to carry out a denial of service attack in order to disrupt the complete execution of the protocol.

- **Fairness.** This property implies that the corrupted parties should receive the outputs of the protocol if and only if the honest parties receive their outputs as well. Guaranteeing this property is critical in some scenarios. For example, in the case of a contract signing protocol, both participating parties must receive the signed contract.

The above list of properties are the general requirements for a secure protocol. However, the security of a privacy-preserving protocol needs to be defined in a more precise manner. Listing a set of properties is not adequate for a complete definition of security because it is possible to miss some properties and unforeseen attacks in this approach. Therefore, the approach used in SMPC is to define the security of a protocol in a practical or real model in relation to a similar fully secure protocol that exists in a theoretical or ideal model. This formulation of security of protocols is termed as the ideal/real simulation paradigm. We give below a brief informal description of this formulation of security as discussed by several authors [70, 121, 190, 295] in this area as well as in our prior work [133].

Intuitively, a multi-party protocol in the ideal model is a protocol that comprises of a Trusted Third Party (TTP) as a participant. The TTP receives all inputs in the protocol and then locally computes the output. On the other hand, a multi-party protocol in the real model is a protocol that does not rely on a TTP and computes the output in a distributed manner.

The privacy preservation property of a real protocol $\mathcal{R}$ under a given adversarial model is stated as follows: An ideal protocol $\mathcal{I}$ is first defined which has the same functionality as the protocol $\mathcal{R}$. This means that the ideal protocol has the same parameters (participants, inputs, outputs, etc.) as the protocol $\mathcal{R}$, with the exception of a TTP as an additional participant. In the ideal protocol, the TTP receives the private inputs of the participants, computes the output, and sends it to the recipients. It is assumed that the privacy of the participants is preserved since the TTP is honest and fully trusted. In an ideal protocol the adversary cannot obtain any more information about the private input of a participant other than what it can learn from the information that it has beforehand and the output of the protocol that it receives. The real protocol, which cannot consider a TTP as a participant, is said to preserve privacy if it can emulate the ideal protocol. Emulating the ideal protocol essentially means that the adversary, cannot obtain any more information about the private input of an agent than it can learn in the ideal protocol.

There is a very limited number of actions that are available to the participating parties in the ideal model. The honest as well as the corrupted parties can only send their inputs to the TTP and then receive the outputs. The corrupted parties have no interaction with the honest parties therefore they are unable to obtain any information directly from them. Moreover, the TTP performs all the computations locally without outside interaction and is considered to do so in a fully trusted and incorruptible manner. Therefore, this setup does not allow the adversary to collect any information other than the output. The only manipulation that the adversary can perform is to alter the inputs of the corrupted parties. However, the TTP still computes the correct output considering the submitted inputs, which is what is expected from a secure protocol.

In the corresponding protocol in the real model, whose security is to be analyzed, the parties compute the same function without the assistance of a trusted and incorruptible third party. The participating parties that include the honest and corrupted nodes must interact and share some intermediate information with each other. This practical setup creates opportunities for the adversary to derive inferences as well as for disruptions to the protocol. This protocol is said to be secure if it can be shown that the adversary cannot achieve more success than it can in the ideal model. More specifically, the outcome of a real protocol execution is compared to the outcome of the execution of the corresponding ideal protocol in order to analyze its security. The protocol is determined to be secure if for any adversary that successfully attacks the real protocol, there also exists an adversary in the ideal protocol such that the execution of both protocols results in the same distributions of the input and output values of all participants (honest as well as corrupted). This implies that the adversaries in the ideal model are able to simulate the executions of the corresponding protocol in the real model.

2.3.2 Cryptographic Building-Blocks for Privacy-Preserving Protocols

There exist a number of useful cryptographic building blocks that can be used for constructing privacy-preserving protocols. These include homomorphic cryptosystems, zero-knowledge proofs, verifiable secret sharing, blind signatures, ring signatures, Trusted Execution Environments (TEEs), etc. among many others. We give below an overview of homomorphic cryptosystems and zero-knowledge proofs, which are some of the building blocks that we use in our systems.

Homomorphic Cryptosystems

Homomorphic cryptosystems are useful for privacy-preserving protocols as they allow operations on multiple values in their encrypted form. The operations can be carried out while obfuscating individual values.

Let $E_u(.)$ denote an encryption function with the public key PK_u of user u in an asymmetric cryptosystem $\mathcal{C}$. The cryptosystem $\mathcal{C}$ is said to be additive homomorphic if we can compute $E_u(x+y)$, given only $E_u(x)$, $E_u(y)$, and PK_u. In other words, a cryptosystem is additive homomorphic if we can compute the encryption of the sum of two plaintexts, given only their ciphertexts and the encrypting public key.

The Paillier cryptosystem [230] is a well-known additive homomorphic cryptosystem. Similarly, a multiplicative homomorphic cryptosystem such as the ElGamal Cryptosystem [107] allows computation of the encryption of the product of two plaintexts from their ciphertexts and the encrypting public key. The Paillier and ElGamal cryptosystems are classified as partial homomorphic cryptosystems as they only allow a single specific operation.

More recent homomorphic cryptosystems such as CKKS (Cheon, Kim, Kim and Song) [77] and GSW (Gentry, Sahai, and Waters) [117] are fully homomorphic. This means that a cryptosystem allows multiple types of operations, for example addition as well as multiplication.

Zero-Knowledge Proofs

A zero-knowledge proof [122] is an interactive proof that allows a prover to convince a verifier that a statement is true without revealing any information other than the fact that the statement is valid. Zero-knowledge proofs are helpful in building privacy-preserving protocols as they enable a node to demonstrate that an input or the result of an intermediate computation is correctly formed while keeping its value private.

As an example, consider a prover who knows an RSA modulus n and its two large prime factors p and q. A verifier knows only n. Factoring n is considered intractable therefore the verifier cannot learn p and q. An interactive proof would be zero-knowledge if it allows the prover to convince the verifier that he knows the factors of n without revealing any information about p and q.

A standard interactive zero-knowledge proof comprises of three moves, that is, three messages exchanged between the prover and the verifier. In the first move, the prover sends a cryptographic commitment to the verifier. In the second move, the verifier sends a random challenge to the prover to test the commitment. The third move is the prover's response to the random challenge of the verifier.

An interactive zero-knowledge proof can be converted to a non-interactive zero-knowledge proof using the Fiat-Shamir heuristic [110]. A non-interactive zero-knowledge proof comprises of only one move, that is, one message sent by the prover to the verifier. An interactive zero-knowledge proof requires three moves because the verifier must provide a random challenge to the prover. In the non-interactive version, this random challenge is replaced with a hash of the prover's commitment that the prover can generate itself. The prover generates the commitment, the challenge as the hash of the commitment, and the response to the challenge. It then sends everything in one move to the verifier. In turn, the verifier independently computes the hash and verifies the proof.

zk-SNARK (zero-knowledge Succinct Non-interactive ARgument of Knowledge) [53] and zk-STARK (zero-knowledge Scalable Transparent ARgument of Knowledge) [46] are two popular types of non-interactive zero-knowledge proof systems that are currently in use. zk-SNARK systems have the advantage that they are relatively efficient in terms of the proof size as well as proof and verification times. Proof size is particularly important for decentralized blockchain-based systems, where storing information on the blockchain can be costly. However, zk-SNARK systems require a trusted setup, which means that when the keys are initially generated for the system, the participants need to be trusted to perform certain operations correctly. Otherwise, the system can lead to false proofs and verifications. zk-STARK systems avoid the requirement of trusted setup and are thus well-suited for systems where trust is deficient. On the other hand, zk-STARK systems are less efficient in terms of proof size and verification times [36], therefore their use can incur higher costs.

Part II

Privacy-Preserving Reputation Systems

Chapter 3

A Survey of Decentralized Privacy-Preserving Reputation Systems

The purpose of a reputation system is to hold the users of a distributed application accountable for their behavior. The reputation of a user is computed as an aggregate of the feedback provided by fellow users in the system. Truthful feedback is clearly a prerequisite for computing a reputation score that accurately represents the behavior of a user. However, it has been observed that users can hesitate in providing truthful feedback because, for example, of fear of retaliation. Privacy-preserving reputation systems enable users to provide feedback in a private and thus uninhibited manner. In this survey, we propose analysis frameworks for privacy-preserving reputation systems. We use these analysis frameworks to review and compare the existing approaches. Emphasis is placed on blockchain-based systems as they are a recent significant development in the area. Blockchain-based privacy-preserving reputation systems have properties, such as trustlessness, transparency, and immutability, which prior systems do not have. Our analysis provides several insights and directions for future research. These include leveraging blockchain to its full potential in order to develop truly trustless systems, to achieve some important security properties, and to include defenses against common attacks that have so far not been addressed by most current systems.

This chapter is an adapted version of the article: "Privacy Preserving Reputation Systems based on Blockchain and other Cryptographic Building Blocks: A Survey." O. Hasan, L. Brunie, E. Bertino. *ACM Computing Surveys.* 2023. Vol. 55, no. 2, article 32, pp 1-37. Published online: January 2022.

3.1 Introduction

Reputation systems are an essential tool for determining the trustworthiness of users in environments where there is no pre-established trust in users. Reputation of a *target* user is computed by aggregating the subjective feedback provided by other users, referred to as *source* users. These are users who have previously interacted with the target user and have consequently gained personal experience regarding her actions in the context of the given application. It is expected that actions perceived as legitimate will lead to high positive feedback and thus to an aggregated positive reputation score. Inversely, a target user acting dishonestly will elicit negative feedback resulting in a low aggregated reputation score. Any users concerned about the legitimacy of future actions of a potential transacting partner can consider the computed reputation score of this partner as an indication of her trustworthiness. Reputation systems thus assist in holding users accountable for their actions despite the initial absence of trust in the users.

E-commerce marketplaces and sharing economy based platforms are some popular applications where reputation systems are employed. Sites and mobile applications such as ebay.com, airbnb.com, and uber.com are significant examples. Additionally, systems by Liu et al. [192], Azad et al. [39], Bag et al. [42], and Schaub et al. [255] are some of the academic approaches for managing reputation in e-commerce and retail environments. Let's consider Airbnb (airbnb.com), which is an online marketplace for vacation rentals. The platform enables independent hosts to offer their private lodgings to guests for short stays. The reputation system of the platform plays a critical role since the guests seeking satisfactory accommodations can only rely on the reputation of the hosts and their offerings stemming from reviews by previous guests. Similarly, hosts concerned about lending out their lodgings to well-behaving guests also depend on the reputation system.

Another application that relies on reputation systems is mobile participatory sensing, where users sense various environmental conditions with their mobile devices and submit sensing data to a central entity for analysis. Reputation is used to discourage users from providing corrupted information. Systems by Jo and Choi [159], Ma et al. [200], and Mousa et al. [218] are examples of reputation systems that focus on this application area. A related application area is participatory sensing in Vehicular Adhoc Networks (VANETs), where vehicles collect and upload information about road conditions. Reputation systems by Zhao et al. [296], Lu et al. [198], and Chen et al. [76] aim to hold the vehicles and their owners accountable for submitting false data. One more notable application area, among several others, that relies on reputation systems is the Internet of Things (IoT). Trusting corrupted devices in the IoT can undermine network security [40]. Recent systems by Azad et al. [38, 40] are instances of reputation systems for this application domain.

It has been documented that users may hesitate to provide truthful feedback [213, 247]. Reasons range from fear of retaliation to negative reviews [213, 247] to concerns about revealing sensitive personal information [215]. Returning to the example of Airbnb, we note that its reputation system escrows the feedback until both parties have submitted their opinion. The reason is to prevent tit for tat retribution by the hosts and the guests. However, the truthfulness and the impartiality of user feedback can still be impacted by the personal nature of the reviews [145]. Hiding the identities of the users has been recommended as a solution [145]. Moreover, it has been observed that the lack of anonymity on Airbnb "causes people to feel pressure to post reviews that lean positive" [221].

Privacy-preserving reputation systems are designed to allay the fears of feedback providers by protecting the confidentiality of their individual feedback. The implication is that providing feedback in a private manner encourages the raters to submit honest and accurate feedback. Another approach that privacy-preserving reputation systems take to motivate users to submit their truthful feedback is guaranteeing their anonymity. Operating in an anonymous manner in the system means that a third party is unable to attribute sensitive personal information to the user or to profile the user in the long term. Privacy-preserving reputation systems are therefore an important category of reputation systems for scenarios where user privacy or anonymity needs to be upheld.

The research area of privacy-preserving reputation systems is fairly mature. All academic reputation systems cited above are in fact privacy-preserving. Reputation systems that support user privacy were first proposed in the mid 2000s. Some notable original works include those by Pavlov et al. [235], Kinateder and Pearson [173], and Dingledine et al. [99], among others. However, privacy-preserving reputation systems continue to evolve to cater for emerging application areas, such as Social IoT (Azad et al. [40]), Industrial IoT-enabled retail marketing (Liu et al. [192]), and Intercloud (Dou et al. [103]). Moreover, the advent of the blockchain technology has recently fueled further research in this area. The use of blockchain as a building block has resulted in privacy-preserving reputation systems that have important novel properties such as trustlessness, transparency, and immutability. For example, Schaub et al.'s [255] system does not require users to trust third parties or any fellow users in order to guarantee their privacy and thus provides trustlessness. This property was absent from prior systems. Another important reason for recent research in the area of privacy-preserving reputation systems is that a number of issues still remain open. As we discuss in Section 3.8, these issues include lack of important security properties and defenses against common attacks.

We believe that a comprehensive survey is needed to offer a uniform perspective to the rich literature in this area. Moreover, we believe that our survey is timely because of the recent emergence of systems based on blockchain as well as novel systems for emerging application domains. In this survey, we analyze 44 privacy-preserving reputation systems proposed between the years 2003 and 2021 inclusive, while placing emphasis on recent systems based on blockchain.

Reputation systems that support user privacy have always mostly relied on cryptographic building blocks and their combinations to provide strong security guarantees. These building blocks include Secure Multi-Party Computation (SMPC), secret sharing, homomorphic encryption, zero-knowledge proofs, and cryptographic signatures. Blockchain is a recent addition to this arsenal of cryptographic building blocks utilized by privacy-preserving reputation systems. We analyze blockchain-based systems as well as systems based on other building blocks and security mechanisms in this survey.

3.1.1 Contributions

This survey makes the following contributions:

- Identification of the various dimensions of privacy-preserving reputation systems. An analysis framework that allows for the decomposition and comparison of privacy-preserving reputation systems in a normalized manner.
- Identification of the security requirements of privacy-preserving reputation systems that cut across multiple types of these systems.
- Definition of broad categories of privacy-preserving reputation systems proposed in the literature according to their security mechanisms.
- Fine-grained analysis and comparison of 44 privacy-preserving reputation systems using the proposed analysis frameworks.
- Detailed review of several significant and representative privacy-preserving reputation systems proposed in the literature.
- Discussion of the analysis results, and based on these results, insights and directions for future work in this research area.

3.1.2 Organization

The rest of the chapter is organized as follows. Section 3.2 proposes an analysis framework that identifies the dimensions and the requirements of privacy-preserving reputation systems. Section 3.3 defines two broad categories of privacy-preserving reputation systems with respect to their security objectives. Section 3.4 develops an analysis framework encompassing the various non-privacy related dimensions of reputation systems. Section 3.5 defines broad categories of the privacy-preserving reputation systems proposed in the literature according to their security mechanisms. Section 3.6 presents a fine-grained analysis of privacy-preserving reputation systems proposed in the literature according to the frameworks established in Sections 3.2 through 3.4. Section 3.7 describes in greater detail some of the blockchain-based systems. Section 3.8 discusses the analysis results and relevant insights. Section 3.9 concludes the survey.

3.2 An Analysis Framework for Privacy-Preserving Reputation Systems

In this section, we introduce our analysis framework that identifies the common dimensions and requirements of privacy-preserving reputation systems. The dimensions of the analysis framework regarding security objectives are described in Section 3.3. We conduct a fine-grained analysis and comparison of privacy-preserving reputation systems proposed in the literature using this framework in Section 3.6.

Some fundamental concepts in reputation systems are as follows:

Source User (Rater). A user u is said to be a source user or rater of a user t if u has feedback about t in a given context.

Target User (Ratee). When a source user assigns feedback to a user t, or a user q initiates a query to determine the reputation of user t, the user t is referred to as the target user or the ratee.

Querying User (Querier, Inquirer). When a user q initiates a query to determine the reputation of a user t, the user q is referred to as the querying user, the querier, or the inquirer.

Reputation. The reputation of a target user is any function that aggregates the feedback of its source users. In Section 3.4, we list some possible realizations of the aggregation function.

Our analysis framework is graphically represented in Figure 3.1.

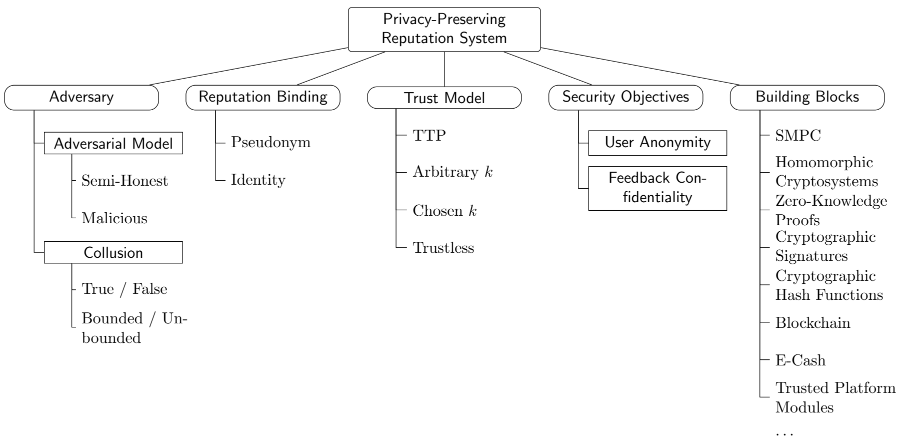


Figure 3.1: Analysis framework for privacy-preserving reputation systems.

3.2.1 Adversary

The goal of a reputation system is to compute the reputation from the inputs of the participants. All participants of the protocol are expected to pursue this and only this goal. An honest participant is one who conforms to this expectation. However, there may exist dishonest participants who have ulterior motives. Those motives may include learning the inputs of other participants, tampering with the output, disrupting the protocol, etc.

Adversarial Model

We list below two standard adversarial models [121] that characterize the behavior of dishonest users. A privacy-preserving reputation system is considered secure under one of these models if it can show correctness and meet its privacy requirements under the given model.

Semi-Honest. In the semi-honest model, the users do not deviate from the specified protocol. In other words, they always execute the protocol according to the specifications. However, the adversary passively attempts to learn the inputs of honest users by using intermediate information received during the protocol execution and any other information that it can gain through legitimate means.

Malicious. Malicious users are not bound to conform to the protocol. Users under a malicious model may deviate from the protocol as and when they deem necessary. They actively attempt to achieve their objectives. A malicious adversary may have either or both of the following objectives: 1) learn the inputs of honest users, and 2) disrupt the protocol for honest users. The reasons for disrupting the protocol may range from gaining illegitimate advantage over honest users to completely denying the service of the protocol to honest users.

Collusion

A dishonest user may act alone or multiple dishonest users may act in agreement to achieve their ulterior motives. The collaboration of multiple dishonest users is referred to as collusion. Privacy-preserving reputation systems either consider that collusion can take place between users or consider that collusion does not take place.

Collusion can be bounded or unbounded. Bounded collusion implies that the number of dishonest colluding participants is limited, for example, by $\frac{1}{2}$ or $\frac{1}{3}$ of all n participants. Unbounded collusion places no limit on the number of dishonest participants who can collude with each other, thus $n-1$ of the participants can be dishonest and collude, except for the one honest participant whose privacy needs to be preserved.

3.2.2 Reputation Binding

A privacy-preserving reputation system can be either pseudonym-bound or identity-bound.

In a pseudonym-bound system, the reputation of the ratee is associated with her pseudonym. If she changes or creates a new pseudonym, then she looses her reputation. The use of pseudonyms has the drawback that the reputation is not transferable between a ratee's multiple pseudonyms.

An additional major drawback is that a dishonest ratee can drop a pseudonym with bad reputation and re-enter the system with a new pseudonym and a fresh reputation. This common attack against pseudonym-bound systems is known as whitewashing.

On the other hand, in an identity-bound system, the reputation of a ratee is bound to her real identity. Even if she changes pseudonyms, she maintains her reputation. This is often made possible by verifying the true identity of a ratee before issuing a new pseudonym.

3.2.3 Trust Model

The security and privacy guarantees that users receive in a privacy-preserving reputation system often require that they trust certain entities, such as a central authority, or some fellow users in the system. The trust implies a belief by the trusting user that the trusted entity or the trusted fellow users will behave in an expected manner in order to ensure their security and privacy. In Section 2.2.1, we identified four different types of trust models that privacy-preserving reputation systems are based on: 1) Trusted Third Party (TTP), 2) Trust on arbitrary k fellow users, 3) Trust on chosen k fellow users, and 4) Trustless.

3.2.4 Building Blocks: Blockchain

In order to achieve their security objectives, privacy-preserving reputation systems utilize various building blocks, which are generally cryptographic in nature. These building blocks include secure multi-party computation, homomorphic cryptosystems, zero-knowledge proofs, blockchain, etc. In Section 2.1.4, we presented an overview of blockchains. The description of the more traditional cryptographic building blocks can be found in cryptography texts as well as in the extended version of this survey [135] released as a research report.

3.3 Security Objectives of Privacy-Preserving Reputation Systems

We have identified two broad categories of privacy-preserving reputation systems with respect to their security objectives. The goal of the systems in the first category is to preserve the anonymity of the users. The systems in the second category do not aim to hide the identity of the users but focus on preserving the confidentiality of the feedback that the users provide. The two categories of privacy-preserving reputation systems are defined as follows:

1. **User anonymity-oriented privacy-preserving reputation systems.** The true identity of the users is hidden in these systems. The feedback providers thus remain anonymous. A user is represented in the system by one or more pseudonyms which are unlinkable to her real identity. This setup allows the user to anonymously carry out transactions with others and submit feedback. The submitted feedback does not need to be confidential since the anonymity of the users prevents the feedback from being linked to them.

2. **Feedback confidentiality-oriented privacy-preserving reputation systems.** These systems do not attempt to hide the identity of the users beyond assigning each user a single pseudonym. Moreover, these systems do not conceal the act of a user assigning feedback to another user. However, the value of the submitted feedback and any other related information are considered private. This type of systems is necessary since complete anonymity is not always possible due to the nature of real world transactions. For example, even if anonymity is preserved online on an e-commerce site, the exchange of physical items sold and bought through the site would reveal the real identities of the participants. Preserving the confidentiality of the feedback is a practical alternative to enable users to submit truthful feedback without fear of retaliation.

The security objectives of a privacy-preserving reputation system can be further categorized as those fulfilling privacy and those fulfilling integrity or correctness. The privacy objectives are related to hiding information about users, for example, preserving the anonymity of the rater and the ratee. On the other hand, the integrity objectives aim at maintaining the correctness of the functions of the reputation system while preserving the privacy of the users. An example of integrity objectives is preventing a malicious user from manipulating the reputation aggregation function to forge an unmerited good reputation.

Figure 3.2 graphically represents the classification of the security objectives of privacy-preserving reputation systems. In Sections 3.3.1 and 3.3.2, we describe specific security objectives of user anonymity and feedback confidentiality-oriented privacy-preserving reputation systems, respectively. A given reputation system may pursue a few or more of these objectives depending on the stringency of its security requirements.

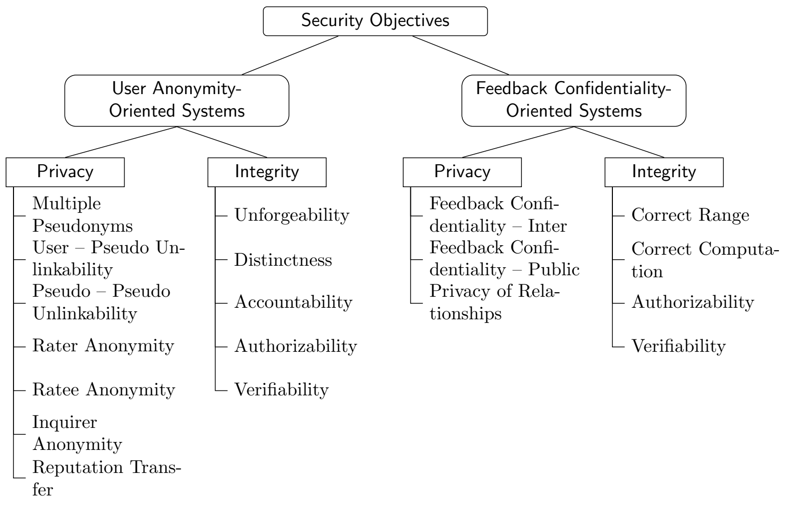


Figure 3.2: Security objectives of privacy-preserving reputation systems.

3.3.1 User Anonymity-Oriented Privacy-Preserving Reputation Systems

Privacy Objectives

Multiple Pseudonyms. A user is able to assume multiple pseudonyms in the system. As noted by Anwar and Greer [33, 34], the variation in the pseudonyms of a user may be on a per context or a per transaction basis. In the first case, a user may adopt a different pseudonym for each context in the system. For example, a tutor could use different pseudonyms for different subjects in an e-learning system. Alternatively, a user may choose a different pseudonym for each transaction in the system.

User-Pseudonym Unlinkability. User-pseudonym unlinkability implies that the true identity of a user is not linkable to any pseudonym that she uses in the system. Androulaki et al. [32] characterize this requirement as follows: Given a pseudonym P that does not belong to a corrupted party, the adversary can learn which peer owns P no better than guessing at random among all non-corrupted peers that appear consistent with P.

Pseudonym-Pseudonym Unlinkability. Pseudonym-pseudonym unlinkability implies that two different pseudonyms that belong to the same user cannot be linked to each other. The adversary is unable to tell whether two given pseudonyms belong to the same user. Androulaki et al. [32] specify this property as follows: Given two pseudonyms P_1, P_2 that do not belong to corrupted parties, the adversary has no advantage in telling whether P_1, P_2 belong to the same peer or not. This requirement should hold as long as there are at least two non-corrupted peers who appear consistent with both P_1 and P_2 (because if there is only one such uncorrupted peer, clearly both pseudonyms belong to the same one).

Rater Anonymity. A user is able to rate another user without her true identity being revealed. The purpose of rating anonymously is to prevent the adversary from linking the rater to her interaction with the ratee and the rating that she submitted. Schiffner et al. [257] specify this property as follows: A pseudonym P_1 that interacted with a ratee R should not be linkable to the pseudonym P_2 that rated R.

Ratee Anonymity. A user is able to receive a rating without her real identity being disclosed. A ratee may not wish to be associated with her past transactions and ratings since they could influence the ratings for her future transactions. According to Schiffner et al. [257], this property implies that a ratee R can use a different pseudonym for each transaction.

Inquirer Anonymity. A user is able to inquire about the reputation of another user. However, others are not able to learn whose reputation she is querying or even the fact that she is inquiring about another user's reputation. Users wish to query the reputation of other users anonymously in order to prevent the adversary from compiling a profile of their interactions and interests.

Reputation Transfer and Aggregation. A ratee is able to transfer reputation among multiple pseudonyms that she owns without letting the adversary infer associations between these pseudonyms. Consequently, a ratee is able to aggregate the reputation of her multiple pseudonyms into the reputation of one pseudonym.

Integrity Objectives

Reputation Unforgeability. A ratee is unable to show reputation higher than the cumulative reputation of her pseudonyms. A ratee is also unable to borrow good reputation from another ratee.

Distinctness. It is possible to prove that the reputation of a ratee is an aggregate of votes or feedback from distinct raters while simultaneously hiding the identities of those raters. The advantage of this property is that one or a few dishonest raters are not able to submit multiple votes or feedback (ballot stuffing) for artificially increasing or decreasing the reputation of the ratee.

Accountability. If and only if a user commits a predefined adversarial act, such as ballot stuffing, then her pseudonym becomes linkable to her real identity. This property ensures that anonymous users are still accountable for adversarial actions.

The properties of authorizability and verifiability are discussed in Section 3.3.3.

3.3.2 Feedback Confidentiality-Oriented Privacy-Preserving Reputation Systems

Privacy Objectives

No Inference from Intermediate Information. This property requires that a rating assigned by a rater to a ratee is never revealed to any other party including the ratee. The system must protect the confidentiality of the feedback such that the feedback is neither divulged explicitly nor inferred from any intermediate information gained by the adversary during a reputation query. The system may define the confidentiality of the feedback as deterministic or probabilistic. In the first case, the adversary is unable to learn any information about the feedback. However, in the latter case of probabilistic confidentiality, the amount of information leakage depends on certain variables, such as the number of raters, the reputation score, etc.

No Inference from Public Information. The reputation score of any ratee is by definition public and any other user in the system is authorized to learn this score. The issue is that a dishonest user may use this public information to derive the private feedback of honest raters. For example, in a basic additive reputation system, the adversary simply needs to observe the reputation score before and after the latest rater submits her feedback to learn its value. The requirement of confidentiality of feedback, with no inference from public information, implies that the adversary is unable to learn information about the feedback even from publicly available information.

Privacy of Relationships. A user may have relationships with multiple users in the system. These other users may include fellow users who have rated the same ratees. The relationships between the users could be in various contexts, for example, the context of trust in preserving each others privacy. This requirement implies that information about the relationships of a rater is not revealed during the course of a reputation query. This information includes the amount of trust that the rater has in the fellow users.

Integrity Objectives

No Out of Range Feedback. A dishonest rater is unable to submit out of range feedback. A dishonest rater may take advantage of the fact that the feedback is confidential and submit out of range feedback in order to mount an attack such as bad mouthing or ballot stuffing. A system enforcing this property does not permit out of range feedback even though the feedback is hidden.

No Incorrect Computations. A dishonest user is unable to carry out incorrect computations. A reputation query may require users to perform certain computations, for example, the summation of some values. This property requires that a dishonest user is unable to submit erroneous results for these computations.

3.3.3 Integrity Objectives Common to Both Types of Privacy-Preserving Reputation Systems

Authorizability of Ratings. The requirement of authorizability of ratings implies that only the users who have had a transaction with the ratee are allowed to rate her. This property prevents users who have not transacted with a ratee from assigning her feedback and thus possibly reduces the impact of attacks such as bad mouthing and self promotion.

Verifiability by Ratee. The requirement of verifiability by ratee, as identified by Kerschbaum [169], suggests that a ratee R should be able to identify all published feedback linked to her identity and verify that they are related to a recorded transaction and the correct transaction partners. Moreover, a ratee R should be able to identify all published feedback linked to her identity and verify that the inquirer has computed its reputation score according to them.

3.4 An Analysis Framework for Reputation Systems

In this section, we develop an analysis framework that identifies the various non-privacy related dimensions of reputation systems. Since privacy-preserving reputation systems are fundamentally reputation systems, we need to establish a uniform framework to analyze and compare their non-privacy features as well. However, we do not describe these dimensions in detail in this chapter since they have been covered extensively by prior works (such as the surveys by Braga et al. [60], Hendrikx et al. [144] and Hoffman et al. [146]). Additionally, the details of these properties can be found in the extended version of this survey [135] released as a research report.

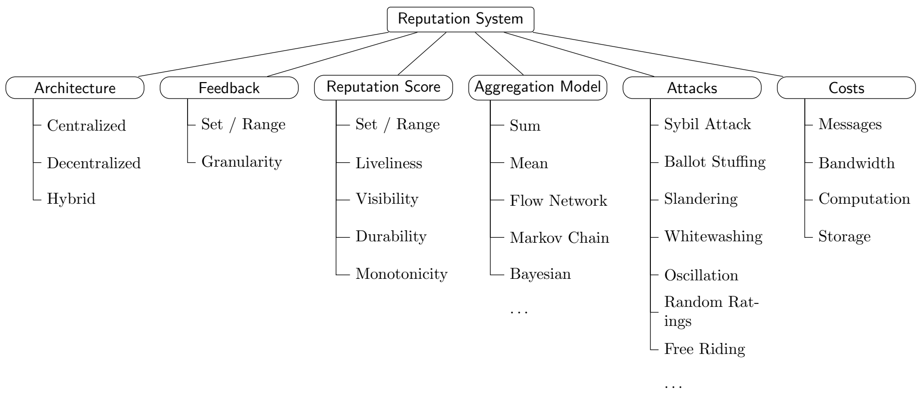


Figure 3.3: Analysis framework for reputation systems.

The architecture of a reputation system is one of the key factors in determining how the following activities are conducted: 1) Feedback collection; 2) Feedback aggregation (reputation computation); and 3) Reputation dissemination. The architecture of a reputation system can be centralized, decentralized, or hybrid.

The properties of feedback include the *set or range* that the feedback belongs to, for example, $\{-1, 0, 1\}$, $[0, 1]$. Additionally, the *granularity* of the feedback of a rater reflects either the experience with the ratee for a single given transaction or the cumulative experience over multiple transactions.

The properties of reputation include the set or range of its values, for example, $\mathbb{R}$, $[0, 1]$. Some other properties of reputation are liveliness, visibility, durability, and monotonicity. As noted by Schiffner et al. [257], reputation *liveliness* implies that a reputation system does not offer users the possibility to reach a final state of reputation in which bad behavior no longer damages their reputation. For example, for a reputation score in the set $\mathbb{R}$, there are no minimum and maximum limits, whereas, for a reputation score in the interval $[0, 1]$, the reputation can reach the mininum value of 0 or the maximum value of 1. The *visibility* of a reputation score may be global or local. Global visibility implies that all nodes in the system view the same reputation score of a certain entity. Whereas with local visibility, the reputation score available to a subset of the nodes may be different than elsewhere in the system. Reputation *durability* refers to the transience of a reputation score. Once a reputation score is computed, it may be stored permanently until the reputation changes or may remain transient and require re-computation for every query. *Monotonic* reputation implies that the reputation score increments in only one direction. For example, consider a reputation system in which a ratee can receive integer feedback between 1 and 5 for each transaction, and reputation is considered as the sum of feedback. The reputation in such a system cannot be decremented.

There are a number of models for aggregating feedback to obtain reputation scores. Some common models include sum, mean, flow network, Markov chain, and Bayesian ones. A comprehensive survey of feedback aggregation models (also referred to as reputation computation engines) is provided by Jøsang et al. [160].

Reputation systems can be classified by the attacks that they address and their success in defending against them. Some of the attacks that reputation systems have to contend with include Sybil attack (a single user owning and exploiting multiple identities for malicious purposes), self-promotion or ballot stuffing (improving a ratee's reputation by providing false positive feedback), slandering or bad-mouthing (damaging a ratee's reputation by providing false negative feedback), whitewashing (leaving the system and then re-entering with a fresh reputation), oscillation (cultivating good reputation with the intention to exploit it for malicious purposes), random ratings (submitting randomly generated feedback in order to demonstrate active participation), and free riding (benefiting from the reputation system without providing any contribution). Surveys by Hoffman et al. [146] and Mármol and Pérez [204] describe some of these attacks in detail.

The operations of a reputation system, which include feedback collection, feedback aggregation (reputation computation), and reputation dissemination, incur various computational costs. The costs of these operations can be measured as follows: 1) number of messages exchanged; 2) bandwidth consumed; 3) computational resources consumed; and 4) storage required.

3.5 Categorization of Privacy-Preserving Reputation Systems according to their Security Mechanisms

In this section, we identify broad categories of the privacy-preserving reputation systems proposed in the literature. These categories are based on the general mechanisms that these systems rely on in order to guarantee privacy and other critical security properties, for example, authorizability, verifiability, etc.

We also briefly discuss the contributions of the systems that belong to each of these categories. Each system is further analyzed in depth and compared in Section 3.6. Five of the listed blockchain-based systems are discussed in detail in Section 3.7.

Note that these categories are not mutually exclusive and a system may belong to multiple of these categories. For example, the system by Schiedermeier et al. [256] can belong to the category of blockchain-based systems as well as SMPC-based systems. However, we place a system under a single category based on its main novel idea. For example, even though Schiedermeier et al.'s work uses SMPC, the novel idea and the main contribution is rather the use of a blockchain-based public ledger as the sole communication medium between the parties of the SMPC protocol. The blockchain-based protocol provides transparency and verifiability properties that are usually missing from SMPC-only systems. The system by Schiedermeier et al. is therefore categorized as a blockchain-based system.

Article Selection Methodology: In this survey, we have included the systems that we are aware of in this area of research as well as those discovered using the following approach. We searched for articles on Google Scholar published during the period of 2000 to July 2021. The search phrases included the keyword 'reputation' along with one of the keywords 'privacy', 'anonymous', and 'anonymity'. For each relevant article found, we studied its list of references to find other potential systems. Moreover, we also looked at the article's "Cited by ..." list on Google Scholar to discover later relevant papers that cite the given article. All articles that present privacy-preserving reputation systems that we discovered have been included in this survey. We have excluded some articles that present systems similar to those that have been included, for example, articles by the same authors that describe predecessors of their subsequent systems. We have also excluded short papers (4 pages or less) that do not describe the proposed systems in sufficient detail.

3.5.1 Blockchain-based Systems

These systems rely on a blockchain or smart contracts as an integral building block for achieving their security objectives. This is one of two categories (the other one being SMPC-based systems, described in the next subsection) that constitute mainly of decentralized systems. Moreover, this is the only category that comprises of systems that can guarantee trustlessness.

Schaub et al. [255] introduced the first blockchain-based trustless privacy-preserving reputation system. The system does not need to rely on trusted third parties, arbitrary trusted nodes, or subjective trust relationships in order to guarantee security. Using blinded tokens issued by service providers, raters anonymously submit feedback, which is recorded on a public immutable blockchain. Issuing a token requires spending the system's cryptocurrency, which provides an incentive to mine and maintain the blockchain and also discourages ballot-stuffing. Bazin et al. [45] present a system, which in addition to protecting rater privacy, enables retrieval of a self-reported reputation score directly from the target service provider. The validity of the reputation score is verifiable and only a constant number of messages need to be exchanged for its retrieval.

Azad et al. [39, 40] propose privacy-preserving reputation systems for online marketplaces and for the Social Internet of Things environment. Self-enforcing computation is a property of their latter system, which implies that the computation process is independent of any trusted third party and it allows verification of the integrity of the scores in an autonomous and public manner. Bag et al. [42] describe a system for computing personalized global reputation of a target, which considers only the feedback from a set of trusted participants. This is done without disclosing the identities of the members of the trusted set and their feedback. The systems by Azad et al. and Bag et al. rely on a public bulletin board for communication, which according to the authors may be realized by a blockchain.

Dou et al. [103] propose a distributed trust evaluation protocol with privacy protection for the Intercloud environment. A distinctive feature of the protocol is that it can continue to function even if some of the feedback providers go offline. Kang et al. [163] devise a blockchain-based scheme for secure data sharing among vehicles in Vehicular Edge COmputing and Networks (VECONs). A reputation system based on a three-weight subjective logic model is employed to manage the trustworthiness of vehicles in terms of the quality of data shared. The anonymity of the vehicles is maintained by allowing multiple pseudonyms. Lu et al. [198] present a privacy-preserving trust model based on blockchain for vehicular adhoc networks. Vehicles can anonymously submit alerts about traffic conditions and neighboring vehicles can provide feedback about the validity of the alerts. The anonymous reputation of a vehicle reflects the feedback received regarding its contributions. Owiyo et al. [229] propose a decentralized privacy-preserving reputation system based on blockchain that is claimed to provide low transaction overheads.

Jo and Choi [159] describe a blockchain-based privacy-preserving reputation framework for participatory sensing systems. The system includes a smart contract that manages the reputation of participants based on their sensing data and the corresponding feedback. The smart contract and the underlying blockchain enable transparency and public auditability of the reputation scores. Liu et al. [192] present an anonymous reputation system for retail marketing in the Industrial Internet of Things environment. The system, which also uses smart contracts on a Proof of Stake blockchain as a building block, is able to provide transparency and public verifiability under the malicious adversarial model. Schiedermeier et al. [256] describe a protocol for holding referendums in trustless networks, which can also serve as a reputation protocol. The protocol combines SMPC with a blockchain as the unique channel for communication between the parties. The protocol ensures transparency, that is, maintaining a public trace of all operations performed and the information exchanged among the participants. Moreover, any participant is able to autonomously verify the correctness of the outcome of the referendum.

Zhao et al. [296] propose a privacy-preserving reputation system that takes advantage of blockchain technology in the resource-constrained environment of mobile crowdsensing. The global reputation scores are updated by a smart contract based on the average of all feedback. The system overcomes the challenge of user dynamics, that is, frequent user turnover, by including a delegation protocol. Zhang et al. [293] present another privacy-preserving reputation management scheme for mobile crowdsensing that is based on blockchain. The well-known Eigentrust distributed reputation computing algorithm is adapted in this system such that participant privacy is preserved.

Dimitriou [96] develops a blockchain-based fully decentralized privacy-preserving reputation system. The participants can change pseudonyms as frequently as they wish, yet they can maintain user-pseudonym and pseudonym-pseudonym unlinkability, while being able to aggregate reputation among those pseudonyms. The system provides fully trustless operations, except for the user registration operation that relies on the trustworthiness of an entity called the Registrar, which may be composed of a single server or a decentralized set of nodes. However, the Registrar is trusted only for ensuring uniqueness of user identities and for reputation soundness.

3.5.2 SMPC-based Systems

These systems use feedback score as direct evidence from witnesses to compute a reputation score. Their goal is to obfuscate the feedback score of the witnesses from the querier as well as from fellow witnesses. These systems use Secure Multi-Party Computation to achieve their goal. The reputation systems in this category focus primarily on feedback confidentiality as their security objective. Decentralization is one of the key advantages of SMPC-based systems over systems in the categories discussed next that are mostly centralized.

Pavlov et al. [235] introduced SMPC-based privacy-preserving reputation systems by proposing a number of protocols for decentralized additive reputation systems. Two of their protocols are secure under the semi-honest and the malicious adversarial models, respectively. The protocols draw their strength from witness selection schemes, which guarantee the inclusion of a certain number of honest witnesses as participants. Gudes et al. [128] and Gal-Oz et al. [114] present several schemes that augment their Knots reputation system [115] with privacy-preserving features. A defining characteristic of the Knots reputation model is the notion of subjective reputation. The reputation of a target member is computed by each querying member using a different set of feedback, thus the reputation is subjective for each querying member. Nithyanand and Raman's system [225] complements an SMPC mechanism for privacy with a fuzzy technique and an Ordered Weighted Average (OWA) operator in order to compute local as well as global reputation scores.

Hasan et al. [139] present a system that operates under the more demanding malicious adversarial model and offers the chosen k trust model (discussed in Section 3.2.3) instead of the usual arbitrary k trust model for privacy preservation. Dimitriou and Michalas [97, 98] describe a decentralized privacy-preserving scheme that is formally shown to be resistant to collusion against up to $n-1$ malicious participants. Dolev et al. [101, 102] propose SMPC-based reputation schemes that are more efficient than the previous ones in terms of the number of messages exchanged. Their schemes privately compute reputation scores with a communication overhead of $O(n)$ messages, where n is the number of participants in the protocol. Clark et al. [83] present a dynamic privacy-preserving decentralized reputation system. They specifically address the problem of the dynamicity of the nodes in a network. Nodes may frequently leave along with their feedback, which then becomes unavailable for reputation computation in a decentralized manner. Clark et al. propose a privacy-preserving reputation information delegation protocol to counter this problem. Bakas et al. [43] propose an SMPC-based privacy-preserving decentralized additive reputation system, which is the first one to practically utilize Functional Encryption (FE), an emerging cryptographic building block that permits selective computations on encrypted data.

3.5.3 Token-based Systems

These systems are a type of privacy-preserving reputation systems in which a cryptographic token is issued to a pseudonymous user participating in a transaction. The token is implemented using a blind signature or another scheme. The token is issued either by a central entity (called the bank in the system by Androulaki et al. [32]) or directly by the ratee to the rater (as in the system by Kerschbaum [169]). A variation of the following approach is then employed in order to credit the ratee with a reputation point while preserving the privacy of the token depositing user. The token is deposited by the user to an account maintained by the central entity using a different pseudonym or even their real identity. The blinded nature of the token unlinks the user from the initial pseudonym while assuring the central entity of the legitimacy of the deposit. The number and the value of the tokens deposited reflect the reputation of the ratee. An advantage of user anonymity-oriented token-based systems over SMPC-based systems is the ability of users to assume multiple pseudonyms.

The system by Androulaki et al. [32] addresses the difficulties outlined by Dingledine et al. [99] for building reputation systems in anonymous user networks. Androulaki et al.'s system achieves: 1) unlinkability between a pseudonym and the identity of its user; 2) no double-awarding or forging of a token; 3) no false accusations of forgery; and 4) non-transferability of reputation, that is, a user cannot borrow reputation from another user. The system by Kerschbaum [169] builds on the blinded token idea to achieve feedback confidentiality while enforcing the property of verifiability. Schiffner et al. [257, 258] improve upon Androulaki et al.'s work by introducing systems that support the properties of liveliness and non-monotonicity.

Zhang et al. [292] propose a reputation system that preserves the privacy of feedback providers and resists Sybil attacks. The system is based on the Camenisch and Lysyanskaya (CL) signature scheme. Busom et al. [66] describe a privacy-preserving reputation system based on Chaum-Pedersen blind signatures that allows users to anonymously submit text feedback about a target entity. Fellow users can in turn anonymously endorse a text feedback that they find helpful. The system thus encourages honest feedback. Moreover, the system offers a privileged status for users who earn sufficient endorsements thus also incentivizing feedback submission.

3.5.4 Proxy-based Systems

These systems aim to maintain privacy through the use of a trusted third party as a proxy between the raters and the reputation querier. The proxy may forward the anonymized feedback scores to the querier or the proxy may compute the aggregated reputation and only report that to the querier. Additionally, the querier and the raters may interact directly. However, in this case, a rater is generally issued an anonymous identity or an encryption key by the proxy to protect their privacy. The proxy may be composed of one or several central entities. Usually, the architecture of these systems comprises of one to three central entities that are considered not to collude with each other in order to guarantee security. The proxy may be considered partially or fully trusted.

Ries et al. [250] propose an approach for privacy-preserving computation of trust. A key contribution of this approach is that in addition to computing reputation based on encrypted private feedback, the querier can also evaluate the trustworthiness of the raters. Petrlic et al. [237] propose a reputation management system that focuses on privacy (anonymity in reputation retrieval, and anonymity in rating) as well as robustness (authorization, authentication, integrity, and accuracy). A semi-honest Reputation Provider (RP) entity serves as an intermediary between the raters and the service providers. The RP manages the reputation of the service providers and helps enforce some of the above listed security objectives.

Mousa et al. [218] present PrivaSense, a privacy-preserving reputation system for mobile participatory sensing applications. The system implements a sequence of registration and authentication phases orchestrated by independent central servers that ensure participants' anonymity and improve the system's resilience against Sybil and replay attacks. Ma et al. [200] propose a privacy-preserving reputation management system for edge computing enhanced mobile crowdsensing. The architecture comprises of a Central Manager (CM), a Reputation Manager (RM), and a Central Authority (CA). Participants submit sensing data in homomorphic encrypted form. The encrypted deviation of a participant's data from the aggregated result is computed and the RM updates reputation according to the deviation.

3.5.5 Signature-based Systems

Inspired by cryptographic digital signatures and group signature schemes, Benthencourt et al. [49] propose a new cryptographic framework called signatures of reputation. In a scheme based on this framework, the verification of the signature of a user reveals her reputation instead of revealing her identity. This is in contrast to a conventional signature scheme where the verification of the signature of a user results in the confirmation of the identity of the user associated with the corresponding public key.

Guo et al. [129] build upon the notion of signatures of reputation to propose a fine-grained attribute-based privacy-preserving reputation system. The system enables users to rate each other's attributes instead of real identities. The signature verification process provides authenticity of the reputation value of a user for a given attribute. Bethencourt et al.'s system is improved by the work of Anceaume et al. [31] and Lajoie-Mazenc et al. [178], who implement non-monotonic signature-based reputation systems. Whereas, Bethencourt et al.'s system only supports monotonic reputation.

Chen et al. [76] present a privacy and reputation-aware announcement scheme for vehicular adhoc networks where vehicles can report road conditions. The scheme is based on the Boneh-Boyen-Shacham (BBS) short group signatures. The scheme overcomes the problem of having to establish a secure channel for reputation score retrieval in prior systems.

3.5.6 Transitory Pseudonym-based Systems

Transitory pseudonym-based systems aim to obfuscate a user's identity by assigning them multiple short-term pseudonyms. The focus is on how to make the multiple pseudonyms of a user unlinkable with the user as well as with one another. Moreover, how to transfer reputation from one pseudonym to another while preventing observation and profiling is also addressed.

One of the first systems in this category is RuP (Reputation using Pseudonyms) by Miranda and Rodrigues [216]. In their system, a user is identified by a certified pseudonym that is valid only for a predefined time slot. The certified pseudonyms are issued by a TTP called Pseudonym Certification Authority (PCA). However, the link between the real identity of the user and the pseudonym is hidden from the PCA as well. The system also includes a scheme based on blind signatures that allows a user to transfer their reputation associated with an old pseudonym to a new one, without disclosing the link between them or their real identity. Another early work in this category is by Steinbrecher [273]. Their system enables simultaneous use of multiple pseudonyms by a user and permits them to regularly change their pseudonyms to achieve anonymity. To prevent an adversary from linking new and old pseudonyms, the system suggests using a set of non-colluding trustworthy third parties who make incremental changes to the pseudonym of the user.

Anceaume et al. [30] propose a privacy-preserving distributed reputation mechanism. The system allows users to themselves generate pseudonyms in order to achieve anonymity. They introduce the concept of mailboxes, which are agents that replicate anonymous feedback, in order to provide resistance against network dynamicity and user misbehavior. Christin et al. [81] present IncogniSense, another improvement on the RuP scheme, which is claimed to achieve better protection against reputation manipulation and reduce the cryptographic overhead for the client.

3.5.7 Other Systems

In this category, we include systems that propose unique approaches and therefore cannot be placed in the above defined categories.

Kinateder and Pearson [173] introduced one of the earliest privacy-oriented decentralized reputation systems. The system requires a Trusted Platform Module (TPM) chip at each agent, which enables an agent to demonstrate that it is a valid agent and a legitimate member of the system without disclosing its true identity. This permits the agent to provide feedback anonymously. Bo et al. [56] present a privacy-preserving reputation system, which offers incentives to users for feedback submission. A user who anonymously submits feedback can also anonymously receive a discount token (an incentive) from the ratee. The architecture of the system comprises of a Card Issuer (CI) entity and a Registration Center (RC) entity that are responsible for issuing smart cards and anonymous identities to users, respectively.

3.6 Fine-Grained Analysis and Comparison of Privacy-Preserving Reputation Systems

In this section, we conduct fine-grained analysis of privacy-preserving reputation systems in the literature according to the frameworks established in Sections 3.2 through 3.4. The analysis is presented in the form of Tables 3.1 through 3.6. The tables also permit side by side comparison of the systems.

We have analyzed 44 privacy-preserving reputation systems in depth and summarized their properties in the given tables. We report information about the systems as gleaned from the articles. In case of multiple variants of a system presented in the same article, we have selected the variant that provides the strongest security guarantees. The systems are grouped in the tables according to the category of their security mechanisms. The categories are ordered by the number of included systems and then alphabetically. Under each category, the systems are ordered chronologically to allow observation of the evolution of the systems.

Table 3.1 identifies the fundamental characteristics of each reputation system according to the analysis framework developed in Section 3.4. The architecture of the systems and the properties of their feedback and reputation are presented.

Table 3.2 and Table 3.3 present the security related fundamentals of user anonymity and feedback confidentiality-oriented systems, respectively. In accordance with the analysis framework for privacy-preserving reputation systems formulated in Section 3.2, the properties reported include the adversarial model, the extent of collusion resistance, reputation binding, the trust model, and the main security building blocks. Multiple adversarial models are listed if a scheme uses different adversarial models for different entities, for example, semi-honest for the server, and malicious for the users. We note strong collusion resistance if t out of the n users in the protocol must collude to breach security, where $t < n$, and t is variable. For example, $t = \frac{1}{2}n$, or $t = \frac{1}{3}n$. Alternatively, we note partial collusion resistance if a constant number of colluding entities, for example, two partially trusted colluding servers, are able to breach security. Multiple trust models are noted for the systems that rely on different models for their different security properties. The aggregation model is stated as open where the system is not constrained to one specific function.

The details of the security objectives of user anonymity and feedback confidentiality-oriented systems are presented in Table 3.4 and Table 3.5, respectively. As discussed in Section 3.3, the security objectives of privacy-preserving reputation systems include those aiming to enforce privacy and those targeting integrity or correctness.

The robustness of the reputation systems against common attacks listed in Section 3.4 is summarized in Table 3.6.

Table 3.1: Fundamentals.

		Feedback		**Reputation**					
System	**Architecture**	**Set / Range**	**Granularity**	**Set / Range**	**Liveliness**	**Visibility**	**Durability**	**Non-Monotonicity**	**Aggregation Model**
Blockchain-based Systems									
Schaub et al. 2016	D	$\mathbb{Z}$	S	$\mathbb{R}$	●	G	○	●	Open
Bazin et al. 2017	D	$\mathbb{Z}$	S	$\mathbb{R}$	●	G	●	●	Open
Azad et al. 2018	D	$\{-,+\}$	S	$\mathbb{Z}$	●	G	○	●	Beta reputation
Bag et al. 2018	D	$\{0,1\}$	M	$[1,10]$	○	L	○	●	Mean
Dou et al. 2018	D		S			G	●		Weighted mean
Kang et al. 2018	H	$[0,1]$, multi-criteria	M	$\mathbb{R}$	●	G	●	●	Subjective logic
Lu et al. 2018	C	$\{-1,0,1\}$, $[0,1]$	S	$\mathbb{R}$, $[0,1]$	●	G	●	●	Polynomial
Owiyo et al. 2018	D		S			G	●		Open
Jo and Choi 2019	H	$\{-1,1\}$	S	$\mathbb{R}$	○	G	○	●	Sum
Liu et al. 2019	C	$[1,10]$	S	$\mathbb{N}$	●	G	●	○	Sum
Schiedermeier et al. 2019	D	$\{-1,1\}$	S	$\mathbb{Z}$	●	G	●	●	Sum
Zhao et al. 2019	C	$[0,1]$	S	$[0,1]$	●	G	●	●	Mean
Azad et al. 2020	D	$\{-1,1\}$	S	$\mathbb{Z}$	●	G	●	●	Weighted sum
Zhang et al. 2020	D	$[0,1]$	M	$\mathbb{R}$	●	G	●	●	Weighted sum
Dimitriou 2021	D		M	$\mathbb{Z}$	●	G	●	●	Sum
SMPC-based Systems									
Pavlov et al. 2004	D	$\mathbb{R}$	M	$\mathbb{R}$, $[0,1]$	●	L	○	●	Sum, beta reputation
Gudes et al. 2009	D	$\mathbb{R}$	M	$\mathbb{R}$	●	L	○	●	Weighted sum, mean
Nithyanand and Raman 2009	D	$\mathbb{R}, \{0,1\}$	M	$\mathbb{R}$	●	L	○	●	Ordered weighted average
Gal-Oz et al. 2010	D	$\mathbb{R}$	M	$\mathbb{R}$	●	L	○	●	Weighted sum, mean
Hasan et al. 2013	D	$[0,1]$	M	$\mathbb{R}, [0,1]$	●	G	○	●	Sum, mean
Dimitriou and Michalas 2014	D	$\mathbb{Z}$	M	$\mathbb{Z}$	●	G	○	○	Sum
Dolev et al. 2014	D	$\{1,2,\ldots,10\}$	M	$\mathbb{R}$	●	L	○	●	Weighted mean
Clark et al. 2016	D	$[0,v_{max}]$	M	$[0,v_{max}]$	●	L	○	●	Mean
Bakas et al. 2021	D	$\{n^1,n^2,\ldots,n^k\}$	M	$\mathbb{Z}$	●	L	○	●	Sum
Token-based Systems									
Androulaki et al. 2008	C	$\{0,1\}$	S	$\mathbb{Z}$	○	G	●	○	Sum
Kerschbaum 2009	C	$\{0,1\}$	S	$[0,1]$	●	G	●	●	Beta reputation
Schiffner et al. 2009	C	$\{-1,1\}$	S	$\mathbb{Z}$	●	G	●	●	Sum
Schiffner et al. 2011	C	$\{-,+\}$	S	$\mathbb{R}$	●	G	●	●	Open
Zhang et al. 2014	H		S	$\mathbb{R}$	●	G	●	●	Open
Busom et al. 2017	C	Text	S		●	G	●	●	Union
Proxy-based Systems									
Ries et al. 2011	C	$\{0,1\}$	M	$[0,1]$	●	L	○	●	Beta reputation
Petrlic et al. 2014	C	Vector, $\{0,1\}$	S	$\mathbb{Z}$	●	G	●		Sum
Mousa et al. 2017	C	$\{-1,0,1\}$, $[0,1]$	S	$[0,1]$	●	G	●	●	Bounded sum
Ma et al. 2018	C	$[0,1]$	M	$[0,1]$	●	G	●	●	Weighted mean
Signature-based Systems									
Bethencourt et al. 2010	H	$\{0,1\}$	S	$\mathbb{Z}$	●	G	●	○	Sum
Guo et al. 2013	C	$\{-1,1\}$	S	$\mathbb{Z}$	●	G	●	●	Sum
Lajoie-Mazenc et al. 2015	H	$\{-,+\}$, $\mathbb{Z}$	S	$\mathbb{R}$	●	G	●	●	Open
Chen et al. 2016	C		S	$\{0,1,\ldots,m\}$		G	●		Time discount function
Transitory Pseudonym-based Systems									
Miranda and Rodrigues 2006	C		S		●	G	●	●	Open
Steinbrecher 2006	C		S		●	G	●	●	Open
Anceaume et al. 2013	D	$[0,1]$	S	$[0,1]$	●	G		●	Beta reputation
Christin et al. 2013	C		S		●	G	●	●	Open
Other Systems									
Kinateder and Pearson 2003	D	$[0,1]$	S	$\mathbb{R}$	●	L	○	●	Open
Bo et al. 2007	H		S		●	G	●	●	Open

Legend

C – D – H	Centralized – Decentralized – Hybrid	●	Property satisfied
S – M	Single – Multiple	○	Property not satisfied
G – L	Global – Local		Property not specified or not applicable

Table 3.2: User Anonymity-Oriented Systems – Security Fundamentals and Building Blocks.

System	**Adversarial Model**	**Collusion Resistance**	**Reputation Binding**	**Trust Model**	**Building Blocks**
Blockchain-based Systems					
Schaub et al. 2016	M	●	P	Trustless	Okamoto / Chaum blind signatures, PoS blockchain
Bazin et al. 2017	M	●	P	A-*k*, TTP	Merkle trees, blind signatures, non-interactive zero-knowledge proofs, blockchain
Dou et al. 2018	SH, M	●	P	A-*k*, TTP	Additive homomorphic encryption, verifiable secret sharing, blockchain for feedback storage
Kang et al. 2018	SH, M	◒	I	A-*k*, TTP	Elliptic curve digital signatures, blockchain, smart contracts
Lu et al. 2018	SH, M	◒	I	TTP	Merkle trees, digital certificates, blockchain
Owiyo et al. 2018	SH		P		SMPC, blind signatures, blockchain
Jo and Choi 2019	SH, M	◒	I	TTP	Group signatures, blind signatures, blockchain, smart contracts
Liu et al. 2019	M	●	I	A-*k*, TTP	PS signature, bulletproof system, non-interactive zero-knowledge proofs, PoS blockchain, smart contracts
Dimitriou 2021	SH, M	●	I	Trustless, TTP	Pedersen commitments, blockchain, zkSNARK proofs
Token-based Systems					
Androulaki et al. 2008	SH, M	●	I	A-*k*, TTP	E-cash, anonymous credential system, blind signatures
Schiffner et al. 2009	SH, M	●	I	A-*k*, TTP	E-cash, cryptographic signatures, one-show credentials
Schiffner et al. 2011	SH, M	●	I	A-*k*, TTP	Symmetric key encryption, homomorphic encryption, DC-Net, Diffie-Hellman key exchange
Zhang et al. 2014	SH, M	●	I	TTP	Bilinear maps, Camenisch and Lysyanskaya (CL) signatures, Pedersen commitment, non-interactive zero-knowledge proofs
Busom et al. 2017	SH, M	●	I	TTP	Chaum-Pedersen zero-knowledge proofs, Chaum-Pedersen blind signatures, verifiable secret sharing, oblivious transfer
Proxy-based Systems					
Petrlic et al. 2014	SH, M	◒	I	TTP	Paillier additive homomorphic encryption, zero-knowledge proofs
Mousa et al. 2017	SH, M	◒	I	TTP	Digital certificates
Signature-based Systems					
Bethencourt et al. 2010	SH, M	◒	I	TTP	Homomorphic encryption, selective-tag weakly CCA-secure encryption, zero-knowledge proofs, one-time signatures
Guo et al. 2013	SH, M	◒	I	TTP	Boneh-Boyen signature scheme, homomorphic encryption, selective-tag encryption, Groth-Sahai non-interactive proofs
Lajoie-Mazenc et al. 2015	SH, M	●	I	A-*k*, TTP	Verifiable secret sharing, non-interactive zero-knowledge proofs, anonymous proxy signatures, SXDH commitments
Chen et al. 2016	SH, M	◒	P	TTP	Boneh-Boyen-Shacham (BBS) short group signature scheme
Transitory Pseudonym-based Systems					
Miranda and Rodrigues 2006	SH, M	◒	I	TTP	Cryptographic signatures, blind signatures
Steinbrecher 2006	SH, M	◒	I	TTP	Identity management, cryptographic credentials, cryptographic signatures
Anceaume et al. 2013	M	●	I	A-*k*, TTP	Overlay network, Distributed Hash Tables (DHTs), cryptographic commitments
Christin et al. 2013	SH, M	◒	I	TTP	Cryptographic signatures, blind signatures
Other Systems					
Kinateder and Pearson 2003	SH, M	◒	I	TTP	Trusted Platform Module (TPM), cryptographic signatures
Bo et al. 2007	SH, M	◒	I	TTP	Smart cards, cryptographic signatures, hash chain, zero-knowledge proof of possession

Legend

SH – M	Semi-Honest – Malicious
I – P	Identity – Pseudonym
A-*k* – C-*k* – TTP	Arbitrary *k* – Chosen *k* – Trusted Third Party
●	Strong resistance to collusion
◒	Partial resistance to collusion
○	Weak or no resistance to collusion
	Collusion resistance not specified or not applicable

Table 3.3: Feedback Confidentiality-Oriented Systems – Security Fundamentals and Building Blocks.

System	Adversarial Model	Collusion Resistance	Reputation Binding	Trust Model	Building Blocks
Blockchain-based Systems					
Azad et al. 2018	SH, M	●	P	A-k	Homomorphic encryption, non-interactive zero-knowledge proofs, public bulletin board (may be implemented by a blockchain)
Bag et al. 2018	M	●	P	A-k	SMPC, homomorphic encryption, zero-knowledge proofs, Schnorr signature protocol, public bulletin board (may be implemented by a blockchain)
Schiedermeier et al. 2019	M	●	P	A-k	SMPC, secret sharing, homomorphic encryption, blockchain
Zhao et al. 2019	SH, M	●	P	TTP	SMPC, additive secret sharing, blockchain, smart contracts
Azad et al. 2020	M	●	P	A-k	SMPC, homomorphic encryption, zero-knowledge proofs, public bulletin board (may be implemented by a blockchain)
Zhang et al. 2020	M	●	P	A-k, TTP	SMPC, Eigentrust algorithm, blockchain, smart contracts, verifiable secret sharing
SMPC-based Systems					
Pavlov et al. 2004	M	●	P	A-k	SMPC, Pederson verifiable secret sharing scheme, discrete-log commitment, zero-knowledge proofs
Gudes et al. 2009	SH	◒	P	A-k	SMPC
Nithyanand and Raman 2009	SH	◒	P	A-k	SMPC, Paillier additive homomorphic encryption
Gal-Oz et al. 2010	SH	●	P	A-k	SMPC, semantically-secure public-key encryption, homomorphic encryption
Hasan et al. 2013	M	●	P	C-k	SMPC, Paillier additive homomorphic encryption, non-interactive zero-knowledge proofs
Dimitriou and Michalas 2014	M	●	P	A-k	SMPC, Paillier additive homomorphic encryption, non-interactive zero-knowledge proofs
Dolev et al. 2014	M	●	P	A-k	SMPC, Paillier additive homomorphic encryption, Polhig-Hellman commutative encryption, ElGamal encryption
Clark et al. 2016	SH	●	P	C-k	SMPC, secret sharing, digital signatures
Bakas et al. 2021	SH	●	P	A-k	SMPC, Multi-Input Functional Encryption (MIFE), Trusted Execution Environment (TEE)
Token-based Systems					
Kerschbaum 2009	SH, M	◒	I	A-k, TTP	Homomorphic encryption, cryptographic pairings, zero-knowledge proofs
Proxy-based Systems					
Ries et al. 2011	SH, M	◒	P	TTP	Homomorphic encryption, zero-knowledge proofs
Ma et al. 2018	SH	◒	P	TTP	Somewhat-homomorphic encryption, cloud

Legend

SH – M	Semi-Honest – Malicious
I – P	Identity – Pseudonym
A-k – C-k – TTP	Arbitrary k – Chosen k – Trusted Third Party
●	Strong resistance to collusion
◒	Partial resistance to collusion
○	Weak or no resistance to collusion
	Collusion resistance not specified or not applicable

3.7 Blockchain-based Privacy-Preserving Reputation Systems

In this section, we describe in greater detail the system by Schaub et al. [255], which is the first blockchain-based trustless privacy-preserving reputation system. This system was developed in the context of the final year projects of Master's students A. Schaub and R. Bazin. Other significant blockchain-based privacy-preserving reputation systems are discussed in the full article [136] that this chapter is based on.

3.7.1 A Trustless Privacy-Preserving Reputation System. A. Schaub, R. Bazin, O. Hasan, and L. Brunie. 2016

Schaub et al. [255] design a reputation system for real-world e-commerce applications. It is therefore assumed that a customer c's real identity will be disclosed to the service provider SP during a transaction. Instead of complete anonymity, the system emphasizes user anonymity specifically for the feedback submission stage. The system requires unlinkability of the rater to the rating, unlinkability of the rating to the transaction, and unlinkability of the rating to other ratings by the same rater. These properties ensure that c can submit a rating without identification by the SP, and thus achieve user anonymity for feedback submission.

In order to receive a rating from a customer, the service provider SP is required to spend a certain amount of coins of the native cryptocurrency of the system. This approach is advantageous in a number of ways. It discourages the ballot stuffing attack, since the SP will need to spend coins proportional to the number of artificial ratings. Moreover, the cryptocurrency allows the system to incentivize mining its blockchain by rewarding the creation of new blocks with coins. The service providers can either mine the coins themselves or they may acquire the coins on open market from other miners. The system thus ensures the continuity of the blockchain through incentivized mining, which in turn also ensures the trustlessness property of the system.

A customer c can compute the reputation of a service provider SP by aggregating the ratings about the SP available in the public blockchain of the system. The ratings are aggregation function agnostic. Therefore, any aggregation function of the customer's choosing can be used for computing

Table 3.4: User Anonymity-Oriented Systems – Security Objectives.

	Privacy							**Integrity**				
System	**Multiple Pseudonyms**	**User-Pseudo Unlinkability**	**Pseudo-Pseudo Unlinkability**	**Rater Anonymity**	**Ratee Anonymity**	**Inquirer Anonymity**	**Reputation Transfer**	**Unforgeability**	**Distinctness**	**Accountability**	**Authorizability**	**Verifiability**
Blockchain-based Systems												
Schaub et al. 2016	●	●	●	●	○	●	○	●	○	○	●	●
Bazin et al. 2017	●	●	●	●	○	●	○	●	○	○	●	●
Dou et al. 2018	○	●		●	○	○	○	●	○	○	●	○
Kang et al. 2018	●	●		●	●	○	○	●	○	●	○	○
Lu et al. 2018	●	●	●	●	●		○	●	○	●	○	○
Owiyo et al. 2018	●	●	●	●	○	○	○	●	○		●	●
Jo and Choi 2019	○	●	○	●	○	○	○	●	○	●	○	●
Liu et al. 2019	○	●		●	○			●		●	●	●
Dimitriou 2021	●	●	●	●	●	○	●	●	○	○	●	●
Token-based Systems												
Androulaki et al. 2008	●	●	●	●	●	●	●	●	○	●	○	○
Schiffner et al. 2009	●	●	●	●	●	●	●	●	○	●	●	○
Schiffner et al. 2011	●	●	●	●	●	●	●	●	○		●	○
Zhang et al. 2014	●	●	●	●	○	○		●		●	●	
Busom et al. 2017	●	●	●	●	○		○	●			●	
Proxy-based Systems												
Petrlic et al. 2014	●	●	●	●	○	●	○	●			●	●
Mousa et al. 2017	●	●	●		●	○	●	●		○	●	○
Signature-based Systems												
Bethencourt et al. 2010	●	●	●	●	●		●	●	●			
Guo et al. 2013	●	●	●	●	●			●				
Lajoie-Mazenc et al. 2015	●	●	●	●	●	●	●	●	●		●	●
Chen et al. 2016	●	●	●	●	○		○	●		●	○	○
Transitory Pseudonym-based Systems												
Miranda and Rodrigues 2006	●	●	●	○	●	○	●	●	○	○	○	○
Steinbrecher 2006	●	●		○	●	○		●	○		●	
Anceaume et al. 2013	●	●	●	●	○	○		●	○	○	●	●
Christin et al. 2013	●	●	●	○	●	○	●	●			●	
Other Systems												
Kinateder and Pearson 2003	●	●	●	●	●	○		●	○	●	○	○
Bo et al. 2007	●	●	●	●	○	○	○		○		○	○

Legend

●	Property satisfied
◒	Property partially satisfied
○	Property not satisfied
	Property not specified or not applicable

Table 3.5: Feedback Confidentiality-Oriented Systems – Security Objectives.

	Privacy			**Integrity**			
System	**Confidentiality (Intermediate Info)**	**Confidentiality (Public Info)**	**Privacy of Relationships**	**Correct Range**	**Correct Computation**	**Authorizability**	**Verifiability**
Blockchain-based Systems							
Azad et al. 2018	●	◒		●	●	●	●
Bag et al. 2018	●	◒	●	●	●	○	●
Schiedermeier et al. 2019	●	○		◒	●	◒	●
Zhao et al. 2019	●	●		◒	●	●	
Azad et al. 2020	●	○	○	●	●	○	●
Zhang et al. 2020	●	○	○	●	●	●	○
SMPC-based Systems							
Pavlov et al. 2004	●	○		●	●	○	○
Gudes et al. 2009	●	○	◒	●	●	○	○
Nithyanand and Raman 2009	●	○		●	●	○	○
Gal-Oz et al. 2010	●	○	◒	●	●	○	○
Hasan et al. 2013	●	◒	○	●	●	○	○
Dimitriou and Michalas 2014	●	◒		●	●	○	○
Dolev et al. 2014	●	○		●	◒	○	○
Clark et al. 2016	●	○	○	●	●	○	○
Bakas et al. 2021	●	○		●	●	○	○
Token-based Systems							
Kerschbaum 2009	●	◒		●	●	●	●
Proxy-based Systems							
Ries et al. 2011	●	◒		●	●	○	○
Ma et al. 2018	●	●		◒	●	●	○

Legend

●	Property satisfied
◒	Property partially satisfied
○	Property not satisfied
	Property not specified or not applicable

Table 3.6: Countermeasures Against Common Attacks.

System	Sybil Attack	Ballot Stuffing	Slandering	Whitewashing	Oscillation	Random Ratings	Free Riding
Blockchain-based Systems							
Schaub et al. 2016	●	◒	◒	◒	○	○	●
Bazin et al. 2017	●	◒	○	●	◒	○	○
Azad et al. 2018	◒	◒	◒	◒	○	○	○
Bag et al. 2018	◒	◒	○	○	○	○	○
Dou et al. 2018	◒	○	○	◒	◒	○	○
Kang et al. 2018	●	◒	◒	◒	●	○	○
Lu et al. 2018	●	◒	●	●	●	●	
Owiyo et al. 2018	○	◒	◒	○	○	◒	○
Jo and Choi 2019	●	●	◒	●	○		
Liu et al. 2019	●	◒	○	○	○	○	○
Schiedermeier et al. 2019	◒	◒	○	○	○	○	○
Zhao et al. 2019	○	◒	◒	○	○	○	●
Azad et al. 2020	○	○	○	◒	○	○	○
Zhang et al. 2020	◒	●	●	○	●	○	◒
Dimitriou 2021	●	●	○	●	○	○	○
SMPC-Based Systems							
Pavlov et al. 2004	○	◒	◒	○	○	○	○
Gudes et al. 2009	◒	◒	◒	○	○	○	○
Nithyanand and Raman 2009	◒	◒	◒	○	○	○	○
Gal-Oz et al. 2010	◒	◒	◒	○	○	○	○
Hasan et al. 2013	○	○	○	○	○	○	○
Dimitriou and Michalas 2014	○	○	○	○	○	○	○
Dolev et al. 2014	○	◒	◒	○	○	◒	○
Clark et al. 2016	○	○	○	○	○	○	○
Bakas et al. 2021	○	○	○	○	○	○	○
Token-based Systems							
Androulaki et al. 2008	●	○	◒	◒	○	●	○
Kerschbaum 2009	●	◒	◒	●	○	○	●
Schiffner et al. 2009	●	○	◒	●	○	●	○
Schiffner et al. 2011	●	◒	◒	●	○	◒	○
Zhang et al. 2014	●		●	●	○	◒	○
Busom et al. 2017	●	●	◒	◒	○	◒	○
Proxy-based Systems							
Ries et al. 2011	◒	◒	◒	○	○	◒	○
Petrlic et al. 2014	●	◒	◒	●	●	◒	○
Mousa et al. 2017	●	●	●	●	◒	●	●
Ma et al. 2018	○	●	●	○	●	●	●
Signature-based Systems							
Bethencourt et al. 2010	●	●	●	◒	◒	●	●
Guo et al. 2013	●		◒	◒	◒		
Lajoie-Mazenc et al. 2015	●	◒	◒	●	○	○	○
Chen et al. 2016		◒	○	◒	◒	○	○
Transitory Pseudonym-based Systems							
Miranda and Rodrigues 2006	●	○	○	◒	○	○	
Steinbrecher 2006		○	○		○	○	
Anceaume et al. 2013	●	●	◒	●	◒	◒	○
Christin et al. 2013	●	●	●	●	○	●	
Other Systems							
Kinateder and Pearson 2003	◒	◒	○	◒	○	○	○
Bo et al. 2007	●	○	◒	○	○	○	○

Legend

●	Strong or explicit countermeasures
◒	Partial or implicit countermeasures
○	Weak or no countermeasures
	Countermeasures not specified or not applicable

the reputation. Moreover, the user can consult text reviews submitted along with the numerical ratings. If the reputation is acceptable, c generates a one time private/public key pair specifically for the transaction with SP.

After the transaction has taken place, c asks SP for a blinded token authenticating the transaction. SP can issue a token to c if SP has at least n coins available on her address on the blockchain. The n coins are necessary, since this amount will be deducted from SP upon submission of a rating by the customer. c then verifies the token and unblinds it, breaking the link between herself and the transaction. When c wishes to rate SP, she broadcasts a message containing SP's address, the unblinded token, and her rating. A miner of the blockchain who creates a new block then verifies and includes this rating in the block, which is eventually appended to the blockchain.

In addition to ballot stuffing, the system also offers resistance against bad mouthing. In order to submit a feedback about SP, a real transaction needs to take place and its cost needs to be paid to the service provider. It is therefore not possible for an adversary to submit frivolous negative feedback about the service provider without incurring a cost. A Sybil attack is not feasible for either the customer or the service provider since owning multiple addresses in the system does not provide any apparent adversarial advantage. The system is also fairly immune to free riding because (other than potentially generating some network traffic) consulting the blockchain for computing the reputation of a service provider does not directly draw any resources from the raters or the ratee. Moreover, the system is robust against out of range feedback since feedback is public and is verified by miners before integration into the blockchain.

As our analysis in Section 3.6 shows, the limitations of the system include the inability to guarantee ratee anonymity, reputation transfer, distinctness, and accountability. Moreover, the system does not offer strong countermeasures against ballot stuffing, slandering, and whitewashing attacks. No countermeasures are offered against the oscillation and random ratings attacks.

3.8 Future Research Directions

The fine-grained analysis and comparison of privacy-preserving reputation systems carried out in this survey, according to the proposed analysis frameworks, reveal a number of insights into this area of research. We are able to identify several future research directions, which are summarized in Table 3.7 and labeled as Di. Our complete discussion that leads to the identification of these future research directions is given in Chapter 12 (Section 13.1) as part of the general discussion on future work on privacy preservation in trust-deficient decentralized systems.

Table 3.7: Future Research Directions.

ID	**Description**
D1	The development of non-blockchain-based privacy-preserving reputation systems still holds importance and should continue in parallel with the development of blockchain-based systems.
D2	The blockchain technology should be leveraged to its full potential in order to build truly trustless systems.
D3	User anonymity-oriented systems should aim for guaranteeing ratee anonymity and inquirer anonymity.
D4	Reputation transfer and aggregation among pseudonyms should be given priority by user anonymity-oriented systems.
D5	The property of authorizability has been incorporated by a higher percentage of user anonymity-oriented systems in recent years than in the past. This trend of providing authorizability should continue.
D6	Future systems that do not prevent inference of feedback values from publicly available information, must take measures to either warn raters or prevent execution of protocol instances when their privacy is at risk.
D7	Feedback confidentiality-oriented systems should protect the privacy of relationships in addition to the confidentiality of feedback.
D8	Future work on feedback confidentiality-oriented privacy-preserving reputation systems should focus on the inclusion of the property of authorizability.
D9	Privacy-preserving reputation systems should be designed such that they provide comprehensive protection against the broad range of common attacks.

3.9 Conclusion

In this survey, we presented an in-depth analysis of a broad range of privacy-preserving reputation systems. We proposed an analysis framework that decomposes privacy-preserving reputation systems according to the following dimensions: the nature of the adversary, reputation binding, the trust model, the security objectives of the system, and the building blocks utilized. Additionally, we identified the security requirements of privacy-preserving reputation systems that cut across multiple types of such systems. It is observed that there are two main types of privacy-preserving reputation systems: 1) systems that preserve the anonymity of the users, and 2) systems that don't necessarily preserve the anonymity of the users but preserve the confidentiality of their feedback. We noted that the security-related requirements can be further subdivided into privacy requirements and integrity requirements. We also presented an analysis framework that covers the fundamental elements that are common to all reputation systems. The following elements were identified for this framework: the architecture of the system, the properties of the feedback, the properties of the reputation, the feedback aggregation model, the attacks addressed, and the reputation query costs.

We conducted a fine-grained analysis and comparison of 44 privacy-preserving reputation systems using our analysis frameworks. We established several categories of systems according to their security mechanisms and classified the privacy-preserving reputation systems according to these categories. Our detailed comparison of privacy-preserving reputation systems in a normalized manner using our analysis frameworks reveals the differences between the systems in the literature as well as their chronological evolution. The survey presented detailed descriptions of a number of blockchain-based systems, which included the first trustless decentralized system by Schaub et al. [255] as well as more recent systems. We discussed the details of their protocols and security approaches as well as highlighted their individual strengths and other salient features.

Our fine-grained analysis, comparison, and discussion led to the identification of a number of insights into this area of research. We observed that the advent of the blockchain technology has provided a fresh impetus to research on privacy-preserving reputation systems. A majority of the systems published since 2016 that are listed in this survey utilize blockchain as one of the building blocks. However, we also noted that one of the future directions is to leverage the blockchain technology to its full potential and build truly trustless systems. We looked at the success of the surveyed systems in guaranteeing the security of users. It was observed that a high majority of both anonymity-oriented and feedback confidentiality-oriented systems are able to guarantee their respective essential privacy and integrity properties. However, there are also many properties that have been mostly ignored. We identified authorizability as one of the important properties that needs to be addressed by systems in the future. Lastly, analyzing the systems in terms of their countermeasures against common attacks, we observed that designing systems that provide comprehensive protection against a broad range of attacks is an evident direction for future research in the area.

Chapter 4

A Decentralized Privacy-Preserving Reputation Protocol for the Malicious Adversarial Model

Users hesitate to submit negative feedback in reputation systems due to the fear of retaliation from the recipient user. A privacy preserving reputation protocol protects users by hiding their individual feedback and revealing only the reputation score. We present a privacy preserving reputation protocol for the malicious adversarial model. The malicious users in this model actively attempt to learn the private feedback values of honest users as well as to disrupt the protocol. Our protocol does not require centralized entities, trusted third parties, or specialized platforms, such as anonymous networks and trusted hardware. Moreover, our protocol is efficient. It requires an exchange of $O(n + log\ N)$ messages, where n and N are the number of users in the protocol and the environment respectively.

This chapter is an adapted version of the article: "A Decentralized Privacy Preserving Reputation Protocol for the Malicious Adversarial Model." O. Hasan, L. Brunie, E. Bertino, and N. Shang. *IEEE Transactions on Information Forensics and Security.* 2013. Vol. 8, no. 6, pp 949-962.

4.1 Introduction

We consider a multi-agent environment composed of N agents. Let us consider an agent t in the environment and refer to it as the target agent. Agent t has interacted with $n < N$ other agents in the environment who have assigned it private feedback values. We refer to these feedback providers as the source agents of the target agent t. The Malicious-k-shares protocol allows a querying agent q to compute the reputation of a target agent t as the mean of the private feedback values held by its source agents. The protocol aims to preserve the privacy of the source agents by preventing disclosure of their private feedback values under the malicious adversarial model (described in Section 4.2.2). The novel contributions of our work are summarized below.

In the Malicious-k-shares protocol, an agent can preserve its privacy by partially trusting on only k fellow feedback providers, where k is much smaller than $n - 1$, the size of the set of fellow feedback providers. This idea is central in our protocol and allows us to build a protocol that requires an exchange of only $O(n + log\ N)$ messages and $O(n^2 + log\ N)$ bytes of information, where n and N are the number of agents in the protocol and the environment respectively. This approach improves on the classic approach (as employed by Gudes et al. [128] and Pavlov et al. [235]) where an agent is required to partially trust on all $n-1$ fellow feedback providers to preserve its privacy which results in high communication complexity. In this work, we use three real and large trust graphs to demonstrate that a high majority of agents can find k sufficiently trustworthy agents in a set of $n - 1$ fellow feedback providers such that k is very small compared to $n - 1$ (Section 4.5.2).

Agents in our protocol can quantify the risk to their privacy before submitting their feedback. This allows us to extend the protocol such that agents can abstain if the risk to their privacy is above the desired threshold. We show using the three real trust graphs that even if agents abstain, accurate reputation values can be computed from the feedback submitted by only those agents whose privacy is preserved (Section 4.5.3).

The Malicious-k-shares protocol prevents malicious agents from taking two actions that are particularly challenging to address in a decentralized privacy preserving reputation system without relying on trusted third parties: (1) A malicious agent can take advantage of private feedback and submit a value that is outside the legal interval for feedback. For example, a malicious agent can submit a value such as -99 when the feedback interval is $[0, 1]$. (2) A malicious agent can make erroneous computations, for example, it can report a random number instead of reporting a correct sum.

The protocol addresses the above challenges through innovative constructions (described in Section 4.4.1) based on set-membership and plain-text equality non-interactive zero-knowledge proofs and an additive homomorphic cryptosystem. To the best of our knowledge, our protocol is the most efficient decentralized additive privacy preserving reputation protocol under the malicious adversarial model. It requires the exchange of only $O(n + log\ N)$ messages and $O(n^2 + log\ N)$ bytes of information. Compare this to the protocol by Pavlov et al. [235] that requires $O(n^3 + N)$ messages and at least $O(n^3 + N)$ bytes of information using similar building blocks.

4.1.1 Outline

In Section 4.2, we give a general framework for decentralized privacy preserving reputation systems in the malicious adversarial model. In Section 4.3, we describe some building blocks that we utilize in the construction of our protocol. In Section 4.4, we present our proposal for a new decentralized privacy preserving reputation protocol. We also analyze the security and the complexity of the protocol in this section. In Section 4.5, we use three real trust graphs to experimentally evaluate two hypotheses that the protocol is based on. In Section 4.6, we give a comparison of our protocol with other reputation systems in the literature. We conclude the work in Section 4.7.

4.2 Framework

In this section, we establish a framework that allows us to describe and analyze the protocol in Section 4.4. However, the reader may skip directly to Section 4.4.1 for a quick overview of the protocol without delving into the specifics of the framework.

4.2.1 Agents, Trust, and Reputation

We model our environment as a multi-agent environment. An agent represents a user. Let $\mathbb{A}$ denote the set of all agents in the environment. $|\mathbb{A}| = N$.

We subscribe to the definition of trust by sociologist Diego Gambetta [116], which characterizes trust as binary-relational, directional, contextual, and quantifiable as subjective probability. Our formal definition of trust attempts to capture each of these characteristics.

Let $\mathbb{D}$ denote an asymmetric binary relation on the set $\mathbb{A}$. Let $\mathbb{T} \subseteq \mathbb{D}$ be the set of all existing trust relationships between agents. $(s, t) \in \mathbb{T}$, where $s, t \in \mathbb{A}$, implies that an agent s has a trust relationship toward an agent t.

Let Ψ denote the set of all actions. Examples of actions include: "prescribe correct medicine", "deliver product sold online", "preserve privacy", etc.

Let $perform$ denote a function, such that $perform : \mathbb{T} \times \Psi \rightarrow \{true, false\}$. The function $perform(s, t, \psi)$ outputs $true$ if agent t performs the action ψ anticipated by agent s, or it outputs $false$ if t does not perform the anticipated action. Let the subjective probability $P(perform(s, t, \psi) = true)$ denote agent s's belief that agent t will perform the action ψ.

Definition 1. ***Trust.*** *The trust of an agent s in an agent t is given as the triple* $\langle s\mathbb{T}t, \psi, P(perform(s, t, \psi) = true)\rangle$, *where* $s, t \in \mathbb{A}$, $(s, t) \in \mathbb{T}$, $\psi \in \Psi$, *and* $P(perform(s, t, \psi) = true) \in [0, 1]$.

When the context of trust (action ψ) is clear, we adopt the simplified notation f_{st} for $P(perform(s, t, \psi) = true)$. We can also refer to f_{st} as agent s's feedback about agent t.

An agent s is said to be a source agent of an agent t in the context of an action ψ if s has a trust relationship toward t in the context ψ. The set of all source agents of an agent t in context ψ is given as $\mathbb{S}_{t,\psi}$. The simplified notation $\mathbb{S}_t$ is used instead of $\mathbb{S}_{t,\psi}$ when the context ψ is clear.

Definition 2. ***Reputation.*** *Let $\mathbb{S}_t = \{s_1 \dots s_n\}$ be the set of source agents of an agent t in context ψ. The reputation of agent t in context ψ is given as:*

$$rep_{\oplus}(f_{s_1 t} \dots f_{s_n t}) = \frac{\sum_{i=1}^{n} f_{s_i t}}{n} \tag{4.1}$$

The function $rep_{\oplus}$ implements the reputation of an agent t as the mean of the feedback values of its source agents. The reputation of an agent t is denoted by $r_{t,\psi}$, or r_t when the context ψ is clear.

A limitation of the reputation protocol that we present in this chapter is that it can only compute reputation as the mean of the feedback values. An alternative function that can be implemented by the protocol is the sum of the feedback values. However, other possible functions for computing reputation such as weighted mean and probability distribution are not supported by the current protocol.

We note that an advantage of reputation computed as mean is that it is an intuitive statistic. The eBay reputation system (`ebay.com`), which is one of the most successful reputation systems, represents reputation as the simple sum of feedback values. We derive the mean from the sum in order to normalize the reputation values. However, it is an interesting avenue for future work to explore efficient decentralized privacy preserving solutions for computing reputation as weighted mean, probability distribution, etc.

Definition 3. ***Reputation Protocol.*** *Let Π be a multi-party protocol. Then Π is defined as a Reputation Protocol, if (1) the participants of the protocol include: a querying agent q, a target agent t, and n source agents of t in the context ψ, (2) the inputs include: the feedback of the source agents in context ψ, and (3) the output of the protocol is: agent q learns the reputation $r_{t,\psi}$ of agent t.*

4.2.2 Adversary

We refer to the coalition of dishonest agents as the adversary. In this work, we propose a solution for the malicious adversarial model. Malicious agents actively attempt to learn private information of honest agents as well as to disrupt the protocol. Specifically, malicious agents may (1) refuse to participate in the protocol, (2) prematurely abort the protocol, (3) selectively drop messages that they are supposed to send, (4) tamper with the communication channels, (5) wiretap the communication channels, and (6) provide illegal information (for example, provide out of range values as their inputs, make incorrect computations).

4.2.3 Privacy

Definition 4. ***Preservation of Privacy (by an Agent).*** *Let x be an agent s's private data that agent s reveals to an agent u. Then agent u is said to preserve the privacy of agent s w.r.t. x, if (1) u does not use x to infer more information, and (2) u does not reveal x to any third party.*

Let action $\rho =$ "preserve privacy".

Definition 5. ***Trusted Third Party (TTP).*** *Let $\mathbb{S} \subseteq \mathbb{A}$ be a set of n agents, and $TTP_{\mathbb{S}} \in \mathbb{A}$ be an agent. Then $TTP_{\mathbb{S}}$ is a Trusted Third Party (TTP) for the set of agents $\mathbb{S}$ if for each $s \in \mathbb{S}$, $P(perform(s, TTP_{\mathbb{S}}, \rho) = true) = 1$.*

We define *security threshold* as a parameter that can be assigned a value in $[0, 1]$ according to the security needs of an application. A value of the *security threshold* closer to 1 indicates a stricter security requirement. We consider as *high* any probability greater than or equal to the *security threshold*, and as *low* any probability less than $1-$ *security threshold*.

We adopt the Ideal-Real approach [121] to define privacy preserving reputation protocols.

Definition 6. ***Ideal Privacy Preserving Reputation Protocol.*** *Let Π be a reputation protocol (Definition 3). Then Π is an ideal privacy preserving reputation protocol under a given adversarial model, if: (1) the inputs of all n source agents of t are private, (2) $TTP_{\mathbb{S}_t}$ is a participant, where $\mathbb{S}_t \equiv \mathbb{S}_{t,\psi}$ is the set of all source agents, (3) $m < n$ of the source agents (given as set $\mathbb{M}$) and agents q and t are considered to be dishonest, however, q wishes to learn the correct output, (4) agents $\mathbb{S}_t - \mathbb{M}$ and $TTP_{\mathbb{S}_t}$ are honest, (5) as part of the protocol, $TTP_{\mathbb{S}_t}$ receives the private inputs from the source agents and outputs the reputation $r_{t,\psi}$ to agent q, and (6) over the course of the protocol, the private input of each agent $s \in \mathbb{S}_t - \mathbb{M}$ may be revealed only to the $TTP_{\mathbb{S}_t}$.*

In an ideal privacy preserving reputation protocol, it is assumed that for each agent $s \in \mathbb{S}_t - \mathbb{M}$, the adversary does not gain any more information about the private input of agent s from the protocol other than what he can deduce from what he knows before the execution of the protocol and the output, with probability $P(perform(s, TTP_{\mathbb{S}_t}, \rho) = true) = 1$.

Definition 7. ***Real Privacy Preserving Reputation Protocol.*** *Let I be an ideal privacy preserving reputation protocol (Definition 6). Then R is a real privacy preserving reputation protocol w.r.t. I, if: (1) R has the same parameters (participants, private inputs, output, adversary, etc.) as I, except that there is no $TTP_{\mathbb{S}_t}$ as a participant (2) with high probability, the adversary learns no more information about the private input of any agent s than it can learn in protocol I.*

4.2.4 Problem Definition

Let $\mathbb{S}_{t,\psi} = \{s_1 \dots s_n\}$ be the set of all source agents of agent t in the context of action ψ. Find a reputation protocol Π, which takes private input $f_{st} \equiv P(perform(s, t, \psi) = true)$ from each agent $s \in \mathbb{S}_t$, and outputs the reputation $r_{t,\psi}$ of the target agent t to a querying agent q. Reputation is computed as $rep_{\oplus}$. Agents q, t, and an additional m of the source agents are considered to be dishonest, where $m < n$. However, q wishes to learn the correct output and therefore does not take any action that alters the output. The reputation protocol Π is required to be decentralized and secure under the malicious adversarial model. If computing $r_{t,\psi}$ is not possible due to the disruptive actions of certain agents, then the protocol outputs the identity of those agents to the querying agent q.

4.3 Building Blocks

4.3.1 Additive Homomorphic Cryptosystem

Let $E_s(.)$ denote the encryption function with the public key PK_s of agent s in an asymmetric cryptosystem $\mathcal{C}$. The cryptosystem $\mathcal{C}$ is said to be additive homomorphic if we can compute $E_s(x + y)$, given only $E_s(x)$, $E_s(y)$, and PK_s. As an example, let us consider two integers, 3 and 4. A cryptosystem $\mathcal{C}$ is additive homomorphic if given only $E_s(3)$, $E_s(4)$, and PK_s, we are able to obtain $E_s(3 + 4) = E_s(7)$.

We use the Paillier cryptosystem [230] as it offers the following properties (in addition to additive homomorphic encryption) that allow us to construct a secure reputation protocol:

- **Randomized encryption.** Randomized encryption implies that an attacker cannot distinguish between the encryptions of different plaintexts even if the plaintexts and the key are known. For example, consider an attacker who is given two integers, 3 and 4, their encryptions, $E_s(3)$ and $E_s(4)$, and the encrypting public key PK_s. The attacker is unable to draw correspondence between the ciphertexts $E_s(3)$, $E_s(4)$, and the integers 3, 4. Cryptosystems that do not support randomized encryption (for example, RSA [251] without padding), always generate the same ciphertext for a given pair of plaintext and encryption key. Such cryptosystems are not suitable when the plaintext space is small (for example, a plaintext space such as $\{1, 2, 3, 4, 5\}$).

- **Non-interactive zero-knowledge proofs.** The Paillier cryptosystem allows construction of efficient non-interactive zero-knowledge proofs of set membership (Section 4.3.2) and plaintext equality (Section 4.3.3).

The additive homomorphic encryption property of the Paillier cryptosystem can be informally stated as: $E_s(\mathtt{m}_1) \times E_s(\mathtt{m}_2) = E_s(\mathtt{m}_1 + \mathtt{m}_2)$, where $\mathtt{m}_1$ and $\mathtt{m}_2$ are two plaintext messages. This implies that the multiplication of two ciphertexts gives the encrypted sum of their plaintexts.

4.3.2 Zero-Knowledge Proof of Set Membership

Let $\mathbb{F} = \{\mathtt{m}_1, \dots, \mathtt{m}_p\}$ be a public set of p messages, and $E(\mathtt{m}_i)$ be an encryption of $\mathtt{m}_i$ with a prover's public key, where $\mathtt{m}_i$ is secret. A zero-knowledge proof of set membership allows the prover to convince a verifier that $E(\mathtt{m}_i)$ encrypts a message in $\mathbb{F}$.

A standard interactive zero-knowledge proof comprises of three moves, that is, three messages exchanged between the prover and the verifier. In the first move, the prover sends a cryptographic commitment to the verifier. In the second move, the verifier sends a random challenge to the prover to test the commitment. The third move is the prover's response to the random challenge of the verifier. An interactive zero-knowledge proof can be converted to a non-interactive zero-knowledge proof using the Fiat-Shamir heuristic [110]. In a non-interactive proof, there is only one move made by the prover in which he sends the cryptographic commitment as well as a hash of the commitment to the verifier. The proof can be verified with this supplemental information without the need for additional moves.

In a non-interactive version of the zero-knowledge proof of set membership, we abstract the part of the proof generated by the prover as the function $setMembershipZKP(E(\mathtt{m}_i), \mathbb{F})$, abbreviated as $smzkp(E(\mathtt{m}_i), \mathbb{F})$.

The zero-knowledge proof of set membership is specified for the Paillier cryptosystem as follows: Let $(\mathtt{n}, g)$ be a prover's public key, $\mathbb{F} = \{\mathtt{m}_1, \ldots, \mathtt{m}_p\}$ be a public set of p messages, and $c = g^{\mathtt{m}_i} \cdot \mathtt{r}^{\mathtt{n}}\ mod\ \mathtt{n}^2$ an encryption of $\mathtt{m}_i$, where i is secret, and $\mathtt{r}$ is a random integer. A zero-knowledge proof of set membership allows the prover to convince a verifier that c encrypts a message in $\mathbb{F}$.

4.3.3 Zero-Knowledge Proof of Plaintext Equality

Let $E_u(\mathtt{m})$ and $E_v(\mathtt{m})$ be the encryptions of a message $\mathtt{m}$ with the public key of agents u and v respectively. A zero-knowledge proof of plaintext equality allows a prover to convince a verifier that $E_u(\mathtt{m})$ and $E_v(\mathtt{m})$ encrypt the same message.

In a non-interactive version of the zero-knowledge proof of plaintext equality, we abstract the part of the proof generated by the prover as the function $plaintextEqualityZKP(E_u(\mathtt{m}), E_v(\mathtt{m}))$, abbreviated as $pezkp(E_u(\mathtt{m}), E_v(\mathtt{m}))$.

The zero-knowledge proof of plaintext equality is specified for the Paillier cryptosystem as follows: Let $(\mathtt{n}_1, g_1)$ and $(\mathtt{n}_2, g_2)$ be the public keys of agents 1 and 2 respectively. Given two encryptions $c_1 = g_1^{\mathtt{m}} \cdot \mathtt{r}_1^{\mathtt{n}_1}\ mod\ \mathtt{n}_1^2$ and $c_2 = g_2^{\mathtt{m}} \cdot \mathtt{r}_2^{\mathtt{n}_2}\ mod\ \mathtt{n}_2^2$, a zero-knowledge proof of plaintext equality allows a prover to convince a verifier that c_1 and c_2 encrypt the same message.

4.3.4 Source Managers

We define a *source manager* of an agent t as a fellow agent who maintains the set $\mathbb{S}_t$. The idea of source managers is inspired by *score managers* in EigenTrust [162]. When a source agent assigns feedback to a target agent t, it reports that event to each of its source managers. The source managers add the source agent to the set $\mathbb{S}_t$ that they each maintain. A Distributed Hash Table (DHT), such as Chord [274], is used to locate the source managers.

It is important to note that the source managers are considered to be potentially dishonest. To learn a set $\hat{\mathbb{S}_t} \supset \mathbb{S}_t$, a querying agent can retrieve the set maintained by each of the source managers and take a union of the sets. The querying agent will learn $\hat{\mathbb{S}_t} \supset \mathbb{S}_t$ as long as at least one of the source managers is honest. Let us consider that there is an even probability that any given source manager is either honest or dishonest. Then the probability that at least one of all η_t source managers of an agent t will be honest is $1 - \frac{1}{2^{\eta_t}}$. This probability is 75% at $\eta_t = 2$, 97% at $\eta_t = 5$, and 99% at $\eta_t = 7$.

4.4 The Malicious-k-shares Protocol

4.4.1 Protocol Overview

In the Malicious-k-shares protocol, each source agent s relies on at most k agents to preserve his privacy. Agent s selects these k agents based on his own knowledge of their trustworthiness in the context of preserving privacy and sends each of them an additive share of his private feedback value. The advantages of this approach are twofold. Firstly, an agent is able to quantify and maximize the probability that its privacy will be preserved. This also allows us to extend the protocol such that an agent can abstain from providing feedback if the risk to its privacy is high. Secondly, limiting the number of shares to $k \ll n$, results in a protocol that requires an exchange of only $O(n + log\ N)$ messages and $O(n^2 + log\ N)$ bytes of information.

In the Malicious-k-shares protocol, each source agent s must prove that it has generated correct shares, that is, the sum of all shares is a value that lies in the legal interval for feedback. An agent s sends each of the k trusted agents a share encrypted with the recipient agent's public key. The shares are relayed through the querying agent q. We would like q to add these shares using the additive homomorphic property, however, this is not possible because the shares are encrypted with different keys. As a solution, agent s also encrypts each of the shares with his own public key and submits them to q. Additionally, it submits a set-membership zero-knowledge proof that the sum of these shares belongs to the legal interval. The querying agent can verify the veracity of this claim by using the additive homomorphic property to add the set of shares encrypted with agent s's key and then by verifying the proof. It still remains to show that the original shares sent to the trusted agents are correct. To show this, agent s gives a plaintext-equality zero-knowledge proof for each share that shows that a share encrypted with the recipient's public key and a share encrypted with the sender's public key contain the same plaintext. Verification of the equality of all pairs of shares verifies that agent s indeed sent correct shares.

In the Malicious-k-shares protocol, each source agent s must prove that it has computed the correct sum of the shares. The querying agent q can compute the encrypted sum of the shares from the copies of the encrypted shares that it received and relayed to agent s. However, q cannot decrypt the sum because it is encrypted with the recipient agent's public key. Agent s computes the sum and sends it to q encrypted with q's public key. Agent s also sends a plaintext-equality zero-knowledge proof that shows that the encrypted sum independently computed by q and the encrypted sum sent by agent s hold the same value. Verification of the proof convinces q that agent s computed the sum correctly.

4.4.2 Protocol Outline

The important steps of the protocol are outlined below. The steps 1, 7, 8, 13, and 14 are performed by the querying agent q. Whereas, the steps $2-6$, and $9-12$, are performed by each source agent $s \in \mathbb{S}_t$.

1. **Initiation.** The protocol is initiated by a querying agent q to determine the reputation $r_{t,\psi}$ of a target agent t. Agent q retrieves $\mathbb{S}_t \equiv \mathbb{S}_{t,\psi}$, which is the set of source agents of agent t in the context ψ. Agent q verifies $\mathbb{S}_t$ from the source managers of t. Agent q then sends $\mathbb{S}_t$ to each source agent $s \in \mathbb{S}_t$.

2. **Select Trustworthy Agents.** Each agent $s \in \mathbb{S}_t$ selects k other agents in $\mathbb{S}_t$. Let us refer to these agents selected by s as the set $\mathbb{U}_s = \{u_{s,1} \dots u_{s,k}\}$. Agent s selects these agents such that: $P(\neg perform(s, u_{s,1}, \rho)) \times \dots \times P(\neg perform(s, u_{s,k}, \rho))$ is low. That is, the probability that all of the selected agents will collude to breach agent s's privacy is low.

3. **Prepare Shares.** Agent s then prepares $k+1$ shares of its secret feedback value f_{st}. The shares, given as: $x_{s,1} \dots x_{s,k+1}$, are prepared in the following manner: The first k shares are random numbers uniformly distributed over a large interval (for example, $[0, 2^{32}-1]$). The last share is selected as follows: $x_{s,k+1} = (f_{st} - \sum_{i=1}^{k} x_{s,i}) \ mod\ M$, where M is a publicly known modulus.

   The preparation of the shares in this manner implies that: $(\sum_{i=1}^{k+1} x_{s,i}) \ mod\ M = f_{st}$. That is, the sum of the shares $mod\ M$ is equal to the feedback value. The sum of the shares, $\sum_{i=1}^{k+1} x_{s,i}$, lies in the interval $[(h_s \times M), (h_s \times M) + F]$, where $h_s = (\sum_{i=1}^{k+1} x_{s,i}) \ div\ M$, and $f_{st} \in [0, F]$.

   Since each of the $k+1$ shares is a number uniformly distributed over a large interval, no information about the secret can be learned unless all of the shares are known.

4. **Encrypt Shares.** Agent s then encrypts each of the $k+1$ shares with its own public key to obtain: $E_s(x_{s,1}) \dots E_s(x_{s,k+1})$. It also encrypts each share $x_{s,i}$ with the public key of agent $u_{s,i}$, for $i \in \{1 \dots k\}$, to obtain: $E_{u_{s,1}}(x_{s,1}) \dots E_{u_{s,k}}(x_{s,k})$.

5. **Generate Zero-Knowledge Proofs.** Agent s computes: $\beta_s = (E_s(x_{s,1}) \times \dots \times E_s(x_{s,k+1})) \ mod\ \mathfrak{n}_s^2$, where $\mathfrak{n}_s$ is the RSA modulus in the public key of agent s. The result of this product is the encrypted sum of agent s's shares, that is $\beta_s = E_s(\sum_{i=1}^{k+1} x_{s,i})$ (due to the additive homomorphic property).

   Agent s then generates one non-interactive set membership zero-knowledge proof: $smzkp(\beta_s, [(h_s \times M), (h_s \times M) + F])$. The proof proves to a verifier that the ciphertext β_s encrypts a value that lies in the interval $[(h_s \times M), (h_s \times M) + F]$. In other words, the proof shows that the ciphertext contains a valid feedback value (considering $mod\ M$).

   Agent s also generates k non-interactive plaintext equality zero-knowledge proofs. Each proof $pezkp(E_s(x_{s,i}), E_{u_{s,i}}(x_{s,i}))$, where $i \in \{1 \dots k\}$, proves to a verifier that the two ciphertexts, one encrypted with the public key of s and the other encrypted with the public key of $u_{s,i}$, contain the same plaintext.

   A verifier who verifies these zero-knowledge proofs will be convinced that agent s has prepared the shares such that they add up to a correct feedback value. Moreover, the verifier will be assured that the shares destined for s's trustworthy agents correspond to those correct shares.

6. **Send Encrypted Shares and Proofs.** Agent s sends all encrypted shares, that is, $E_s(x_{s,1}) \dots E_s(x_{s,k+1})$ and $E_{u_{s,1}}(x_{s,1}) \dots E_{u_{s,k}}(x_{s,k})$, as well as all zero-knowledge proofs, that is, $smzkp(\beta_s, [(h_s \times M), (h_s \times M) + F])$ and $pezkp(E_s(x_{s,i}), E_{u_{s,i}}(x_{s,i}))$, $i \in \{1 \dots k\}$, to agent q.

7. **Verify the Proofs.** Agent q independently computes β_s and verifies the proofs received from each agent s. Their verification confirms that agent s has prepared the shares correctly. Agent q receives and verifies the proofs of all source agents before proceeding to the next step.

8. **Relay the Encrypted Shares.** Agent q relays to each agent $u \in \mathbb{S}_t$, the encrypted shares received for it from agents who considered it trustworthy. That is, each encrypted share $E_{u_{s,i}}(x_{s,i})$, prepared by an agent s for agent $u_{s,i}$, is relayed to agent $u_{s,i}$.

   The shares are relayed through agent q, therefore, any agent who drops a message would be easily identified. However, agent q does not learn any of the shares by relaying them since the shares are encrypted with the public keys of the destination agents.

9. **Compute Sum of the Shares.** Each agent $s \in \mathbb{S}_t$ receives the encrypted shares of the agents who considered it trustworthy. Agent s computes γ_s as the product of those encrypted shares along with the ciphertext of its own $(k+1)$'th share $x_{s,k+1}$. Due to the additive homomorphic property, γ_s is an encryption of the sum of the plaintexts of those shares. Agent s decrypts γ_s to obtain the plaintext sum σ_s.

   Adding the $(k+1)$'th share provides security in the case when s receives only one share. If there is no $(k+1)$'th share to add, then agent q would learn the received share. Secrecy of the $(k+1)$'th share itself is not critical to the security of the protocol.

10. **Encrypt the Sum.** Agent s then encrypts σ_s with agent q's public key to obtain $E_q(\sigma_s)$.

11. **Generate Zero-Knowledge Proof.** Agent s then generates a non-interactive plaintext equality zero-knowledge proof: $pezkp(\gamma_s, E_q(\sigma_s))$. The proof proves to a verifier that the two ciphertexts, one encrypted with the public key of s and the other encrypted with the public key of q, contain the same plaintext.

    Agent q, who can independently compute γ_s, can be convinced by this proof that $E_q(\sigma_s)$ contains the correct sum of the shares.

12. **Send Encrypted Sum and Proof.** Agent s sends the encrypted sum $E_q(\sigma_s)$ and the zero-knowledge proof $pezkp(\gamma_s, E_q(\sigma_s))$ to agent q.

13. **Verify the Proof.** Agent q independently computes γ_s and verifies the zero-knowledge proof received from each agent s. Its verification confirms that the agent has computed the sum of the shares correctly. Agent q receives and verifies the proofs of all source agents before proceeding to the next step.

14. **Compute Reputation.** Agent q decrypts $E_q(\sigma_s)$ to obtain σ_s for each agent $s \in \mathbb{S}_t$. Agent q then computes $r_{t,\psi} = ((\sum_{s \in \mathbb{S}_t} \sigma_s) \ mod \ M)/n$.

4.4.3 Protocol Specification

The protocol is specified in Figures 4.1 and 4.2.

For the purpose of this protocol, we consider feedback values to be integers in the range $[0, F]$ (for example, $[0, 10]$). The reputation computed by the protocol can be normalized to the interval $[0, 1]$ by dividing the result by F.

Let M be a publicly known modulus, such that $M > F$, and $\forall t \in A : \sum_{s \in \mathbb{S}_t} f_{st} < M$. Moreover, M is sufficiently smaller than $2^{\Bbbk}$, where $\Bbbk$ is the security parameter — the length in bits of the RSA modulus $\mathbb{n}$ in the cryptographic keys of the agents (for example, $\Bbbk = 2048$, and $M = 2^{80}$).

Let $[0, X]$ be a large interval (for example, $[0, 2^{32} - 1]$).

To generate the zero-knowledge proof $setMembershipZKP(\beta_s, [(h_s \times M), (h_s \times M) + F])$ in step 10 of the event PREP, an agent s requires the randomization $\mathbb{r}_{\beta_s}$ of the encryption β_s, which can be computed as follows: $\mathbb{r}_{\beta_s} = \mathbb{r}_{s,1} \times \dots \times \mathbb{r}_{s,k+1}$, where $\mathbb{r}_{s,i}$ is the randomization used for the encryption of $E_s(x_{s,i})$.

To generate the zero-knowledge proof $plaintextEqualityZKP(\gamma_s, E_q(\sigma_s))$ in step 4 of the event VERIFIED_SHARES, an agent s requires the randomization $\mathbb{r}_{\gamma_s}$ of the encryption γ_s, which can be computed as follows: $\mathbb{r}_{\gamma_s} = (g^{-\sigma_s} \cdot \gamma_s)^{1/\mathbb{n}_s \ mod \ (\mathbb{p}-1)(\mathbb{q}-1)} \ mod \ \mathbb{n}_s$, where g and $\mathbb{n}_s$ are in the public key of s, and $\mathbb{p}$ and $\mathbb{q}$ are in the secret key, $\mathbb{n}_s = \mathbb{p}\mathbb{q}$.

The protocol uses the following functions: **timestamp**() – Returns current time. **random**(ξ,ζ) – Returns a random number uniformly distributed over the interval $[\xi, \zeta]$, where $\xi < \zeta$. **set_of_trustworthy**$(s, \mathbb{S})$ – Returns a set of agents $\mathbb{U}_s = \{u_{s,1} \dots u_{s,k}\}$, where $\mathbb{U}_s \subseteq \mathbb{S}$. The set $\mathbb{U}_s$ is selected such that: $P(\neg perform(s, u_{s,1}, \rho)) \times \dots \times P(\neg perform(s, u_{s,k}, \rho))$ is low.

Protocol: Malicious-k-shares
Participants: Agents: q, t, and the agents in the set $\mathbb{S}_t \equiv \mathbb{S}_{t,\psi} = \{s_1 \dots s_n\}$. Agents q, t, and a subset of $\mathbb{S}_t$ of size $m < n$ are considered to be dishonest, however, q wishes to learn the correct output and therefore does not disrupt the protocol. $n \geq 3$.
Input: Each agent $s \in \mathbb{S}_t$ has a private input $f_{st} \equiv P(perform(s, t, \psi) = true)$.
Output: Agent q learns $r_{t,\psi}$, the reputation of agent t in the context ψ, or agent q learns the identity of the agents who disrupt the protocol.
Setup: Agent t maintains $\mathbb{S}_t \equiv \mathbb{S}_{t,\psi}$, the set of its source agents in the context ψ. All communication takes place over authenticated point-to-point channels that are resistant to wire-tapping and tampering.
Events and Associated Actions:

agent q initiates the protocol to determine r_t

```
1  send tuple $(\text{REQUEST\_FOR\_SOURCES}, \psi)$ to $t$
2  receive tuple $(\text{SOURCES}, \psi, \mathbb{S}_t)$ from $t$
3  verify $\mathbb{S}_t$ from the source managers of $t$
4  retrieve the public key $PK_s$ of each agent $s \in \mathbb{S}_t$ from a certificate authority
5  $\mathbb{S}'_t \leftarrow \mathbb{S}_t$ ▷ initialize the set of agents who are expected to send their shares
6  $\theta \leftarrow 0$ ▷ a cumulative sum for computing reputation
7  $\mathbb{V}_s \leftarrow \phi$, for each agent $s \in \mathbb{S}_t$ ▷ initialize the sets of encrypted shares
8  $\tau \leftarrow \text{timestamp}()$
9  send tuple $(\text{PREP}, q, t, \tau, \mathbb{S}_t)$ to each agent $s \in \mathbb{S}_t$
```

agent t receives the tuple $(\text{REQUEST\_FOR\_SOURCES}, \psi)$ from agent q

```
1  send tuple $(\text{SOURCES}, \psi, \mathbb{S}_t)$ to $q$
```

an agent $s \in \mathbb{S}_t$ receives the tuple $(\text{PREP}, q, t, \tau, \mathbb{S}_t)$ from agent q

```
    ▷ select trustworthy agents
 1  $\mathbb{U}_s \leftarrow \text{set\_of\_trustworthy}(s, \mathbb{S}_t - \{s\})$
    ▷ prepare shares
 2  for $i \leftarrow 1$ to $k$
 3      do $x_{s,i} \leftarrow \text{random}(0, X)$
 4  $x_{s,k+1} \leftarrow (f_{st} - \sum_{i=1}^{k} x_{s,i}) \ mod \ M$
 5  $h_s \leftarrow (\sum_{i=1}^{k+1} x_{s,i}) \ div \ M$
    ▷ retrieve public keys
 6  retrieve the public key of each $u \in \mathbb{U}_s$ and the public key of $q$
    from a certificate authority
    ▷ encrypt shares
 7  encrypt $x_{s,1} \dots x_{s,k+1}$ with the public key of $s$
    to obtain $E_s(x_{s,1}) \dots E_s(x_{s,k+1})$
 8  encrypt $x_{s,1} \dots x_{s,k}$ with the public key of $u_{s,1} \dots u_{s,k}$
    respectively to obtain $E_{u_{s,1}}(x_{s,1}) \dots E_{u_{s,k}}(x_{s,k})$
    ▷ generate zero-knowledge proofs
 9  $\beta_s \leftarrow (E_s(x_{s,1}) \times \dots \times E_s(x_{s,k+1}))$ mod $\mathbb{n}_s^2$
10  generate $\text{setMembershipZKP}(\beta_s, [(h_s \times M), (h_s \times M) + F])$
11  for $i \leftarrow 1$ to $k$
12      do generate $\text{plaintextEqualityZKP}(E_s(x_{s,i}), E_{u_{s,i}}(x_{s,i}))$
    ▷ send the encrypted shares and the proofs to $q$
13  $\overrightarrow{\mathcal{I}_s} \leftarrow \langle \mathbb{U}_s, E_s(x_{s,1}), \dots, E_s(x_{s,k+1}), E_{u_{s,1}}(x_{s,1}), \dots, E_{u_{s,k}}(x_{s,k}),$
      $h_s, \text{setMembershipZKP}(\beta_s, [(h_s \times M), (h_s \times M) + F]),$
      $\text{plaintextEqualityZKP}(E_s(x_{s,1}), E_{u_{s,1}}(x_{s,1})), \dots,$
      $\text{plaintextEqualityZKP}(E_s(x_{s,k}), E_{u_{s,k}}(x_{s,k})) \rangle$
14  send tuple $(\text{SHARES}, q, t, \tau, \overrightarrow{\mathcal{I}_s})$ to $q$
```

Figure 4.1: Protocol: Malicious-k-shares

4.4.4 Security Analysis – Correctness

In the protocol Malicious-k-shares (Figure 4.1), agent q either learns the correct reputation of agent t in the context ψ, or learns the identity of a malicious agent who has disrupted the protocol, under the malicious adversarial model.

In this section we analyze correctness in the context of the messages sent by the source agents under the malicious adversarial model. Correctness under the semi-honest model is analyzed in detail in the extended technical report [137].

Each agent $s \in \mathbb{S}_t$ communicates exclusively with agent q. If an agent s takes any of the actions 1 to 3 (Section 4.2.2), it would be exposed as malicious to agent q. *Note:* Agent q can then remove the malicious agent from the set of source agents and restart the protocol. Eventually, only those agents who do not take actions 1 to 3 will remain in the set of source agents.

Protocol: Malicious-k-shares (contd.)

agent q receives the tuple $(\text{SHARES}, q, t, \tau, \overrightarrow{\mathcal{I}_s})$ **from an agent** $s \in \mathbb{S}_t$

```
    ▷ verify the set membership proof
1   $\beta_s \leftarrow (E_s(x_{s,1}) \times \ldots \times E_s(x_{s,k+1}))$ mod $\mathfrak{n}_s^2$
2   verify setMembershipZKP($\beta_s$, $[(h_s \times M), (h_s \times M) + F]$)
    ▷ verify the plaintext equality proofs
3   for $i \leftarrow 1$ to $k$
4       do verify plaintextEqualityZKP($E_s(x_{s,i}), E_{u_{s,i}}(x_{s,i})$)
    ▷ manage the sets of encrypted shares to be relayed
5   for $i \leftarrow 1$ to $k$
6       do $\mathbb{V}_{u_{s,i}} \leftarrow \mathbb{V}_{u_{s,i}} \cup \{E_{u_{s,i}}(x_{s,i})\}$
    ▷ subtract $s$ from the set of agents who are yet to send their shares
7   $\mathbb{S}'_t \leftarrow \mathbb{S}'_t - \{s\}$
    ▷ if shares have been received from all source agents then relay the shares
8   if $\mathbb{S}'_t = \phi$
9       then $\mathbb{S}'_t \leftarrow \mathbb{S}_t$ ▷ initialize the set of agents who are yet to send their sum
10          send tuple $(\text{VERIFIED\_SHARES}, q, t, \tau, \mathbb{V}_u)$ to each agent $u \in \mathbb{S}_t$
```

an agent $s \in \mathbb{S}_t$ receives the tuple $(\text{VERIFIED\_SHARES}, q, t, \tau, \mathbb{V}_s)$ **from agent q**

```
    ▷ compute sum of the shares
1   $\gamma_s \leftarrow ((\prod_{c \in \mathbb{V}_s} c) \times E_s(x_{s,k+1}))$ mod $\mathfrak{n}_s^2$
2   $\sigma_s \leftarrow D_s(\gamma_s)$
    ▷ encrypt the sum
3   encrypt $\sigma_s$ with the public key of $q$ to obtain $E_q(\sigma_s)$
    ▷ generate zero-knowledge proof
4   generate plaintextEqualityZKP($\gamma_s, E_q(\sigma_s)$)
    ▷ send the encrypted sum and the proof to $q$
5   send tuple $(\text{AGGREGATE}, q, t, \tau, E_q(\sigma_s), pezkp(\gamma_s, E_q(\sigma_s)))$ to $q$
```

agent q receives the tuple $(\text{AGGREGATE}, q, t, \tau, E_q(\sigma_s), pezkp(\gamma_s, E_q(\sigma_s)))$ **from an agent** $s \in \mathbb{S}_t$

```
    ▷ verify the proof
1   $\gamma_s \leftarrow ((\prod_{c \in \mathbb{V}_s} c) \times E_s(x_{s,k+1}))$ mod $\mathfrak{n}_s^2$
2   verify plaintextEqualityZKP($\gamma_s, E_q(\sigma_s)$)
    ▷ decrypt the sum
3   $\sigma_s \leftarrow D_q(E_q(\sigma_s))$
    ▷ compute intermediate sum for reputation
4   $\theta \leftarrow \theta + \sigma_s$
    ▷ subtract $s$ from the set of agents who are yet to send their sum
5   $\mathbb{S}'_t \leftarrow \mathbb{S}'_t - \{s\}$
    ▷ if sum has been received from all source agents, compute reputation
6   if $\mathbb{S}'_t = \phi$
7       then $r_{t,\psi} \leftarrow (\theta \; mod \; M)/n$
```

Figure 4.2: Protocol: Malicious-k-shares (contd.)

An agent $s \in \mathbb{S}_t$ is unable to tamper with the communication channels since we assume that all communication takes place over authenticated point-to-point channels that are resistant to tampering. Since each agent $s \in \mathbb{S}_t$ communicates exclusively with agent q, it will be exposed as malicious if it does not conform to these requirements.

Wiretapping the communication channels has no effect on the correctness of the protocol.

The first tuple of information that an agent $s \in \mathbb{S}_t$ provides to agent q is: $(\text{SHARES}, q, t, \tau, \overrightarrow{\mathcal{I}_s})$, where $\overrightarrow{\mathcal{I}_s} = \langle\, \mathbb{U}_s$, $E_s(x_{s,1})$, …, $E_s(x_{s,k+1})$, $E_{u_{s,1}}(x_{s,1})$, …, $E_{u_{s,k}}(x_{s,k})$, setMembershipZKP(β_s, F), plaintextEqualityZKP$(E_s(x_{s,1}), E_{u_{s,1}}(x_{s,1}))$, …, plaintextEqualityZKP$(E_s(x_{s,k}), E_{u_{s,k}}(x_{s,k}))\, \rangle$.

The correctness of the first four elements of the tuple and the set $\mathbb{U}_s$ can be trivially verified by agent q. The remaining information pertains to the shares prepared by agent s. The shares have been prepared correctly if the following conditions hold true: **(1)** the shares add up to a value in $[(h \times M), (h \times M) + F]$; **(2)** $E_{u_{s,1}}(x_{s,1})$, …, $E_{u_{s,k}}(x_{s,k})$ encrypt the same shares as $E_s(x_{s,1})$, …, $E_s(x_{s,k})$ respectively; **(3)** $E_{u_{s,1}}(x_{s,1})$, …, $E_{u_{s,k}}(x_{s,k})$ are encrypted with the public keys of agents $u_{s,1} \ldots u_{s,k}$ respectively.

The first condition holds true for an agent s if the verification of setMembershipZKP$(\beta_s, [(h_s \times M), (h_s \times M) + F])$ by agent q is successful. Agent q can verify the proof since it can independently compute β_s (due to the additive homomorphic property of the cryptosystem), F and M are publicly known, and h_s is provided by agent s. An incorrect value of h_s will result in failure of the verification of the zero-knowledge proof. A zero-knowledge proof that shows membership in an interval with an incorrect h_s has no effect on the final output of the protocol since it is computed as $mod\ M$.

The second and third conditions hold true for an agent s if the verification of each plaintextEqualityZKP$(E_s(x_{s,i}), E_{u_{s,i}}(x_{s,i}))$ by agent q is successful, where $i \in \{1 \ldots k\}$. Agent q can verify these proofs since it can independently retrieve the public keys of agents s and $u_{s,1} \ldots u_{s,k}$ from a certificate authority.

If the verification of the one set-membership zero-knowledge proof and the k plaintext-equality zero-knowledge proofs provided by an agent s succeeds, it implies that agent s has provided correct information pertaining to the shares that it prepared. Otherwise, agent s can be considered as malicious.

The second tuple of information that an agent $s \in \mathbb{S}_t$ provides agent q is: $(\text{AGGREGATE}, q, t, \tau, E_q(\sigma_s), \mathit{pezkp}(\gamma_s, E_q(\sigma_s)))$.

The correctness of the first four elements of the tuple can be trivially verified by agent q. The remaining information pertains to the sum σ_s. The sum has been computed correctly if the following condition holds true: γ_s and $E_q(\sigma_s)$ encrypt the same plaintext.

The condition holds true for an agent s if the verification of $\mathit{pezkp}(\gamma_s, E_q(\sigma_s))$ by agent q is successful. Agent q can verify the proof since it can independently compute γ_s (due to the additive homomorphic property of the cryptosystem) and it can independently retrieve the public key of agent s from a certificate authority.

4.4.5 Security Analysis – Privacy

The probability that the protocol will not preserve agent s's privacy can be stated as: $P(\neg \mathit{perform}(s, u_{s,1}, \rho)) \times \ldots \times P(\neg \mathit{perform}(s, u_{s,k}, \rho))$. We assume that the agents $u_{s,1} \ldots u_{s,k}$ are selected such that this probability is low. Therefore, with high probability, the adversary learns no more information about f_{st} than it can learn in the ideal protocol with what it knows before the execution of the protocol and the outcome.

The protocol Malicious-k-shares is a real privacy preserving reputation protocol (Definition 7) under the malicious model, because: (1) Malicious-k-shares has the same parameters as the ideal protocol (except the TTP), and (2) the adversary does not learn any more information under the malicious adversarial model about the private input of any agent s in Malicious-k-shares than it can learn in the ideal protocol, with high probability: $1 - (P(\neg \mathit{perform}(s, u_{s,1}, \rho)) \times \ldots \times P(\neg \mathit{perform}(s, u_{s,k}, \rho)))$.

In this section we analyze privacy only in the context of attack 6 in which an agent provides illegal information (Section 4.2.2) under the malicious adversarial model. Privacy under the semi-honest model and under the other attacks of the malicious model is analyzed in detail in the extended technical report [137].

If a source agent $u \in \mathbb{S}_t$ provides illegal information, it has no effect on the condition that all first k shares of agent s must be known to breach agent s's privacy. Agent u provides no information to agent s or agent q that would result in agent s divulging any extra information.

Agent t may provide an illegal $\mathbb{S}_t$, however, that has no effect on the protocol since q also retrieves and verifies $\mathbb{S}_t$ from agent t's source managers.

Agent q sends two types of messages to source agents: PREP, and VERIFIED_SHARES.

PREP: Agent q may create $\mathbb{S}_t$ itself in order to attack an agent $s \in \mathbb{S}_t$. The set may be created such that it contains all dishonest agents except agent s who is under attack. However, we assume that $P(\neg \mathit{perform}(s, u_{s,1}, \rho)) \times \ldots \times P(\neg \mathit{perform}(s, u_{s,k}, \rho))$ is low. That is, there exist trustworthy agents in the protocol such that agent s receives a high enough privacy guarantee.

VERIFIED_SHARES: Agent q may substitute the shares sent by other agents to an agent s with shares that it has created itself. Agent q may also not relay a share at all. In both these cases, the best outcome for q would be to learn agent s's $(k+1)$'th share. This has no effect on the privacy of agent s since agent q is still unable to learn its first k shares. Each of those shares is encrypted and can only be decrypted by its destination agent.

The protocol may be extended such that an agent s is allowed to abstain if the privacy guarantee is not sufficient. The extension would be as follows: The agent who wishes to abstain would generate the shares such that their sum equals zero. The abstaining agent would inform the querying agent that it has abstained, and would prove that the sum of the shares equals zero.

An Attack on the Ideal Protocol

We describe an attack in which the adversary attempts to determine the private feedback of a source agent over the course of two reputation queries. Consider the scenario when a new agent s is added to the set of source agents $\mathbb{S}_t$. Let $\mathbb{S}_t^{'} = \mathbb{S}_t \cup \{s\}$. Let the reputation of the target agent t be r_t and $r_t^{'}$ for the set of source agents $\mathbb{S}_t$ and $\mathbb{S}_t^{'}$ respectively. A querying agent q that queries the reputation of the target agent t with both sets of source agents can compute the private feedback of agent s as $f_{st} = (r_t^{'} \times (n+1)) - (r_t \times n)$, where $n = |\mathbb{S}_t|$. A similar attack can also determine the private feedback of an existing source agent that drops out from the set of source agents.

The ideal protocol (Definition 6) is vulnerable to this attack. Consequently, the Malicious-k-shares protocol is also vulnerable as it emulates the ideal protocol. We note that this is a general issue for all protocols (including Pavlov et al. [235] and Gudes et al. [128]) that can produce the sum of the feedback values of the following sets of source agents: $\mathbb{S}_t$, and $\mathbb{S}_t \cup \{s\}$ or $\mathbb{S}_t - \{u\}$, where $s \notin \mathbb{S}_t$ and $u \in \mathbb{S}_t$.

We have previously described this attack in our paper [134] on the non-cryptographic k-shares decentralized privacy preserving reputation protocol for the semi-honest model. Additionally, we have discussed three possible defenses to counter this attack in our previous work. These solutions can also be applied to the protocol described in the current work. We briefly describe the three solutions below and refer the reader to our previous work for further details.

1. Each source agent retrieves a random number from a trusted random number generator and adds it to its feedback. The sum of the random numbers perturbs the reputation score with a different value each time the reputation is calculated.

2. Two or more source agents who trust each other in the context of preserving privacy form a trusted subset. The agents in a trusted subset submit feedback in tandem, that is, they either all submit their feedback or none of them does. Thus, at best, an attacker can only learn the cumulative sum of multiple feedback values instead of an individual feedback value.

3. A source agent probabilistically decides whether to participate in the protocol or to abstain from the protocol. The effect achieved is that the attacker can no longer deterministically create the set of agents who will submit their feedback about a given target agent.

4.4.6 Complexity Analysis

Table 4.1: Protocol Malicious-k-shares – Complexity.

Tuple	**Occurrences**	**IDs**	**Ciphertexts**	**SMZKPs**	**PEZKPs**
REQUEST_FOR_SOURCES	1				
SOURCES	1	n			
PREP	n	$n \times n = n^2$			
SHARES	n	kn	kn	n	kn
VERIFIED_SHARES	n		kn		
AGGREGATE	n		n		n
Total	$4n + 2$	$n + n^2 + kn$	$2kn + n$	n	$kn + n$
Complexity	$O(n)$	$O(n^2)$	$O(kn)$	$O(n)$	$O(kn)$

Table 4.1 presents an analysis of the complexity of the Malicious-k-shares protocol. The column “Occurrences” analyzes the number of messages exchanged. Whereas, the columns “IDs”, “Ciphertexts”, “SMZKPs”, and “PEZKPs” analyze the bandwidth usage of the protocol.

The protocol requires $O(n)$ messages to be exchanged. The protocol also performs a DHT lookup in the initiation phase, which requires an additional $O(log\ N)$ messages (assuming Chord). Thus, the total number of messages exchanged is $O(n) + O(log\ N) = O(n + log\ N)$, where n is the number of source agents in the protocol and N is the total number of agents in the system respectively.

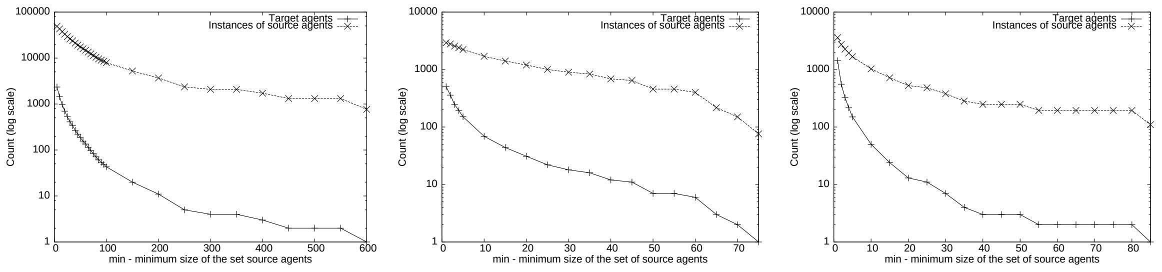
In terms of bandwidth usage, the protocol requires transmission of the following amount of information: $O(n^2)$ agent IDs, $O(kn)$ ciphertexts, $O(n)$ non-interactive zero-knowledge proofs of set membership, $O(kn)$ non-interactive zero-knowledge proofs of plaintext equality, and an additional $O(log\ N)$ messages of constant size for the DHT lookup.

The size of the IDs, the ciphertexts, and the PEZKPs is constant. Moreover, the size of the SMZKPs is also constant, given that $p = |\mathbb{F}|$ is constant. Thus, the complexity of the protocol in terms of bytes of information transferred can be stated as $O(n^2) + O(kn) + O(n) + O(kn) + O(log\ N) = O(n^2 + log\ N)$. We observe in Section 4.5.2 that $k \ll n$.

Moreover, k can be considered as a constant in the protocol. k can be set as a system-wide constant. Alternatively, in the extended version of the protocol where agents can abstain, the querying agent can be given the choice to set k. The trade-off between a lower and a higher value of k is possible lower participation from the source agents and higher bandwidth usage respectively.

4.5 Experimental Evaluation

We conduct experiments to evaluate the following two hypotheses that the Malicious-k-shares protocol is based on:



(a) Advogato (b) Squeak (c) Robots

Figure 4.3: Distribution of the potential target agents and the instances of source agents

1. A source agent can preserve its privacy by trusting on only k fellow source agents, where k is much smaller than $n-1$, the size of the set of all fellow source agents.
2. Accurate reputation values can be computed even if the source agents whose privacy can not be preserved abstain and thus do not provide their feedback values.

The first hypothesis places emphasis on the notion that a source agent can find k sufficiently trustworthy agents that enable it to preserve its privacy, even if k is much smaller than $n-1$. The fact that a source agent is able to preserve its privacy with k trustworthy agents, where $k \ll n-1$, has already been validated in Section 4.4.5.

4.5.1 Datasets

A trust graph can be defined as a weighted directed graph $G = (\mathbb{A}, \mathbb{T}, \mathbb{F})$, in which the set of vertices corresponds to the set of agents $\mathbb{A}$, the set of edges corresponds to the set of binary trust relationships $\mathbb{T}$, and the set of weights of the edges is given as a set of feedback values $\mathbb{F}$.

We use three real trust graphs as the datasets for our experiments. These three trust graphs have been independently evolved by the communities of `advogato.org`, `squeak.org`, and `robots.net`. The members of each of these communities rate each other in the context of being active and responsible members of the community. A common element between the three sites is that they use the same reputation system and thus offer the same set of feedback values. The choice of feedback values are *master*, *journeyer*, *apprentice*, and *observer*, with *master* being the highest level in that order. The trust graphs were obtained from the site `trustlet.org` on May 30, 2012.

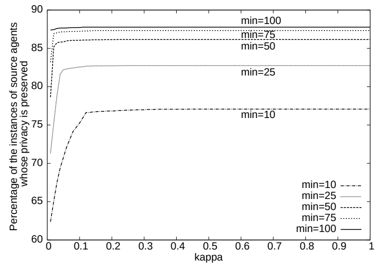
Table 4.2 lists the number of users, the number of ratings, and the distribution of the ratings in each of the three trust graphs. Figure 4.3 shows the distribution of the potential target agents in each trust graph according to the minimum size of the set of their source agents. The graphs in Figure 4.3 also plot the instances of source agents in the trust graphs.

Table 4.2: Trust Graphs.

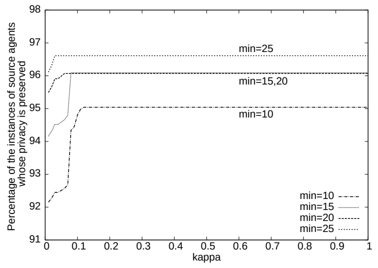
	Advogato	**Squeak**	**Robots**
No. of users	14,020	766	16,620
No. of ratings	56,652	2,928	3,593
Ratings / user	4.04	3.82	0.22
master ratings	31.9%	31.8%	35.4%
journeyer ratings	40.0%	32.0%	26.0%
apprentice ratings	18.7%	33.2%	35.2%
observer ratings	9.4%	3.0%	3.4%

The members of the communities are expected to not post spam, not attack the reputation system, etc. Thus, we consider that the context "be a responsible member of the community" comprises of the context "be honest". Since we quantify trust as probability, we heuristically substitute the four feedback values of the trust graphs as follows: *master* $= 0.99$, *journeyer* $= 0.70$, *apprentice* $= 0.40$, and *observer* $= 0.10$.

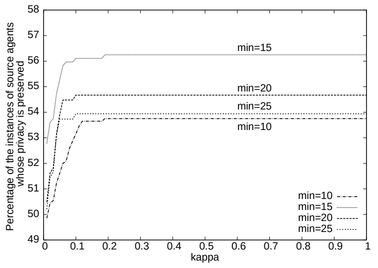
For the experiments, we define the lowest acceptable probability that privacy will be preserved as 0.90. This implies that a set of two trustworthy agents must include either one *master* rated agent or two *journeyer* rated agents for this threshold to be satisfied.



(a) Advogato

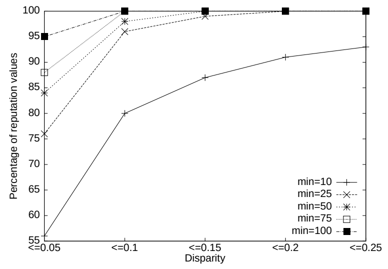


(b) Squeak

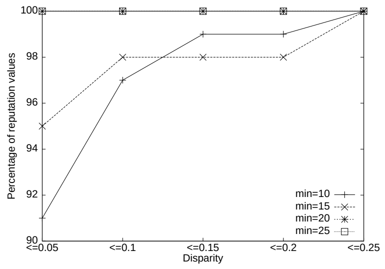


(c) Robots

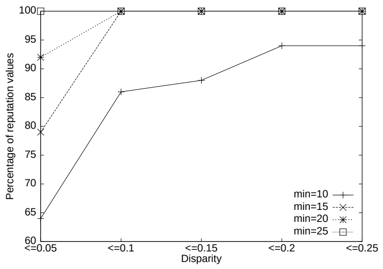
Figure 4.4: Effect of increasing κ on the percentage of instances of source agents whose privacy is preserved



(a) Advogato



(b) Squeak



(c) Robots

Figure 4.5: Disparity

4.5.2 Experiment 1

Objective

Observe the effect of increasing the value of k on the percentage of the instances of source agents whose privacy is preserved.

Setup

The maximum number of fellow source agents that an agent can trust on is $n-1$. A fraction of the size of this set can be stated as $\kappa \times (n-1)$, where $\kappa \in [0,1]$. For our experiments, we equate $k = \lceil \kappa \times (n-1) \rceil$. This allows us to use κ (kappa) to vary the value of k as a fraction of $n-1$.

We query the reputation of all agents with at least *min* source agents. We vary κ from 0.01 to 1 with an increment of 0.01 and observe the percentage of the instances of source agents whose privacy is preserved. The set of experiments is run with $min \in \{10, 25, 50, 75, 100\}$ for the Advogato trust graph and with $min \in \{10, 15, 20, 25\}$ for the Squeak and Robots trust graphs. As discussed in Section 4.4.5, the privacy of a source agent s is preserved if $P(\neg perform(s, u_{s,1}, \rho)) \times \ldots \times P(\neg perform(s, u_{s,k}, \rho))$ is low, which is less than or equal to 0.1 in our case. $u_{s,1} \ldots u_{s,k}$ are the agents that agent s trusts.

Analysis

In the results of the experiment on the Advogato trust graph (Figure 4.4a), we observe that for $min = 25$, the privacy of 71% of the instances of source agents is preserved when $\kappa = 0.01$. That is, 71% of the source agents find sufficiently trustworthy agents among only 1% of their fellow source agents in order to preserve their privacy. The percentage is 82% at $\kappa = 0.04$ at which stage the function nearly converges and there is no significant improvement in the percentage by increasing κ any further. Convergence is reached at $\kappa = 0.03$ for the functions of $min = 50$ and above. Even for $min = 10$, convergence is reached at the fairly low value of $\kappa = 0.12$. It is thus evident that in the Advogato trust graph, a source agent can preserve its privacy by trusting on only k fellow source agents, where k is much smaller than $n-1$, the size of the set of all fellow source agents. Raising k over a certain threshold offers no advantage. This conclusion is also supported by the results of the experiments on the Squeak (Figure 4.4b) and Robots (Figure 4.4c) trust graphs. The Robots trust graph is quite sparse as compared to the other two graphs. It has an average user to ratings ratio of only 0.22 compared to the Advogato and Squeak trust graphs that have a ratio of

4.04 and 3.82 respectively. Yet the percentage of the instances of source agents whose privacy is preserved converge very early in the Robots trust graph as well.

The privacy of a source agent s is preserved if $P(\neg perform(s, u_{s,1}, \rho)) \times \ldots \times P(\neg perform(s, u_{s,k}, \rho))$ is low, that is, if the probability that all the k fellow source agents that s considers trustworthy will turn out to be dishonest is low. Thus, whether the privacy of a source agent s will be preserved depends on whether s can find k sufficiently trustworthy agents among fellow source agents. However, in certain cases, an agent s is unable to find k sufficiently trustworthy agents even if $kappa = 1$ (that is, $k = n - 1$) and even if there are more than 100 source agents in the protocol. This is because the agent s either does not have trust relationships toward its fellow source agents or the relationships are not strong enough. The functions in Figure 4.4 do not converge to 100% due to the existence of such agents whose privacy cannot be preserved no matter how large is k or how high is min. However, we observe in Section 4.5.3 that such agents are in the minority. A high percentage of the agents are able to find sufficiently trustworthy agents among their fellow source agents in order to preserve their privacy in real trust graphs. We note that protocols in the literature such as Pavlov et al. and Gudes et al. that rely on all $n - 1$ fellow source agents of an agent to preserve its privacy will also fail to protect the privacy of the agents who cannot preserve their privacy in our protocol. This is because all $n - 1$ fellow source agents are not trustworthy enough. On the contrary, our protocol can be extended to allow agents to abstain from submitting feedback thus protecting their privacy.

4.5.3 Experiment 2

Objective

Observe the accuracy of the reputation values computed when source agents whose privacy can not be preserved abstain and thus do not provide their feedback values.

Setup

Let $\mathbb{B}$ be the set of source agents that abstain and thus do not provide their feedback values, where $\mathbb{B} \subset \mathbb{S}_t$ and $\mathbb{S}_t$ is the set of all source agents of the target agent t. Let r_t be the reputation computed using feedback from all source agents in $\mathbb{S}_t$ and let r'_t be the reputation computed using feedback from only the agents who do not abstain, that is, the agents in the set $\mathbb{S}_t - \mathbb{B}$.

We define the disparity of a reputation value as $|r_t - r'_t|$. That is, the absolute difference between the reputation computed with all source agents and the reputation computed with only the source agents in $\mathbb{S}_t - \mathbb{B}$. The disparity ranges from 0 to 1. The lower the disparity, the more accurate is the reputation. A disparity of 0 means that a reputation value computed with less than all source agents is exactly the same as it would be if computed with all source agents.

We compute the reputation of all target agents with at least min source agents twice. Firstly, with all source agents submitting their feedback. Secondly, with only those source agents submitting feedback whose privacy can be preserved. We then compute the disparity between the two values of reputation for each target agent. We count the number of instances of reputation values where disparity is less than the values in $\{0.05, 0.1, 0.15, 0.20, 0.25\}$ respectively.

Analysis

In the results of the experiment on the Advogato trust graph (Figure 4.5a), we observe that for $min = 25$, the disparity of over 76% of reputation values is less than or equal to the low value of 0.05. Over 96% of reputation values have a disparity of less than or equal to just 0.1. For $min = 75$ and above, the disparity of 100% of the instances of reputation values is less than or equal to the fairly low value of 0.15. Thus, it is evident that even if source agents whose privacy can not be preserved abstain, the reputation of a high percentage of target agents can still be calculated with high accuracy as the mean of the feedback values. This inference is supported by the results of the experiments on the Squeak (Figure 4.5b) and Robots (Figure 4.5c) trust graphs. For $min = 25$, 100% of all reputation values have a disparity of less than or equal to 0.05 in both the Squeak and the Robots trust graphs.

4.6 Related Work

Pavlov et al. [235] propose a decentralized privacy preserving reputation protocol for the malicious adversarial model. The protocol comprises of two steps: (1) The first step is the execution of a witness (feedback provider) selection scheme, which guarantees the inclusion of a certain number of honest witnesses as participants. (2) The second step is the decentralized computation of the reputation as the sum of the feedback values. According to our analysis, the protocol requires an exchange of $O(n^3+N)$ messages and at least $O(n^3+N)$ bytes of information, where n is the number of witnesses selected and N is the number of all potential witnesses. The witness selection scheme requires $O(N)$ messages. The decentralized computation of the reputation uses Pederson Verifiable Secret Sharing. Each witness sends messages to every other $n-1$ participating witnesses. The result is an additional exchange of $O(n^3)$ messages. In contrast, our protocol requires an exchange of only $O(n + log\ N)$ messages and $O(n^2 + log\ N)$ bytes of information. In addition to innovative cryptographic constructions, an important reason for the lower complexity is that each agent in our protocol selects k fellow agents based on their trustworthiness, where $k \ll n$. This eliminates the need for a costly witness selection scheme as well as the need for each agent to exchange messages with $n-1$ fellow agents. Another key difference is that our protocol allows entities to quantify and to minimize the risk to their privacy before feedback is submitted. We use three real and large trust graphs to demonstrate that a high majority of agents can find k sufficiently trustworthy agents in a set of $n-1$ fellow feedback providers even with k as very small compared to $n-1$.

Gudes et al. [128] present a protocol for the malicious adversarial model that augments their Knots reputation system [115] with privacy preserving features. The Knots reputation system is a personalized reputation system, which implies that feedback is collected only from the entities whom the querying entity trusts. The protocol by Gudes et al. eliminates the need for a witness selection scheme but still requires $O(n^3)$ messages for the decentralized computation of reputation.

Further related work and comparison is discussed in the full article [139] that this chapter is based on.

4.7 Conclusion

In this work, we have presented a privacy preserving reputation protocol for the malicious adversarial model. The protocol counters attacks by malicious agents such as submitting illegal feedback values or making erroneous computations. The characteristics that differentiate the protocol from other protocols in the literature include: (1) full decentralization, (2) no need for trusted third parties and specialized platforms, (3) low communication complexity.

Our experiments on three real and large trust graphs demonstrate the validity of the two hypotheses that the Malicious-k-shares protocol is based on: (1) A source agent can preserve its privacy by trusting on only k fellow source agents, where k is much smaller than $n-1$, the size of the set of all fellow source agents. (2) Accurate reputation values can be computed even if the source agents whose privacy can not be preserved abstain and thus do not provide their feedback values.

Chapter 5

A Privacy-Preserving and Reputation-Aware Mobile Participatory Sensing System

The integration of privacy into reputation systems is a crucial need for building secure and reliable participatory sensing applications. Participants are given the assurance that their privacy is preserved even if they contribute some personal sensitive data. In addition, reputation systems allow an application server to monitor participants' behaviors and evict those who provide the system with corrupted data. However, this integration requires achieving seemingly conflicting objectives. Reputation systems monitor participants behaviors along subsequent interactions. Whereas, one of the major objectives of privacy preserving systems is to unlink subsequent interactions. In this work, we define a new attack (RR attack), which exploits this conflict in order to detect the succession of contributions provided by the same participant and to subsequently re-identify their original identity. We show that using this attack, more than 35% of contributions can be associated to their successive contributions in each campaign. We then propose PrivaSense as a new privacy-preserving reputation system that integrates both reputation and privacy such that their objectives are simultaneously achieved. Experiments are conducted using a real data-set. The results show that PrivaSense decreases the number of contributions linked to their original providers by up to 80%.

We note that this work addresses a centralized scenario where an application server plays a central and trusted role. However, our proposed solution attempts to remove the requirement of trusting a single entity for all tasks. The solution introduces additional entities and distributes tasks over those entities and the application server such that each of them is only partially trusted.

This chapter is an adapted version of the article: "PrivaSense: Privacy-Preserving and Reputation-Aware Mobile Participatory Sensing." H. Mousa, S. B. Mokhtar, O. Hasan, L. Brunie, O. Younes, and M. Hadhoud. *In Proceedings of the 14th International Conference on Mobile and Ubiquitous Systems.* November 2017. Pp. 38-47. This work was carried out in the context of the Ph.D. of H. Mousa, co-supervised with L. Brunie and S. B. Mokhtar.

5.1 Introduction

The advancement and widespread use of mobile computing smart devices have helped towards the emergence of a new kind of application called participatory sensing [64]. These applications exploit both the mobility of participants and the sensing capabilities of their devices to construct mobile sensor networks [180] with much less cost and effort compared with traditional Wireless Sensor Networks (WSNs). During the last decade, several participatory sensing applications have been widely used to serve in different areas including health, commerce, etc [170]. Researchers have studied numerous challenges that should be addressed to build reliable and secure participatory sensing systems [78, 164, 219]. These challenges include the assurance of participants' privacy and management of data reliability, which we discuss below.

On the one hand, different applications collect different types of sensed data (e.g. spatial, temporal, images, pollution, sound samples, accelerometer, biometric, barometric, etc) [78]. These data can be exploited to leak participants' privacy through accurately re-identifying their identity, their location at some given time, with whom they were, their movements (e.g. walking, running, sitting down, etc.) [164]. Subsequently, participants can be physically traced and hacked or robbed based on these data [164]. In [59, 91], De Montjoye et al. and Boutet et al. showed that 95% of original identities are re-identified through sharing four contributions including time and location data. Different techniques were proposed to assure the privacy of participants [78]. The main objective of those techniques is to detach the link between each contribution and its provider as well as among multiple contributions provided by the same provider (i.e participant).

On the other hand, these applications are vulnerable to malicious participants who disrupt the system by contributing fabricated or corrupted contributions which affect data reliability and accuracy. To enhance data reliability, application severs adopt reputation systems to trace participants' behaviors along subsequent contributions in order to estimate their honesty and to evaluate the quality of their contributions. In [219], we have extensively studied, analyzed and compared reputation systems that were proposed in this context. From which, it is evident that existing reputation systems manage the link between successive contributions of participants and their real identities to evaluate their behaviors and to evict malicious ones from the campaign. This objective contradicts with the objective of privacy preserving systems mentioned before. That is, managing the linkage among successive contributions leads to privacy leakage. This conflict is referred to as the linkability problem. As a simple example of linkability problem, consider a participant p_i who has a reputation score x in some campaign. When a new campaign starts, this participant shares his new contribution tagged with his current reputation x. The application server evaluates the contribution according to a reputation system and assigns a feedback f to p_i's contribution. x is updated to $x + f$. Consequently, in an upcoming campaign, it is evident that the contribution tagged with reputation $x + f$ has been forwarded from the same identity with reputation x in the previous campaign even if the identity and the data are anonymized. That is, both contributions and their associated pseudonyms are linked according to the reputation account. That is why, monitoring reputation scores for a sequence of contributions clearly leads to the profiling of participants' contributions and subsequently re-identifying their identities.

In the context of participatory sensing, both the challenges of privacy preservation and reputation management have been individually studied in the literature (e.g. [79] and [219]). However, the integration of both these systems in this context is still in its infancy. Existing privacy-preserving reputation systems in participatory sensing do not have the ability to fulfill the objectives of both privacy and reputation systems simultaneously (e.g. [212] [283]). Indeed, such systems either allow participants to launch other attacks (e.g Sybil, or report flooding) which affect the data reliability (e.g. [283]). Other systems allow to profile participants subsequent contributions, which leads to participant re-identification (e.g. [212]).

5.1.1 Contributions

Our contributions in this work can be summarized as follows:

1. We define a new attack (RR attack) that aims to link multiple contributions from the same participant, and subsequently re-identify participants' identities.

2. We present a novel *privacy-preserving and reputation-aware mobile participatory sensing system* **PrivaSense**. In this system, each participant is assigned a new pseudonym for each contribution. The application server evaluates a participant's contribution, assigns it a feedback, forwards this feedback to the reputation server who updates the corresponding reputation account and transfers this account to the next pseudonym of the same participant. Reputation scores are anonymized and transferred in the form of anonymous certificates. This allows participants to conserve their reputation scores across multiple interactions while preventing associations between consecutive contributions.

3. We undertake an analysis revealing the robustness of our proposal against the attacks considered in the threat model described in Section 5.3.2.

4. We then present some experiments based on a real-world data-set [238] to measure both the resilience of our system against the RR attack (i.e. privacy issues) added to the effect of the proposed system on the data reliability. The experimental results indicate that our system introduces higher anonymity (i.e. better privacy) with more accurate data aggregation which enhances the system reliability compared with the state-of-the-art.

5.1.2 Outline

The rest of this chapter is organized as follows: Section 5.2 states the previous work and its limitations. We then define the considered threat model in Section 5.3.2. Next, we present our proposed system and discuss its details in Section 5.4. In Section 5.5, we present an analysis of our proposal. Experimental results are presented in Section 5.6. Finally, this chapter is concluded in Section 5.7.

5.2 Related Work

Significant research effort has been directed toward ensuring privacy preservation in participatory sensing applications, as discussed by Christin et al. in [79]. However, such works consider how to anonymize participants' real identities and/or to anonymize their provided data. Some other works consider the problem of reputation management such as [148] and our system presented in [220]. We have surveyed and compared these systems in [219]. Those systems essentially manage the link to participants real identities in order to assess their reputation. However, very few works consider the problem of privacy-preserving reputation systems in mobile participatory sensing applications.

In [148], each participant is assigned different pseudonyms for each time interval and to exchange the assigned reputation between those pseudonyms through a trusted server. Christin et al. in [80] propose a similar scheme which adopts the blind signature scheme to create pseudonyms. Through this system, a malicious participant can create multiple identities (i.e. Sybil attack) such that he can disrupt the system by providing multiple sensing contributions for the same task.

Another privacy-preserving reputation system that assures the participants' anonymity through the group signature technique is presented by Michalas et al. [212]. Although, the system assures the anonymity of participants, it allows some entities to record a profile of participants through subsequent interactions. That is, the participants' privacy is leaked.

The scheme presented by Wang et al. in [283] utilizes the blind signature technique in order to ensure participant's anonymity. In this scheme, malicious participants can create multiple authentic identifiers based on a single blind identity granted for them (i.e. Sybil attack). In addition, they can launch a report flooding attack. Subsequently, they can submit numerous contributions for the same task while the application server can't detect such behavior.

To sum up, existing works either accurately manage participants' reputation and allow for participants' re-identification and privacy leakage or focus on participants' anonymity and allow participants to launch other attacks that disrupt the system and affect data reliability. In this chapter, we attempt to propose a privacy-preserving and reputation-aware system that allows the participatory sensing applications to be more reliable and secure.

5.3 System Model

5.3.1 Participants, Contributions, and Reputation

In a sensing campaign we have mainly two roles: a participant p_i is a member in the participants set P, where $|P| = n_P$ and $i \in \{1, 2, 3, ..., n_P\}$, and an application server noted as *App.Server*. The application server announces some tasks (e.g. T_j) asking for some sensory data. Each participant selects the task that he wishes to participate in and senses the required observation. Participants are interested to preserve their privacy therefore they carry out some local processing that adopts some privacy preserving mechanism on the sensed data. Then, these data are included in a contribution $C_{p_{ij}}$ (which is the contribution from participant p_i of task T_j). This contribution is forwarded to the application server. The application server aggregates all the received contributions from all participants. It adopts a reputation system that evaluates participants' contributions. It also assigns feedback f_{p_i} for each contribution. This feedback represents the trustworthiness of the contribution from the App.Server's point of view. This feedback is used to update the reputation of the contribution provider $\hat{R}_{p_i}$ to a new score R_{p_i}. Then, the data are processed and visualized for the users.

5.3.2 Threat Model

There are a number of attacks that can be launched. These attacks affect both data reliability and participants' privacy.

Reliability oriented attacks:

- **Sybil Attack:** Malicious participants attempt to generate multiple pseudonyms to increase their reputation through cross-recommendations or providing multiple reports for the same tasks which subsequently leads to system disruption.
- **Replay Attack:** Malicious participants attempt to replay an old pseudonym which has a good reputation score. Hence, replay attackers artificially increase their own reputation. This attacker has the ability to inject the system with corrupted data that could be considered valid since it is submitted in association with a pseudonym that has high reputation score.
- **Reporting Falsified Sensor Readings:** Adversaries try to report falsified sensor readings to disrupt the system measurements. They may also provide false data on behalf of others to degrade their reputation.

Privacy oriented attacks:

- **Identity and Data Re-identification (IDR attack):** we have used this term to describe all the attacks that either leverage contribution content or link contributions to attack participants' privacy. It is showed, in [91], that 95% of participants' identities can be re-identified after submitting 4 spatial and temporal observations. Therefore, participants can use pseudonyms instead of their original identities to anonymize their contributions. Subsequently, attackers cannot link multiple contributions to the same identity. However, adversaries try to infer the original identities of the contribution providers based on the content of their contributions [164]. In [59], Boutet et al. demonstrate that 94% of original identities are re-identified when participants share a multi-sensor data-set. For example, participant may share time, location, image, sound, accelerometer data, etc.
- **Reputation-based Re-identification (RR attack):** We introduce the RR attack as a new type of attack. Using this attack, contributions provided from a participant are associated to each other by analyzing the reputation score assigned to their provider.

  An RR attacker applies four consecutive phases (i.e. monitoring, calculating the expected reputation, uniqueness assessment, and profiling). Firstly, through the monitoring phase, an attacker listens to the network. The attackers can listen to the network since the communication channel is not assumed to be secure. The attacker records the message exchange among the different parties in the sensing campaign. For each task T_j, the attacker keeps the following information for each contribution (1) the pseudonyms $RID^j_{p_i}$ of participant p_i ($\forall i \in 1, 2, 3, ..., n_P$), (2) the contents of p_i's contribution including location, time, and sensed data $(x_{T_j}, y_{T_j}), t_{T_j}, data_{T_j})$, (3) and the reputation scores of the pseudonym ($\hat{R}_{p_i}$), (4) the feedback calculated based on the evaluation of p_i's contribution. The feedback is associated to its corresponding pseudonym $(RID^j_{p_i}, f_{RID^j_{p_i}})$. That is, for each pseudonym, an attacker keeps a record for each task containing the following information $(RID^j_{p_i}, (x_{T_j}, y_{T_j}), t_{T_j}, data_{T_j})$ $(\hat{R}_{p_i}, f_{RID^j_{p_i}})$.

  Secondly, through the calculation of the expected reputation, the attacker updates the monitored reputation score $\hat{R}_{p_i}$ according to the feedback $f_{RID^j_{p_i}}$ for the same task to get the expected reputation score noted as ER_{p_i} and appends it to its corresponding record. Hence, the attacker knows in advance the reputation score (R_{p_i}) that is going to accompany the upcoming contribution of the next task T_{j+1}. However, he does not know the new pseudonym that is going to carry this score. That is why a uniqueness assessment step is required.

  Thirdly, through uniqueness assessment, unique reputation scores R_{p_i} monitored at task T_{j+1} are identified. Intuitively, a pseudonym (e.g. $RID^j_{p_i}$) having unique reputation score R_{p_i} is linked to the pseudonym having the same unique value of expected reputation ER_{p_i} calculated at task T_j. If the reputation R_{p_i} score is not unique, this means the update process functions such that multiple participants are assigned the same reputation. In this case, all the pseudonyms carrying the same reputation at T_{j+1} are considered as potential successors.

  Therefore, the pseudonym carrying reputation R_{p_i} is linked to the pseudonym with the same expected reputation ER_{p_i}, if they are unique and they both have the same value. The RR attacker not only links pseudonyms but also contributions from both pseudonyms and records them in a profiling table under the same identity. An example of a profiling table for a set of subsequent tasks is depicted in Table 5.1.

Table 5.1: An example of the profiling table

Task	p_1	p_2	...	p_n
T_1	$((x_{T_1}, y_{T_1}), t_{T_1}, data_{T_1})RID^1_{p_1}$	...	...	$((x_{T_1}, y_{T_1}), t_{T_1}, data_{T_1})RID^1_{p_n}$
T_2	...	...	...	$((x_{T_2}, y_{T_2}), t_{T_2}, data_{T_2})RID^2_{p_n}$
T_3	$((x_{T_3}, y_{T_3}), t_{T_3}, data_{T_3})RID^3_{p_1}$	$((x_{T_3}, y_{T_3}), t_{T_3}, data_{T_3})RID^3_{p_2}$	...	...
T_4	$((x_{T_4}, y_{T_4}), t_{T_4}, data_{T_4})RID^4_{p_1}$	...	...	$((x_{T_4}, y_{T_4}), t_{T_4}, data_{T_4})RID^4_{p_n}$
T_5	...	$((x_{T_5}, y_{T_5}), t_{T_5}, data_{T_5})RID^5_{p_2}$	...	...
....	...	...	...	...
T_N	$((x_{T_N}, y_{T_N}), t_{T_N}, data_{T_N})RID^N_{p_1}$	$((x_{T_N}, y_{T_N}), t_{T_N}, data_{T_N})RID^N_{p_2}$	...	$((x_{T_N}, y_{T_N}), t_{T_N}, data_{T_N})RID^N_{p_n}$

5.4 PrivaSense

We propose PrivaSense, a new privacy-preserving reputation system for participatory sensing applications. PrivaSense takes into account the attacks considered in the threat model. To deal with this threat model, we are going to integrate several techniques:

1. An anonymous authentication technique to keeping the anonymity of participants' identities.
2. A reputation system to enforce the data reliability and to defend against reliability oriented attacks.
3. A technique for anonymizing contributions' content to defend against privacy oriented attacks.
4. A technique that defends against RR attack through managing the conflict arisen as a result of integration between reputation and privacy systems.

The framework of the privacy-preserving and reputation-aware mobile participatory sensing system, proposed in this work, is depicted in Figure 5.1. The proposed framework involves four main parties: a participant p_i, an application server noted as App.Server, an authentication server noted as Auth.Server, and a reputation server referred to as Rep.Server. Participants and App.Server are the standard parties in any participatory sensing campaign. Auth.Server and Rep.Server are additional entities that are involved in order to manage the threat model described earlier.

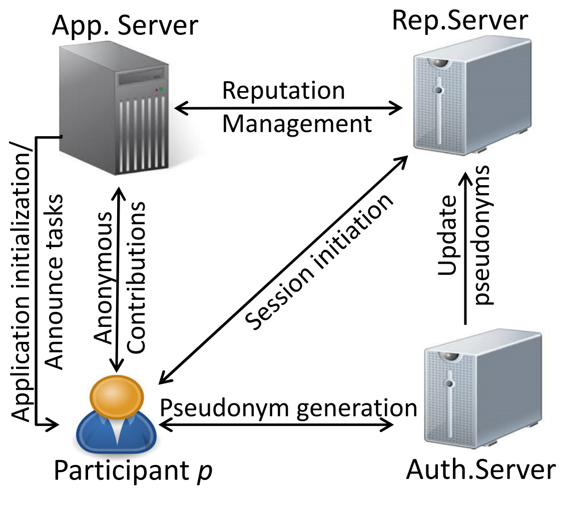


Figure 5.1: PrivaSense Architecture

The assumptions related to each entity are defined as follows:

- **Participant** A participant (1) authenticates himself with Auth.Server, (2) selects a task from the tasks announced by the App.Server, (3) constructs a sensing report (i.e. contribution), (4) forwards it to the App.Server, (5) re-authenticates with Auth.Server for a new task.
- **App.Server** The App.Server (1) initiates the sensing campaign, (2) receives from participants and aggregates sensing reports, (3) assigns a feedback to each contribution, (4) and forwards this feedback to Rep.Server.

- **Auth.Server** is the entity that (1) generates pseudonyms for each participant, (2) Forwards these pseudonyms to Rep.Server, (3) renews these pseudonyms after each campaign, (4) and sends the new pseudonyms to both the participant and to the Rep.server. Auth.Server keeps track of the succession of the participants pseudonyms. These functions allow the Auth.Server to manage the identity issues where it has not got any information about the participant's contributions or about his reputation. Auth.Server is assumed to be honest but curious. It does not collude with other entities.
- **Rep.Server** is the entity that (1) receives session initiation information from the participant, (2) verifies participant's identity and sends anonymous reputation certificates to App.Server, (2) receives feedback from the App.Server, (3) uses the feedback to update reputation scores, (4) receives pseudonym updates from Auth.Server, (5) and links the reputation score of the old pseudonyms to the current one, (6) discards old pseudonyms. Rep.Server manages reputation issues while it has no information about participants' original identities nor contributions' contents. Rep.Server is also assumed to be honest but curious. It is also considered to not collude with other entities.

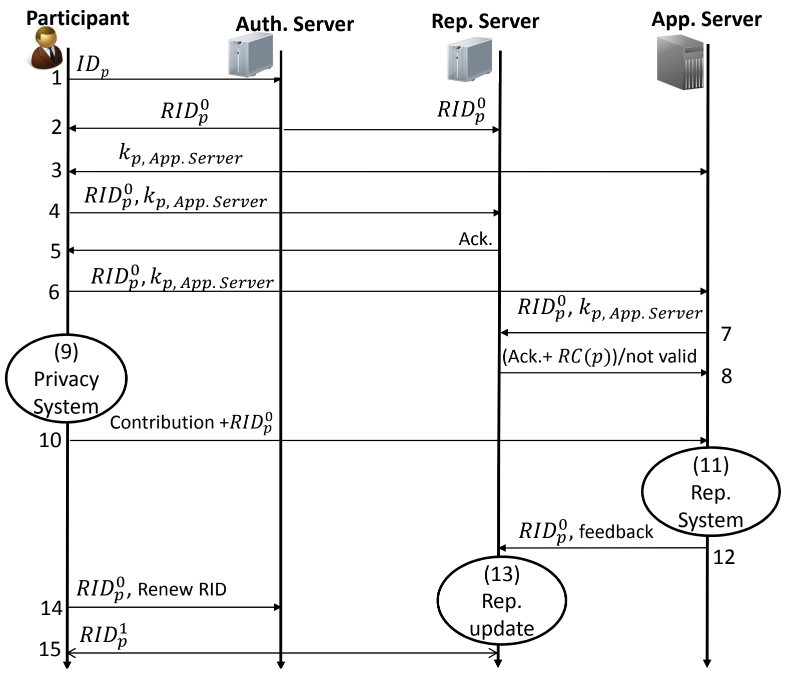


Figure 5.2: PrivaSense: a detailed scenario

5.4.1 Protocol Overview

The complete scenario of PrivaSense is depicted in Figure 5.2. First, participant p_i registers with Auth.Server. The Auth.server creates the first random authentication identifier $RID_{p_i}^0$ as described in [158] (step 1). This $RID_{p_i}^0$ is the first pseudonym granted to participant p_i and it is simultaneously sent to the reputation server (Rep.Server) (step 2). Both p_i and the App.Server communicate to create a session key using an algorithm such as Diffie-Hellman (step 3). The participant p_i then sends this generated session key to the Rep.Server to initialize the session (step 4). Rep.Server records this key and acknowledges p_i (step 5). The acknowledgment indicates that the session has been initiated at the Rep.Server. p_i forwards his first anonymous identity $RID_{p_i}^0$ to App.Server along with the session key (step 6). App.Server sends a query about $RID_{p_i}^0$ to the Rep.Server (step 7). Rep.Server checks if the received identifier matches a user identity. If so, a valid acknowledgment is sent back to the App.Server in line with a reputation certificate of the considered participant. This certificate contains both his current identifier and his current reputation score ($RID_{p_i}^0$, $\hat{R}_{p_i}$). Otherwise, a non-valid acknowledgment is sent (step 8). p_i senses his environment, constructs his first contribution, applies a privacy preserving mechanism on the report contents (e.g. [79]) (step 9). Then, this contribution is forwarded to App.Server (step 10). App.Server accepts the contribution, assesses its trust by applying a reputation management system (e.g. [220]) and calculates a feedback (step 11). This feedback is shared with Rep.Server along with the pseudonym $RID_{p_i}^0$ (step 12), to update the reputation score of the considered participant (step 13). p_i contacts Auth.Server to get a new pseudonym $RID_{p_i}^1$ by submitting his last pseudonym $RID_{p_i}^0$ (step 14). The old and new pseudonyms are simultaneously forwarded for both the participant and for the Rep.Server as well (step 15). Rep.Server updates the pseudonym of each participant. In the following section, the proposed PrivaSense system is explained in detail.

5.4.2 Protocol Description

Our privacy preserving reputation system goes through four consecutive phases, (1) Participant Registration and Authentication, (2) Issuing a Reputation Certificate, (3) Privacy and Reputation Assessment, and (4) Re-Authentication. The details of these phases are described as follows:

Participant Registration and Authentication

Through this phase, participant p_i joins a sensing campaign. First, p_i sends his permanent identifier (e.g. ID_{p_i}) to Auth.Server. The first pseudonym $RID_{p_i}^0$ is then generated by Auth.Server. We assume that pseudonyms are calculated as described in [158] using the public key of Auth.Server (i.e. $k_{Auth.Server.pub}$) as shown in Equation 5.1. Only Auth.Server is able to decrypt it and to reveal the real identity of the participant. This pseudonym is forwarded to both the participant and Rep.Server.

$$RID_p^0 = E(ID_{p_i}, r_{p_i}^0)_{k_{Auth.Server.pub}} \tag{5.1}$$

where $RID_{p_i}^0$ is the pseudonym generated, E is an encryption function of the identity ID_{p_i} and a random variable $r_{p_i}^0$. $k_{Auth.Server.pub}$ is the Auth.Server public key.

A participant then authenticates himself with the App.Server. First, both the participant p_i and App.Server communicate through some key exchange mechanism, (e.g. Diffie-Hellman key exchange mechanism), and generate a session key ($k_{p_i,App.Server}$). The participant sends this key to Rep.Server along with his pseudonym $RID_{p_i}^0$. Rep.Server records the received data and acknowledges the participant. The participant sends his $RID_{p_i}^0$ and the session key previously generated ($k_{p_i,App.Server}$) to App.Server. App.Server sends these data to Rep.Server asking for the validity of $RID_{p_i}^0$ to make sure that the identity is valid and it has already initiated a session through the key exchange mechanism. Rep.Server checks the validity of the received anonymous identity and if the session key matches the one received earlier from this identity. If so, Rep.Server sends a valid acknowledgment and the reputation score of $RID_{p_i}^0$ embedded in a reputation certificate to App.Server as shown in the upcoming phase. Otherwise, a non-valid acknowledgment is sent.

Registration and authentication prevent malicious participants from launching identity related attacks such as a report flooding attack. Participants are asked to authenticate themselves before submitting their contributions. Thus, they can not fabricate multiple authentic identities and subsequently submit multiple contributions for the same task (i.e. report flooding). In addition, they could not report false data on behalf of the others since authentication is required before submitting.

Issuing a Reputation Certificate

RR attackers link participant's contributions to each other based on the relation inherent in the reputation accounts. Therefore, we tend to release the relation between consecutive reputation score of each participant. We adopt a double fold anonymization mechanism. First, we adopt a cloaking mechanism on the reputation scores before their transfer to the App.Server. Then, cloaked reputation scores are embedded within anonymous reputation certificates.

Firstly, to anonymize reputation scores, we cloak them by adding a small random noise. Reputation scores are randomly incremented or decremented by a value noted as $i$ncrement/$d$ecrement $a$mount referred to as ida. This change prevents the App.Server to link two consecutive reputation scores assigned to the same participant. For this, a $r$andom $i$ncrement/$d$ecrement variable noted as rid is generated such that $rid \in \{0, 1\}$. 0 is used to increment, and 1 is to decrement. Then, ida is generated such that it belongs to a specified cloaking interval noted as clk_{intr}, $ida \in [0, clk_{intr}]$. Reputation scores are incremented such that they do not surpass 1 or decremented such that they are not less than 0. The maximum change imposed on the reputation score is $\pm clk_{intr}$. The cloaked reputation is referred to as $\hat{AR}_{p_i}$. The cloaking process is formulated according to Equation 5.2.

$$\hat{AR}_{p_i} = \begin{cases} min\left\{\hat{R}_{p_i} + ida, 1\right\} & \text{if } rid == 0 \\ max\left\{\hat{R}_{p_i} - ida, 0\right\} & \text{if } rid == 1 \end{cases} \tag{5.2}$$

Where $\hat{R}_{p_i}$ is the reputation score of p_i and $\hat{AR}_{p_i}$ is the output cloaked reputation score.

Using large values for clk_{intr} adds more noise to the reputation scores. Intuitively, larger values of the cloaking interval have a better performance from the anonymization point of view. However, this has a negative impact on the reputation and, then on the accuracy of the aggregated data. That is clk_{intr} is a system parameter that manages the trade-off between anonymity and accuracy. Subsequently, we test this parameter to see its effect on the system performance (e.g. anonymity and reliability) in Section 5.6.2.

Through this obfuscation process, the obfuscated reputation score of a participant $\hat{AR}_{p_i}$ reduces the linkability to the original score $\hat{R}_{p_i}$. Since, $\hat{AR}_{p_i}$ does not perfectly equal to $\hat{R}_{p_i}$. An RR attacker is unable to directly observe which participant had which score and which pseudonym previously. This ensures better anonymity and better robustness against the RR attack.

Secondly, the cloaked reputation of a participant is transferred to the App.Server in the form of an anonymous **r**eputation **c**ertificate noted as RC_{p_i}. This certificate is generated such that, it contains both the current pseudonym of the participant, $RID^0_{p_i}$, and its corresponding cloaked reputation score $\hat{AR}_{p_i}$ (See Equation 5.3). Anonymous reputation certificate RC_{p_i} is signed by the Rep.Server's private key. Therefore, reputation certificate cannot be fabricated and a Replay attack mentioned before cannot be launched. Furthermore, adversaries have to access the private key of Rep.Server in order to fabricate a reputation certificate and to have the ability to deceive the App.Server.

$$RC_{p_i} = \left[RID^0_p|\hat{AR}_{p_i}\right]_{k_{Rep.Server.priv}} \tag{5.3}$$

Privacy and Reputation Assessment

Participant p_i senses the required observation and constructs a contribution $C_{p_{ij}}$. We propose that participants apply one of the existing privacy preserving mechanisms summarized by Christin et.al. in [79]. Subsequently, the data provided by a participant are anonymized. These data resist the inference of the original identity of the contribution provider. This ensures the non-associativity property. Therefore, the effect of data based re-identification attack is mitigated [78, 164]. The anonymous contribution is subsequently forwarded to App.Server.

In PrivaSense, App.Sever adopts one of the existing reputation systems such as our system presented in [220]. According to this system, each contribution is evaluated and assigned a feedback which reflects the participant's honesty. Feedback is noted as $f_{RID^0_{p_i}}$. This feedback is forwarded to the Rep.Server in the following form.

$$f_{p_i} = \left[RID^0_{p_i}|\hat{RC}_{p_i}|f_{RID^0_p}\right]_{k_{App.Server.priv}} \tag{5.4}$$

Rep.Server depends on this feedback to update the reputation score of the considered participant such that participant's reputation is increased if the feedback exceeds some threshold ϵ. Otherwise, participant's reputation is decreased as follows:

$$R_{p_i} = \begin{cases} min\left\{\hat{R}_{p_i} + f_{RID^0_p}, 1\right\} & \text{if } f_{RID^0_{p_i}} \geq \epsilon \\ max\left\{\hat{R}_{p_i} - f_{RID^0_p}, 0\right\} & \text{if } f_{RID^0_{p_i}} < \epsilon \end{cases} \tag{5.5}$$

Re-Authentication

A participant asks for a new pseudonym to join an upcoming campaign with a different identifier. Thus, he sends his current identifier $RID^0_{p_i}$ to the Auth.Server. Now, Auth.Server knew earlier that this identity is valid. Thus, this time and in each subsequent renewal, the Auth.Server generates a new identifier (e.g. $RID^1_{p_i}$) using a new random value $r^1_{p_i}$ as described in the following equation.

$$RID^1_{p_i} = E(ID_{p_i}, r^1_{p_i})_{k_{Aut.Server.pub}} \tag{5.6}$$

The new identifier is forwarded to both the participant and the Rep.Server in the form of a new identifier as follows.

$$New_{ID} = \left[RID^0_{p_i}|RID^1_{p_i}\right]_{k_{Aut.Server.priv}} \tag{5.7}$$

There are several reasons for sending both the old and the new identifiers simultaneously to the Rep.Server. First, the Rep.Server links the reputation score of $RID^0_{p_i}$ to $RID^1_{p_i}$. Secondly, recording this new identity allows the participant to use this new identifier to contact the App.Server for a new task such that he is correctly validated by the Rep.Server. In addition, the Rep.Server discards the old pseudonym $RID^0_{p_i}$ such that it cannot be used again by an adversary. Malicious participants who try to use some old pseudonym (another form of the replay attack) are detected through the authentication phase since old pseudonyms are discarded.

5.5 Security Analysis

The goal of our evaluations is threefold: (1) analyze the robustness of our proposal against the threats identified in Section 5.3.2, (2) empirically evaluate the performance of PrivaSense, (3) compare PrivaSense with the existing systems. The first goal is discussed in this section whereas the second and the third goals are discussed in Section 5.6.

5.5.1 Objectives

A comprehensive privacy preserving reputation system in participatory sensing campaign has to guarantee some objectives for the participants. In this section, these objectives are recalled and PrivaSense is analyzed according to them.

1. ***Anonymous demonstration***: In PrivaSense, reputation scores are transferred from Rep.Server to the App.Server in a form of reputation certificate. Reputation certificates do not include original identities of participants, they include the pseudonym of the considered participant. Thus, reputation scores are anonymously demonstrated. The App.Server can verify that this demonstrated score is correct while keeping the anonymity of the original identity.

2. ***Anonymous assignment***: A participant is allowed to be anonymously assigned an initial reputation score. At the beginning of the campaign, Auth.Server sends the first pseudonym granted for a specific participant. There are not any old pseudonyms assigned to this participant. Thus, the pseudonym message just includes this first pseudonym. Rep.Server detects that this participant has just logged in the sensing campaign. Since then, an initial reputation score is assigned to this participant. In addition, the feedback calculated according to the trust of the most recent contributions should be assigned to the correct identity anonymously. For this, App.Server evaluates the contribution of each participant and assigns a feedback for the pseudonym of its provider. This feedback is transferred to the Rep.Server, which links the feedback to the same pseudonym stored at his databases. Therefore, participants are assigned both initial scores and feedback anonymously.

3. ***Anonymous reputation update***: An old reputation score $\hat{R}_{p_i}$ of a specific participant p_i is updated to a new reputation score R_{p_i} according to the considered feedback at the Rep.Server using the pseudonym of the participant. In addition, participants change their pseudonyms by asking Auth.Server for a new pseudonym. Auth.Server sends both the old and new pseudonym in one message to the Rep.Server. Thus, Rep.Server updates the pseudonym while preserving the same reputation account. Consequently, PrivaSense has the ability to anonymously update the reputation of participants according to their assigned feedback. Moreover, it anonymously preserves the reputation accounts even if participants change their pseudonyms.

The previous objectives concern the process of reputation management and tracking participants' behaviors in the absence of participants permanent identifiers. We have argued how PrivaSense could achieve these objectives. Our second concern is to show that these objectives are ensured such that they do not conflict with the privacy preserving objectives. Thus, looking at the privacy preserving objectives, we can observe the following:

1. ***Non-associativity***: To avoid the privacy oriented attacks, contributions should not be associated to a specific identity even if this identity is anonymous. First, contributions are submitted anonymously (i.e. using pseudonyms), therefore it is difficult to link them to a specific original identifier. Contributions are associated to the pseudonyms which change for every interaction. Second, a participant p_i applies a privacy preserving transformation to the contributed content. That is why, it is expected that attackers could not link a contribution to a specific identity.

2. ***Un-linkability***: Linking multiple contributions to the same identity leads to re-identifying original identities. We have considered the parameters that are used to link different contributions to a specific identity. First, pseudonyms change after each contribution. This

unlinks different contributions from the identity. Second, anonymizing reputation scores and transferring them in an anonymous reputation certificate prevent linking consecutive contributions to a specific identity. Adopting a privacy preserving mechanism on the contribution content makes it difficult for the attacker to observe the similarity inherited in every two consecutive contributions.

5.5.2 Attack Resilience

We observe below the resilience of PrivaSense against the reliability oriented attacks:

- ***Sybil Attack***: Through Sybil attack, adversaries try to generate multiple authentic identifiers for the same identity. PrivaSense system is protected against this attack. If the attacker tries to generate a new RID based on the previous one using a new random number according to Equations 5.6 and 5.1, the resulting pseudonym RID will not match any valid authentication RID stored at the Rep.Server. This is because the attacker does not know the randomization seed used by the Auth.Server, and hence the attacker will not be able to generate the same series of randomized RIDs that match the real ones.

- ***Replay Attack***: The attacker cannot directly use the pseudonym RID twice since each RID is allowed to be used only once and is discarded after the use by the Rep.Server. Replaying the same pseudonym for the second time will result in failure in the authentication phase. The Rep.Server by looking up the history of this pseudonym can find that it has joined a session before. Then, it sends a non-valid acknowledgment to the App.Server. This allows the App.Server to know that this pseudonym is not valid.

  A replay attacker can also attempt to demonstrate a recent pseudonym and an old reputation score to the App.Server. Attackers are prevented from launching such a replay attack as both reputation scores are updated and demonstrated by the Rep.Server. Thus, reply attackers can not access their reputation certificates such that they can replay them. Moreover, attackers may have the ability to access reputation certificates through sniffing the network. However, even then they are not be able to replay them since the certificates include pseudonyms which become invalid after their first usage.

- ***Reporting falsified sensor readings***: Malicious participants may themselves provide the system with corrupted data or they may provide these data on behalf of the others. Thus, they cannot only disrupt the system but also degrade the reputation of the others. First, PrivaSense adopts a reputation system that is responsible for detecting bad contributions and their providers. Second, if malicious participants try to report falsified sensor readings on behalf of other participants to degrade their reputation, PrivaSense protects honest participants against this attack by requesting participants to authenticate with Auth.Server and Rep.Server. These entities verify the validity of the pseudonyms before considering their contributions. Such an attack would only be successful if attackers access the original identities of the targeted participants and those of their respective pseudonyms.

Resilience against privacy oriented attacks:

- ***Identity and Data Re-identification (IDR)***: Participants authenticate themselves using their anonymous identities RID. RID is encrypted by Auth.Server. Thus, it does not reveal information concerning the participant's real identity. Therefore, an adversary cannot reveal the real identity of the sensing report provider unless he has access to the private key of the Auth.Server. In addition, participants adopt one of the existing privacy preserving cloaking mechanisms to anonymize their sensed data. Therefore, our system ensures participants anonymity from both the identity and data point of views.

- ***Reputation based Re-identification (RR attack)***: Given that the anonymity of identity and the sensed data are ensured, the supplementary objective of designing a privacy preserving reputation protocol is to simultaneously ensure the un-linkability based on the reputation scores assigned to participants. PrivaSense adopts a double fold anonymization for reputation accounts. First, Rep.Server forwards an anonymous reputation certificate to App.Server as described in Equation 5.4. Consequently, the App.Server cannot link the assigned reputation score to the original identity. In addition to that, reputation scores are cloaked based on Equation 5.2. So, App.Server knows only the cloaked reputation related to pseudonym RID. That is, adversaries (e.g. any entity sniffing the network or even the

Table 5.2: Default Parameter Settings

Parameter	Value
Number of participants for each task, n_P	150
Number of adversaries for each task, A	40
The mean value of correct data, μ	60
The standard deviation of correct data	5
The mean value of adversary data, $\mu + \mu/3$	80
The standard deviation of adversary data	0

App.Server) cannot link two consecutive reputation scores assigned to the same participant based on neither the identity nor the value of the reputation. Rep.Server is the entity that knows the sequence of randomized identifiers RIDs assigned to each participant. However, this is not an issue since Rep.Server does not know any more data concerning the participant (e.g. sensed data).

5.6 Experimental Results

5.6.1 Evaluation Setup

Simulation Model

Let us now describe the simulation model of a noise monitoring participatory sensing application similar to [242]. Each simulation involves running the example application in the sequence described in Section 5.4. Each participant is assigned GPS timestamps and coordinates taken from a taxi mobile traces real dataset [238]. The dataset contains the GPS timestamps and coordinates of approximately 500 taxis collected in the San Francisco Bay Area. For the first interaction, the Rep.Server simply assigns participants with some initial reputations while subsequent reputation values are calculated as described in Section 5.4.2.

We synthesize the noise distribution in an urban environment by assuming the data agree with the real noise levels described in [86]. We consider a quiet sensing area where the mean μ and standard deviation of the correct noise data is 60 db and 5 db respectively. An honest participant sends correct sensing data. We also include malicious participants in the simulation to reflect a more realistic usage scenario. A malicious participant sends false sensing data. We set the false data to contradict with the correct data. Therefore, the mean of false data is $\mu + \mu/3$ (i.e. 80 db). This means that malicious participants contribute data which correspond to a completely different level of noise. Thus, even one false report has a significant impact on the measurements. In addition, all false reports support each other. Thus, the standard deviation of false data is set to 0. Hence, we look for the worst case when all malicious participants collude to cause the biggest possible disturbance to the system. Table 5.2 lists our default parameter settings.

Evaluation Metrics

We evaluate our proposal according to three metrics to measure the levels of anonymity and reliability as follows:

Links First, RR Links metric measures the number of contributions linked to their successors in each campaign based on the RR attack defined in Section 5.3.2, whereas, PrivaSense Links are the number of links detected based on RR attack even after the adoption of PrivaSense.

Linkability PrivaSense makes the Links metric irrelevant, because not only the participant's identity and data are anonymized but also the reputation scores. Thus, the links that can be detected due to the reputation scores are mostly removed. To measure the effect of PrivaSense on the participants' anonymity, we use another metric called linkability. For this metric, a set of potential successors of each pseudonym is defined such that it contains a list of pseudonyms that can be the successor of the target participant. A larger potential successors set ensures a better anonymity. Note that the following description is applied to one reputation update. First, the Euclidean distances between the location of $RID^0_{p_i}$ and the location of all the pseudonyms used for providing subsequent contributions are calculated. Then, the pseudonyms which ensure a distance less than λ are considered as the potential successors for $RID^0_{p_i}$. Next, an adversary selects a subset of the potential successors whose reputation scores are closer to the reputation of $RID^0_{p_i}$ noted as (β_s). The linkability metric is defined as $\frac{1}{\beta_s}$. Smaller values of this metric indicate better anonymity and vice versa. We have considered linkability according to location and reputation scores. Whereas, both time and sensed data can also be incorporated. Intuitively, this would result in higher levels of linkability.

Accuracy App.Server calculates a weighted sum of the aggregated data using reputation scores as weights. To measure the disruption incurred due to the anonymization of reputation scores, we compare the RMSE of the average noise levels calculated based on the anonymized reputation scores against the RMSE calculated using the original reputation scores without anoymization. RMSE measures how much error there is between two data sets. In other words, it compares a predicted value and an observed or known value. The RMSE between two vectors of values is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{nc} (\upsilon_{1,i} - \upsilon_{2,i})^2}{nc}} \tag{5.8}$$

Where nc is the number of contributions, $\upsilon_{1,i}, \upsilon_{2,i}$ is the ground truth and the collected data respectively.

5.6.2 Results

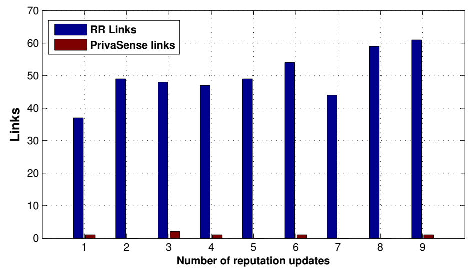
Privacy

Links In this experiment, we measure the effect of RR attack. We compute the number of contributions that can be linked to their successive contributions in each two consecutive campaigns. clk_{intr} is set to 0.5. This value imposes a slight effect on cloaking reputation scores beside assuring an acceptable level of anonymity. The results are depicted in Figure 5.3. It is evident that a large number of participants are linked to their contributions. In sub-figure (a), where P=100, RR attack detects around 40 up to 60 RR links out of 100 contributions received from 100 participants. Which means, on average, that half of the contributions are linked to their successors. Whereas in sub-figure (b), where P=300, the number of RR links detected vary from 30 to 80 out of 300 contributions (i.e. on average approximately 20% of RR links detected). However, in sub-figure (c), the links detected almost exceed in range 60 to 90 links with each interaction. In sub-figure (d), the number of participants becomes 500, where the number of the links detected decreases. This is because the number of participants increases leading to more repetitions in the reputation score assigned and subsequently less uniquenesses and less links detected. That is, the number of RR links detected depends on the number of participants P involved in the campaign. In our experiments, we observe between 20% and 50% links detected. We may estimate that approximately 35% of RR links are detected on average.

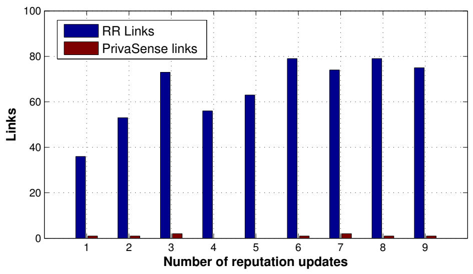
In the same figure, we can notice the links detected when PrivaSense is adopted. In both sub-figures (a) and (b), the PrivaSense links detected are very small. This is because the anonymization of the reputation scores added to using anonymous certificates as described in Section 5.4.2 are effective. The few PrivaSense links detected refer to the ones for which the noise added is zero (i.e. $ida = 0$), since $ida \in [0, clk_{intr}]$. That is, the links detected can be removed if we do not include 0 as a member of the cloaking interval. That is $ida \in (0, clk_{intr}]$ since zero as a noise conserves the original values and subsequently keeps the links.

Linkability In this experiment, we measure the linkability metric as depicted in Figure 5.4. We show the linkability for a subset of participants selected randomly from the dataset, as well as the average values of this metric over all participants. We set clk_{intr} to 0.5 and 0.3 and show the results in Figure 5.4 (a) and (b) respectively.

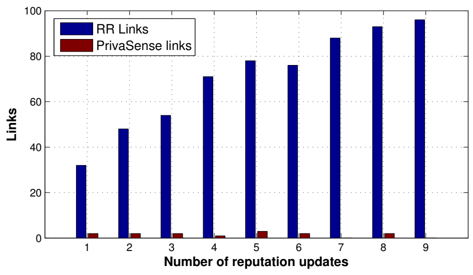
From the results of RR attack, we concluded that adversaries can construct a profile for each participant. Then, we attempt to measure the effect of PrivaSense to defend against this attack. In the results of our system depicted in Figure 5.4 (a), the average probability of a successful linkage is reduced to around 30% at the beginning of the sensing campaign with the first reputation update. As time progresses, the average probability continues to decrease and it reaches 6% at the end of the campaign. This is equivalent to a 94% average improvement. To explain the declining trend in Figure 5.4, we recall that the linkability technique works on location coordinates in successive time intervals. That is, if the adversary made a false link between contributions in Task t and $t + 1$, the error in the spatial information would propagate and compound to that at $t + 2$, which makes it increasingly difficult to track participants.



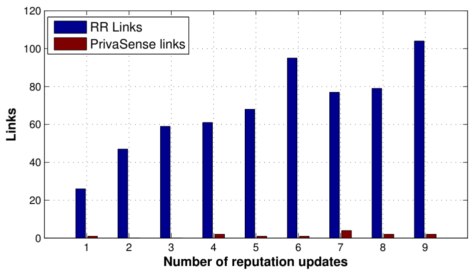
(a) $P = 100$



(b) $P = 300$



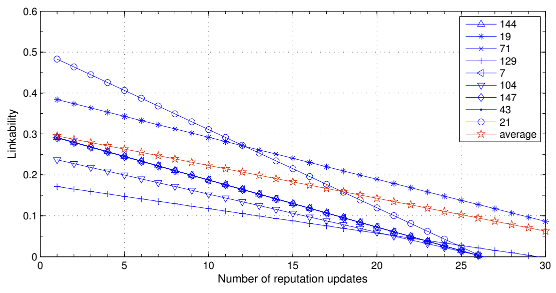
(c) $P = 400$



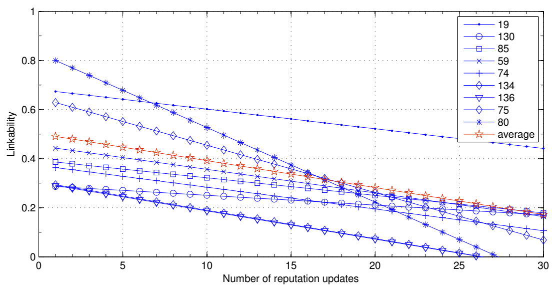
(d) $P = 500$

Figure 5.3: Number of links detected using RR attack with and without the protection provided by PrivaSense

In Figure 5.4 (b), we have used a much lower value of cloaking range $clkintr$. We can see that the average linkability starts at 50% at the beginning of the sensing campaign and decreases to around 20% at the end of the campaign. This is equivalent to 80% average improvement. Comparing the results in sub-figure (b) with the one in (a), we can conclude that using higher values for clk_{intr} in Sub-figure (a) allows for much lower values of the average linkability. This leads to better anonymity.



(a) $clk_{intr} = 0.5$



(b) $clk_{intr} = 0.3$

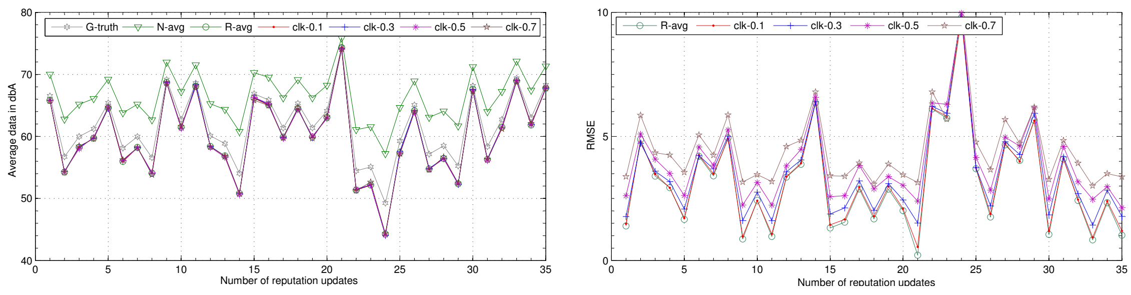
Figure 5.4: The linkability for randomly selected participants

According to this experiment, it is clear that our reputation cloaking mechanism introduces better anonymity and un-linkability for the participants engaged in the sensing campaign. However, we should now evaluate the effect of using such cloaked reputation scores on the aggregation of the collected data to define how much it deviates from the aggregation based on the original reputation scores.

Accuracy In this second experiment, we have used different values of clk_{intr} to cloak the reputation scores. Although, using large values for this parameter allows for better anonymity and un-linkability, as demonstrated in the previous experiment, it also affects the accuracy of the aggregated data. We consider a real world participatory sensing application as discussed above. Figure 5.5 (a) depicts the average of the weighted sum of the aggregated contributions. $R-avg$ denotes the weighted sum average of the contributions using the original reputation scores as weights. $clk-0.1$, $clk-0.3$, $clk-0.5$, and $clk-0.7$ are the average weighted sum of the contributions based on the cloaked reputation scores using different values for the cloaking interval (i.e.. $clk_{intr} \in \{0.1, 0.3, 0.5, 0.7\}$). The figure also includes the average of the aggregated data calculated without incorporating any weights noted as $N-avg$.

It is clear from figure 5.5 (a) that, the normal average $N-avg$ calculated without adding any weights is far from the ground truth while the reputation based average $R-avg$ is much closer to the ground truth. It is obvious that incorporating reputation scores in data aggregation gives better insights about the ground truth. In addition, it is clear from the figure that each of the averages calculated based on the cloaking intervals of $clk-0.1$, $clk-0.3$, $clk-0.5$, and $clk-0.7$ do not deviate significantly from the average calculated based on the original reputation values $R-avg$. That is, our cloaking does not disrupt the aggregated data significantly.

To better quantify the distortion occurred due to the incorporation of our reputation cloaking mechanism, we measure the RMSE for the same experiment and depict it in Figure 5.5 (b). It appears that the RMSE calculated according to the actual values of the reputation scores, $R-avg$, has usually a lower RMSE value compared to those calculated based on cloaked reputations. In addition, using lower values of cloaking intervals clk_{intr} ensures RMSE values which are closer to those calculated according to the actual reputation scores (i.e. $RMSE(R-avg) < RMSE(clk-0.1) < RMSE(clk-0.3) < RMSE(clk-0.5) < RMSE(clk-0.7)$). That is, cloaking based on lower values of cloaking interval clk_{intr} allows for aggregating data closer to the ground truth and a lower RMSE.



(a) Aggregated Data (b) RMSE

Figure 5.5: The effect of cloaking interval's size on the aggregated data and RMSE

Comparisons

Table 5.3 provides us with a general overview of the objectives considered of our work compared with the previous work. It is clear that the work presented in [148] is the one that tries to ensure the same privacy objectives.

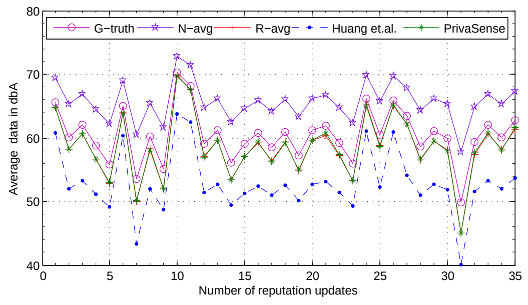
We compare our proposal against a state of the art previous work [148]. This work specifically intended to ensure the same objectives and to defend against the same attacks as PrivaSense. That is, we compare the RMSE of the aggregated data based on [148] with the RMSE of PrivaSense. In Figure 5.6, the RMSE of PrivaSense has lower values compared with the previous work by Huang et al. This shows that the aggregated data according to our proposal can be more accurate.

To summarize, PrivaSense introduces better anonymity through better un-linkability. In addition, the aggregated data are more accurate compared with the previous work [148]. In PrivaSense, the overhead for generating pseudonyms has been transferred to the Auth.Server and Rep.Server. In contrast, in the literature, participants are usually engaged with other entities in order to generate their pseudonyms. This ensures that in PrivaSense, participants' devices are much less overloaded with cryptographic and computational tasks which may lead to battery drain. We consider implementing a real world application and measuring the battery drain as a future work.

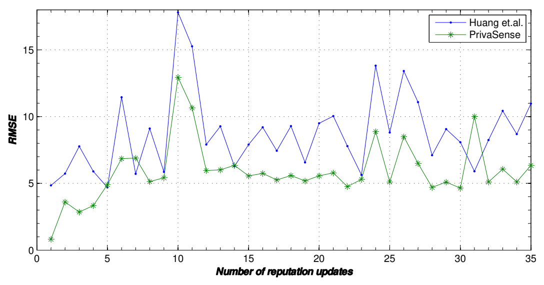
Table 5.3: Comparison of the systems

System	Distribution	Methodology	Anonymity	Characteristics										
				TPM			Reputation				Anonymity			
				Attestation	Data Protec.	Secure Boot	Traceability	Freshness	Separability	Exposure	Anon. login	Non-associative	Unlinked	Anon demon.
Dua et al. [105]	Hybrid	TPM	N-anon	√	-	-	-	-	-	-	-	-	-	-
Dua et al. [104]	Hybrid	TPM	N-anon	√	√	-	-	-	-	-	-	-	-	-
Gilbert et al. [118]	Hybrid	TPM	Anon	√	√	√	-	-	-	-	-	-	-	-
Saroiu et al. [254]	Hybrid	TPM	Anon	√	-	-	-	-	-	-	√	-	-	-
Gilbert et al. [119]	Hybrid	TPM	N-anon	√	-	-	-	-	-	-	-	-	-	-
Reddy et al. [245]	Hybrid	Rep	N-anon	-	-	-	√	√	-	√	-	-	-	-
Reddy et al. [244]	Hybrid	Rep	N-anon	-	-	-	√	√	-	√	-	-	-	-
Huang et al. [147, 149]	Hybrid	Rep	N-anon	-	-	-	√	√	-	√	-	-	-	-
Jkalidindi et al. [161]	Coll.	Rep	N-anon	-	-	-	√	√	-	√	-	-	-	-
Manzoor et al. [203]	Cen	Rep	N-anon	-	-	-	√	√	-	√	-	-	-	-
Amintoosi et al. [26]	Cen	Rep	N-anon	-	-	-	√	√	-	√	-	-	-	-
Amintoosi et al. [27, 29]	Cen	Rep	N-anon	-	-	-	√	√	-	√	-	-	-	-
Amintoosi et al. [28]	Cen	Rep	N-anon	-	-	-	√	√	-	√	-	-	-	-
Chang et al. [73]	Cen	Rep	N-anon	-	-	-	√	√	-	√	-	-	-	-
Wang et al. [283, 284]	Hybrid	Rep	Anon	-	-	-	√	√	√	√	√	√	√	√
Huang et al. [148]	Cen	Rep	Anon	-	-	-	√	√	√	√	√	√	-	√
Restuccia et.al. [248]	Cen	Rep	N-anon	-	-	-	√	√	-	√	-	-	-	-
Michalas et.al. [212]	Coll	Rep	Anon	-	-	-	√	√	√	√	√	√	√	√
Gisdakis et.al. [120]	Cen	Rep	Anon	-	-	-	√	√	-	√	√	√	√	-
Mousa et al. [218]	Cen	Rep	Anon	-	-	-	√	√	√	√	√	√	√	√

Anon	Anonymoys	**Cen**	Centralized	**Coll**	Collaborative
N-anon	Non anonymous	**Rep**	Reputation	√, **-**	Goal satisfied or not



(a) Aggregated Data



(b) RMSE

Figure 5.6: The effect of cloaked reputation scores in different systems

5.7 Conclusion

In this work, we proposed a privacy-preserving reputation system, PrivaSense, for participatory sensing applications. Our system adopts registration and authentication phases that ensure participants' anonymity and improve the system resilience against the Sybil and replay attacks. In addition, a privacy preserving mechanism is adopted for the contents of the participants' contributions, which prevents adversaries from using the data to infer the identity of participants. Moreover, data reliability is ensured due to the incorporation of a reputation system. Finally, PrivaSense adopts a mechanism to cloak the reputation scores of participants. That is, the participants can not be linked to their contributions according to their assigned reputation scores. The discussion and the results obtained based on a real dataset demonstrate that the PrivaSense system ensures better anonymity and un-linkability with a ratio that reaches about 80%, with a much low mean square error introduced into the aggregated data. For future work, we would like to consider ensuring the same objectives within a system model that avoids relying on trusted entities such as Rep.Server and Auth.Server.

Chapter 6

Anonymous Voting using Distributed Ledger-assisted Secure Multi-Party Computation

High voter turnout in elections and referendums is desirable in order to ensure a robust democracy. Secure electronic voting is a vision for the future of elections and referendums. Such a system can counteract factors that hinder strong voter turnout such as the requirement of physical presence during limited hours at polling stations. However, this vision brings transparency and confidentiality requirements that render the design of such solutions challenging. Specifically, the counting must be implemented in a reproducible way and the ballots of individual voters must remain concealed. In this work, we propose and evaluate a referendum protocol that ensures transparency, confidentiality, and integrity, in trustless networks. The protocol is built by combining Secure Multi-Party Computation (SMPC) and Distributed Ledger or Blockchain technology. The persistence and immutability of the protocol communication allows verifiability of the referendum outcome on the client side. Voters therefore do not need to trust in third parties.

This chapter is an adapted version of the paper: "A Transparent Referendum Protocol with Immutable Proceedings and Verifiable Outcome for Trustless Networks." M. Schiedermeier, O. Hasan, L. Brunie, T. Mayer, and H. Kosch. *In Proceedings of the 8th International Conference on Complex Networks and their Applications.* December 2019. Pp. 647-658.

6.1 Introduction

In recent years, a sharp decline in voter turnout has been observed in major elections [208]. For example, in the 2022 French legislative elections, the turnout was only 46.2% in the critical second round of the elections [90]. The voter turnout for the 2018 US midterm election was at 53.4% [63]. Though, compared to previous elections this is a high value, almost half of the population at voting age did not make use of their right to vote. In the US midterm elections in 2014 and 2010, the turnout was as low 36.7% and 41.8% respectively [44].

It is a longstanding goal to render the voters' active participation as effortless and convenient as possible in order to discourage low voter turnout. A secure voting system based on remote clients could greatly improve the flexibility of potential voters. It would significantly reduce the administrative overhead of postal voting and eliminate voters' obligations to be physically present at a voting station during limited hours.

In this work, we focus on referendums, which can be seen as a special instance of elections, with only two options offered for vote. Even though referendums are a simpler case of elections, implementing them correctly is still challenging [84] [266]. Many parties may have an interest in manipulation of the outcome. Furthermore, we consider the context of trustless networks, where we assume that participants place little to no trust in one another and there does not exist a central trusted authority, or such an entity is not desirable. A breach of the ballot-secrecy may result in harmful consequences for voters. Given this sensitive context, voters naturally seek solutions they can trust.

The classic analog way of conducting a secret referendum is having voters cast their ballots into boxes. This way they remain unlinkable to their votes. However, the logistic effort that is required for such an approach is tremendous. Ballot boxes must be set up, ballots with voting options must be printed and afterwards the counting must be realized by fair participants. The complex chain of implicit actions makes it hard to provide a proof of compliance for every single step. In this work, we try to address this problem with an electronic-referendum scheme that puts emphasis on transparency that is to say, full client-side verification of correctness.

6.1.1 Contribution

We propose a transparent referendum protocol with immutable proceedings and verifiable outcome. We define this immutability as the impossibility to tamper with the log of participant actions. Although there already exist protocols with similar ambitions, they usually require trusting third parties. We suggest a protocol that is based on a creative combination of existing cryptographic tools. In order to achieve transparency, we also asses the viability of our proposal considering mobile clients and discuss to which extent the protocol can withstand adversary attacks. Our evaluation concentrates on confidentiality of votes, transparency and immutability of proceedings and a verifiable outcome.

The key idea behind our contribution is to use a blockchain as a complete log of all communication between participants. While the secrecy of individual votes is ensured by an SMPC scheme, the log allows anyone with access to the ledger, to autonomously compare the actual proceedings to the expected protocol. This verification can occur locally. Participants therefore gain proof of correctness by themselves and not via third parties.

6.1.2 Outline

In this chapter, we first give an overview of related work (Section 6.2). Some of them follow strategies that are very different to our approach. We point out the issues that they pose and how we intend to address them. Next comes a presentation of our model (Section 6.3), followed by an enumeration of the cryptographic tools we apply within our protocol (Section 6.4). Afterwards, we delve into the exact phases and actions that describe our protocol (Section 6.5), followed by an evaluation in two parts. The first part (Section 6.6) discusses how well our initial objectives are met by the proposal. The second (Section 6.7) provides a security analysis where we evaluate different adversary strategies and their potential impact. Finally we present our conclusions and delineate the potential topics of further investigations (Section 6.8).

6.2 Related Work

In this section, we present articles that discuss how to design a protocol for electronic referendums. For each one, we outline the key idea and highlight associated disadvantages.

In [47] the authors describe how secret sharing schemes can be used as SMPC for Secret-Ballot elections. This work unarguably is the cryptographic foundation to our proposal. However Benaloh's formal model by itself provides no practical transparency for the participants. In his approach, security lies entirely in the applied threshold system, that is to say, participants have no dynamic feedback on the effectiveness of the applied security mechanisms. Our proposal not only protects the privacy of voters, it also transparently monitors if ballots have been potentially compromised.

In [94], an architecture for a privacy-aware electronic petition system is suggested and evaluated. As petitions typically express only two opinions (non-participation meaning approval, participation meaning disapproval to a topic), it can be considered a referendum system. The core element in this approach is involving anonymous certificates to elegantly restrict the referendum to eligible participants and eliminate double-spending in a privacy aware manner. However the suggested protocol does not provide enough transparency for an anonymous voter's participation: The act of participation by signing is not publicly transparent, therefore a dishonest petition server could discard signatures. The outcome would be indistinguishable from a case where the voter has never even contacted the server. Notably the voter has no way to prove the misbehavior of the signature-server. While our approach also involves anonymous credentials, we make sure that the semantic of issued tokens is independent of effectuated voting decisions. This allows us to ensure transparency, which ultimately renders dishonest server behavior detectable.

[300] and [301] provide a description of the distributed ledger-enabled privacy-preserving computation platform, *ENIGMA*. Our contribution differs in two aspects: 1) *ENIGMA* was not explicitly designed for referendums. Though the authors mention a general compatibility for such scenarios, its applicability for this context is not assessed in much detail. 2) In their platform, the ledger is

neither an exclusive data-store, nor is it used as the exclusive channel for inter-participant communication. Therefore participants do not obtain the same level of communicative transparency as in our solution.

[87] rely on a threshold system that can defend the secrecy of ballots up to a fixed number of colluding adversaries. However, their protocol provides no control mechanism to monitor whether such collusion was attempted or has already occurred. As such voters can not obtain certainty that their votes have actually remained undisclosed.

[65] identified similar objectives. They introduce a metric to measure voter privacy and examine how compromised systems perform under that metric. In reaction to this evaluation, they then suggest a protocol that performs well, given the metric. However, their protocol does not provide mechanisms for some other important goals, such as the prevention of ballot dropping.

The proposal by [189] discusses an IoT enabled protocol. The presented approach gains security by persisting encrypted votes in a blockchain. However there are two fundamental differences to our approach: 1) It does not include a client side analysis of communication meta-data, excluding an additional verification of protocol proceedings. 2) In the described model, there is a clear and intentional separation between the blockchain infrastructure and the voting devices. For registration and notably casting of ballots, the voters access the blockchain via a gateway. This separation of blockchain and clients also eliminates the possibility to perform integrity checks on client's side. Clients thus have to rely on external entities for full integrity checks of the blockchain.

[182] describe a blockchain based voting protocol. In contrast to our proposal, their solution involves a trusted third party for vote filtering. [37] also suggest a blockchain based voting system. However, in their system the blockchain arranges persisted votes in an immutable order. Therefore, voters can not update their vote, once it has been submitted. Our system does not rely on such a mechanism and therefore does not come with this restriction. In [249], the authors introduce a taxonomy of further notions for distributed voting protocols.

Song et al. [265] tackle the problem of scalability in anonymous voting implementations on the Ethereum platform. They identify several bottlenecks that impede scalability of prior solutions on Ethereum. One of the issues they resolve is the tallying failure due to the "no vote" from registered voters. The scheme demonstrates substantioal reduction in "gas" (the unit of resource consumption on Ethereum). For example, with 60 voters, the scheme consumes 1/53 of the gas compared to another state-of-the-art solution.

Zaghloul et al. [290] introduce the *d*-BAME electronic voting scheme that emphasizes mobility in addition to anonymity. The scheme relies on the participation of two opposing parties to ensure election integrity and accountability. The proposed scheme preserves voter privacy by using secure multiparty computations, which must be performed by parties that have conflicting allegiances. Their simulations show that the scheme can be deployed on smartphones in large-scale elections. Similar to our work, Zaghloul et al.'s scheme leverages blockchain as a public tamper-resistant bulletin board. However, they use a blockchain platform that implements smart contracts, whereas we do not impose this requirement in our work. Moreover, we note that in contrast to Zaghloul et al.'s proposal, our scheme does not have the requirement of the participation of opposing parties in the voting process.

Onur and Yurdakul [228] propose ElectAnon, a ranked-choice election protocol that focuses on anonymity, robustness, and scalability. ElectAnon uses zero-knowledge proofs to enable voters to cast their votes anonymously. Experiment results show that ElectAnon reduces "gas" consumption on Ethereum by up to 89% as compared to the state-of-the-art. The work also discusses how to reduce the requirement of trust in the election authorities.

Uribe et al. [278] describe anonymous voting solutions specifically for Decentralized Autonomous Organizations (DAOs). They observe that current DAOs use voting schemes that are not anonymous. According to the authors, the lack of anonymity in DAO voting results in numerous issues when it comes to confidentiality, voter influence, and voter turnout. Uribe et al. present a voting scheme for DAOs that enables confidentiality, in addition to maintaining ballot integrity.

6.3 Our Model

6.3.1 Participants

We distinguish between physical entities, identifiers and roles. Each physical entity possesses a unique and anonymous identifier. Furthermore, there are three roles that the physical entities can personify. A single entity can personify multiple roles, but not all combinations are allowed. The restrictions are explained in section 6.3.1.

Roles

Our protocol involves the following roles:

- **Initiator**: The initiator ensures all participants obtain the information required for the protocol execution. This role I is represented by a single physical entity $init$. The initiator provides a referendum context that comprises all information required by other participants to follow the referendum procedure. It is the only action $init$ ever performs. He notably does not participate in the subsequent voting or counting. The physical entity behind $init$ must not personify another role. This restriction hinders collusion, as it isolates referendum preparations from the entities executing the protocol.

- **Voters**: Voters are the devices of natural persons eligible to provide their opinion on the referendum context. We define the eligible set of k physical voter entities to a given referendum as: $V = \{v_1, ..., v_k\}$.

- **Workers**: Workers contribute to the execution of the protocol's underlying SMPC and provide intermediate results required to compute the referendum outcome and verification checksum. The set of n physical worker entities is a subset of the voter entities: $W = \{w_1, ..., w_n\}$, $W \subset V$. Workers are an example for physical entities personifying multiple roles. The physical entity behind each worker also, at some point acts as a voter. One advantage of this decision is that the total amount of entities, required to run our protocol decreases by $\mid W \mid$. In general, allowing a single entity to act on behalf of multiple roles is critical, as this gathers additional information at an entity. However, in this case the applied security mechanisms ensure that knowledge about a single ballot does not enable the worker to infer further information.

Identities

When we talk of *participants* P, we implicitly mean the physical entities behind voters and workers. Although with the definition $P = V \cup W$, P is equal to V, we intentionally introduce P for participants. Participants do not know one another directly, but only by an anonymous pseudo-identifier $\bar{p}$. Likewise we introduce the set of all pseudo-identifiers as $\bar{P}$. Only for illustration purposes, we denote a mapping function $id : P \rightarrow \bar{P}$ that translates a specific entity $p \in P$ to its associated identifier $\bar{p} \in \bar{P}$. It is important to state that in practice no entity must ever possess such a function. Participant anonymity is an essential element in our protocol. From this point on when we talk of *identifiers*, we implicitly mean *pseudo-identifiers*. Each participant holds a keypair. The private key is used for signatures and decryption. It never leaves the participant. The public key is used for encryption and also serves as a participant's identifier. We assume, that the initiator holds a complete list of all eligible voters' identifiers $\bar{V} = \{id(v) \mid v \in V\}$. We consider this to be a fair assumption, since *Diaz et al.* demonstrated how anonymous credentials can be issued among eligible voters, using an external credential server [94].

6.3.2 Ledger

A key component of our model is an immutable and integrity-protected data-store that is directly accessible by all participants. This is the ledger L. Access to the ledger enables the retrieval of persisted records and submission of new records. Persisted records however can be neither modified nor erased.

Ledger-Restricted Communication

Every participant locally operates a ledger-access node that allows him to retrieve records, submit new records and notably fully verify the ledger's integrity locally. We use the ledger as the *exclusive* communication medium among participants. As participants only know one another by their identifiers, they exchange messages by adding and polling ledger records whenever they communicate.

Message Notation

Every record added by communicating participants represents a message of format $m_{\alpha\beta}$. The index α specifies the sender's identifier, β the recipient's identifier. In case of broadcast messages no recipient β is provided. We distinguish between the following message types:

- b_α with $\alpha = id(init)$
  The Initiator's broadcast message, specifying the referendum parameters.
- $s_{\alpha\beta}$ with $\alpha = id(v_i), v_i \in V$, $\beta = id(w_j), w_j \in W$
  A voter sending a voting-related message to a worker.
- r_α with $\alpha = id(w_j), w_j \in W$
  A worker's broadcast message that contributes to the referendum outcome.
- c_α with $\alpha = id(w_j), w_j \in W$
  A worker's broadcast message that contributes to the referendum validation.

The authenticity of message origins is ensured by the author's signature. As the registration of voters' public keys, described in 6.3.1, can be realized over the ledger, it is fair to assume a trusted key-exchange among participants, prior to the referendum execution.

6.3.3 Adversary Model

We consider all voters and workers as potential adversaries. In section 6.5.2, we outline the exact *expected* behavior of referendum participants. Our adversary model covers that any Voter or Worker may deviate from this expected behavior at any time.

Malicious communication

In terms of message exchange, we consider:

- submission of syntactically incorrect messages, for instance messages that lack mandatory meta-information such as the signature.
- submission of semantically incorrect messages. This notably covers the submission of values out of a legal range, as well as incorrect result-values for delegated computations. This may also arise for header information, such as the sender field.
- submission of messages that by format or content are not covered by the phase in progress.
- inactivity where interaction is requested, that is deliberate non-communication.

Adversaries may deviate from the expected behavior individually or in groups.

Assumptions

We assume that all information in $b_{id(init)}$, verifiable by each individual participant, is correct. This is a fair assumption, as the referendum will not take place unless the participants agree to the published parameters. Furthermore, we assume that it is infeasible for adversaries to fake RSA signatures or break encrypted messages. Adversaries are not able to resolve the physical identity of other participants by inspecting network traffic. This is realistic if participants use TOR. Finally, we assume that adversaries do not have the resources to break the ledger's integrity. We assume that the ledger is based on a blockchain thus this property is ensured. One of the characteristics of blockchains is that it requires a practically infeasible computational effort to break their integrity protection [113]. We assume that the protocol either results in a provably correct result, or the participants can detect anomalies. However, we do not expect the participants to correct detected issues.

6.3.4 Objectives

We set the following four objectives for our proposed protocol:

1. ***Confidentiality***: The referendum must be conducted in such a way that it is impossible to infer the choices made by individual voters (other than what can be inferred from the outcome).
2. ***Transparency***: The referendum must be *transparent*. This means that every participant must obtain a complete trace of the operations performed, by whom and when. This notably covers the communication among participants throughout the referendum.

3. ***Verifiability of the outcome***: The referendum result must be verifiable to every participant. That means he must be able to autonomously evaluate the correctness of the result.

4. ***Immutability of Proceedings***: Proceedings are the logs of all actions performed by participants from the moment of referendum initialization until the determination of the result. Proceedings must arise directly upon execution of the described actions. Once persisted, proceedings must be immutable. That is to say it must be impossible to modify or even delete persisted proceedings.

6.4 Building Blocks

6.4.1 Secret Sharing Scheme

We chose the SEPIA [202] specification of Shamir's Secret Sharing, due to its good documentation and ease of integration. Shamir's Secret Sharing is an instance of SMPC schemes. As such, it allows to perform computations without having to reveal the original inputs to individual parties.

(t, n) Threshold Systems

A (t, n) threshold system allows splitting a secret into n shares in such a way that any t of them suffice to reconstruct the original secret. Subsets with less than t shares do not reveal any information about the secret. If the shares are distributed to multiple parties, we can create an effective mechanism against collusion. If shares of a secret are distributed among n parties, t of them must cooperate, to reconstruct the secret. With a greater value of t, the protection against collusive reconstruction rises. However, in case of a desired reconstruction, increasing the offset between t and n leads to enhanced robustness, as it makes the reconstruction redundant to the unavailability of single parties. The ratio of t to n can thus be adjusted, to meet a distributed protocol's security and robustness requirements.

Homomorphic Operations

We make use of a secret sharing scheme that supports additive and multiplicative homomorphic operations. This means the secret sharing scheme provides a way to perform operations on the shares of different secrets, so that the fusion of the resulting shares provides values that are equivalent to calculations done on the original secrets. Shamir's secret sharing supports both additive and multiplicative homomorphic operations. However the multiplicative component has side effects that limit its practical application. Specifically, it increases the amount of shares required for a later reconstruction of the result value. This problem is was first mentioned in [47]. The practical consequences for our protocol are discussed in section 6.7.

6.4.2 Distributed Ledger Technology

Our protocol relies on a precise log of communication that cannot be tampered with. We therefore use Distributed Ledger Technology as the communication channel among participants. Specifically, using a blockchain ensures that manipulation of persisted data is computationally infeasible. To do so an adversary would have to outperform the honest majority of mining participants.

6.4.3 Asymmetric Encryption

Although our protocol requires a complete listing of communication meta-data, there are good reasons to delimit the content of messages to the recipient. We therefore use asymmetric encryption to generate public-private keypairs which allow encryption and decryption of directed messages. Furthermore, these keys are used for message signing and authorship validation.

6.4.4 Anonymous Credentials

Anonymous credentials allow a restricton of services to specific users, without a need to verify identities at the moment of access. The key idea is to introduce an external entity that hands out cryptographic tokens to eligible users [74]. Those users can later use their credentials to gain admission to an access controlled service. Though modern implementations [199] respond to advanced requirements such as detection of double spending or a privacy aware verification of user

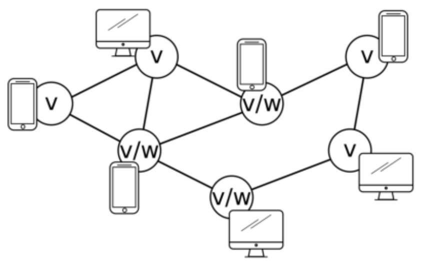


Figure 6.1: Illustration of referendum participants connected as blockchain nodes. Every referendum participant replicates the ledger. Although the nodes constantly synchronize, referendum related messages are exchanged exclusively via ledger-records. Therefore all clients hold a transparent copy of the proceedings. As pictured above, the protocol does not bind specific roles to particular hardware.

specific attributes, we only make use of the key feature, as it allows the anonymous registration of eligible voters.

6.5 The Protocol

The key idea behind our contribution is to use the ledger as a complete log of all communication between participants. This allows anyone, with access to the ledger, to autonomously compare the actual proceedings to the expected protocol. This verification can occur locally. Participants therefore gain proof of correctness by themselves and not via third parties. Furthermore, the anonymity of individual participants poses a hurdle for communication via side channels. This is discussed in more detail in section 6.7.

As our protocol is based on a secret sharing system, the introduction of a public ledger is counter-intuitive. Secret sharing systems usually gain security by dividing information into separate shares. Yet we suggest to store such shares side by side in a public ledger. We make this design feasible, by additionally protecting persisted shares with asymmetric encryption. This ensures that only an intended target entity has access to a specific set of sensitive information. At the same time, the ledger as an exclusive communication channel allows us to monitor the message meta-data of all participants. This allows clients to autonomously verify the absence of adversary collusion, targeted on the underlying (t, n) threshold system. Secret communication via side-channels is difficult as participants only know one another by their anonymous identity.

6.5.1 Protocol Overview

Our referendum protocol is based on a secure multi-party computation scheme, with the restriction added that all inter-participant communication occurs exclusively over a public ledger. That is to say, parties can only communicate by placing public messages in the ledger. Messages clearly state the recipient and are furthermore signed by the author. This provides a transparent and clear trace of all arising inter-participant communication.

The SMPC by itself allows a privacy aware computation of the referendum outcome. The SMPC's homomorphism ensures that computing entities do not learn about sensitive input data, since they work on an encrypted transformation of the data.

Proof of conformity to the designated protocol is supported by the ledger's immutability. Voters can analyze communication meta-data of the executed SMPC. This way every participant can assess whether the actual communication followed the protocol. As all information required to perform this validation is stored in the ledger, referendum participants can implement all compliance checks locally, without the need to trust third parties. This allows the protocol to function in a trustless network environment.

Ultimately, after a successful validation of the proceedings, each voter holds the certainty that the outcome was determined correctly and no vote has been compromised.

6.5.2 Protocol Outline

Figure 6.2 illustrates how individual roles chronologically submit and retrieve messages to the ledger. For each action, it also indicates the corresponding protocol phase.

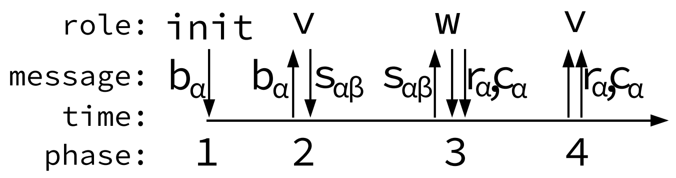


Figure 6.2: Illustration of *protocol phases*. Downward arrows indicate the persistence of messages *types* into the ledger, upward arrows indicate the lookup of messages (indicated by *type*). Time advances from left to right.

1. ***Initiation***: In this step, the referendum conditions are written to the ledger: *Referendum context, voting options, identities of registered voters, etc.*
2. ***Vote submission***: Voters look up the referendum conditions and deposit their ballots, secured by the secret sharing scheme and asymmetric encryption.
3. ***Intermediate result computation***: Workers perform homomorphic operations on the secured ballots, then write intermediate results and checksums back to the ledger.
4. ***Determination and validation of the outcome***: Voters pick up the intermediate results and checksums to determine the final outcome and run verifications.

The next section provides more details regarding the individual phases.

Initiation

The goal of the first phase is to ensure that all participants operate on identical referendum parameters. The referendum initiator *init* ensures this with a single broadcast message:

1. *init* places an initial broadcast message $b_{id(init)}$ in the ledger. The content of this message, $\tilde{b}_{id(init)}$ accumulates all static referendum parameters. It includes:
   - The identities (public keys) of all eligible voters: $\bar{V} = \{id(v_i) \mid v_i \in V\}$.
   - A subset of identities that names the designated workers: $\bar{W} = \{id(w_j) \mid w_j \in W\}$ as well as the individual share affiliation. The latter is required by the voters in the next phase, so they know which share belongs to which worker.
   - The referendum context and semantics of numeric voting options. This can be for instance:
     Are cats cooler than dogs? Yes $= +1$, *No* $= -1$.
   - A set of time-stamps (or block numbers) that define the transitions between subsequent phases $Q = \{q_{1-2}, q_{2-3}, q_{3-4}\}$. The fixed time stamps (or block numbers) are required to ensure that at the start of each phase all required input data is present in the ledger. As q_{1-2} marks the transition to phase 2, this timestamp matches the moment of placing $b_{id(init)}$ in the ledger.

   By communicating these conditions through a ledger, all participants obtain the exact same understanding of the expected referendum proceedings. This initial message contains all information required to outline further communication among participants.

Vote Submission

In the second phase, voters cast their votes. Each voter $v_i \in V$ does the following:

1. v_i retrieves the initiator's broadcast message from the ledger.
2. v_i secretly chooses his personally preferred voting option and determines the corresponding numeric value ψ_i. The mapping is specified in $b_{id(init)}$.
3. Based on ψ_i, voter v_i then generates a set of n shares $\{\sigma_{i1}, ..., \sigma_{in}\}$. He does so following a (t, n) threshold secret sharing scheme. The exact parameters for this step are provided in $b_{id(init)}$.

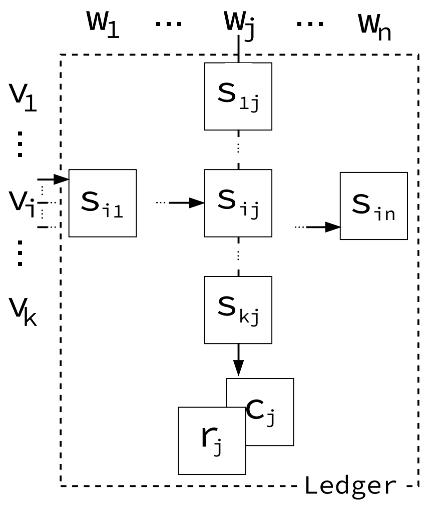


Figure 6.3: Illustration of *vote submission* by a voter v_i and *intermediate result computation* by a worker w_j. Note that all messages arising throughout these steps are persisted in the ledger.

4. Each generated share is intended for a specific worker w_j. Voter v_i encrypts each generated share σ_{ij} with the corresponding worker w_j's public key $\tilde{w}_j$. The exact mapping of shares to workers is once more described in $b_{id(init)}$. The target worker's id is also the public key to use for encryption.

5. v_i packs all n cypher-shares $\tilde{s}_{ij} = pub_j(\sigma_{ij})$, $j \in \{1, ..., n\}$ individually into n messages s_{ij} and initiates their persistence in the ledger. The horizontal arrows in Figure 6.3 illustrate this step.

Voters can perform the above steps until timestamp (or block number) q_{2-3} is reached. Repeated submissions before the deadline are allowed. Those are considered an update to one's own ballot. Messages s_{ij} submitted after q_{2-3} are considered non-compliant to the protocol and will be ignored.

Intermediate result computation

In the third phase, each worker w_j performs the following actions to contribute intermediate result values for the referendum outcome and checksum computations:

1. w_j retrieves the set of k encrypted share-messages destined to him: $\{s_{1j}, ..., s_{kj}\}$.

2. w_j retrieves the payload of received messages and this way holds k shares, each encrypted with his public key: $\tilde{s}_{1j}, ..., \tilde{s}_{kj}$.

3. w_j decrypts every single share using his private key and obtains a set of k unencrypted shares: $\{\sigma_{1j}, ..., \sigma_{kj}\}$. These are the k shares, the voters $V = \{v_1, ..., v_k\}$ securely communicated to him via ledger.

4. Based on $\{\sigma_{1j}, ..., \sigma_{kj}\}$, w_j participates in the homomorphic calculation of intermediate result shares:

   - He contributes to obtaining the sum of all votes, with an intermediate result share $\tilde{r}_j$.
   - He contributes to obtaining the sum of all squared votes, with an intermediate result share $\tilde{c}_j$.

   The sum of squared votes will later serve to detect illegal inputs. Note that intermediate result shares $\tilde{r}_j, j \in \bar{W}$, respectively $\tilde{c}_j, j \in \bar{W}$ must be combined to obtain the actual results.

5. w_j converts $\tilde{r}_j$ and $\tilde{c}_j$ to broadcast messages r_j, c_j and makes those get persisted in the ledger.

The execution of the above steps by a worker w_j, leading to persistence of r_j and c_j, is illustrated in Figure 6.3 by a downward arrow. q_{3-4} marks the moment by which workers must have their intermediate results persisted.

Determination and validation of the outcome

In the final phase, voters individually reconstruct the referendum outcome and evaluate public proceedings' conformity. To achieve this, every voter v_i performs the following actions on the intermediate result shares $\{\tilde{r}_j \mid j \in \bar{W}\}$ and $\{\tilde{c}_j \mid j \in \bar{W}\}$:

1. v_i picks up the corresponding result and checksum messages: $\{r_j \mid j \in \bar{W}\}$ and $\{c_j \mid j \in \bar{W}\}$.
2. v_i obtains two sets of shares, by combining the message payloads: $\{\tilde{r}_j \mid j \in \bar{W}\}$ and $\{\tilde{c}_j \mid j \in \bar{W}\}$
3. He removes the protection of the threshold system for two specific values. Precisely, he combines the intermediate result shares $\{\tilde{r}_j \mid j \in \bar{W}\}$, respectively $\{\tilde{c}_j \mid j \in \bar{W}\}$. These sets of shares express the homomorphic equivalent of:
   - The referendum outcome, $r = \sum_{i \in V} \psi_i$
   - A referendum checksum, $c = \sum_{i \in V} \psi_i^2$

   Consequently by combining the corresponding shares, v_i obtains r and c. The checksum c allows the detection of illegal votes. As all votes are expected to be either of ± 1, it must hold that $c = k$. If that is not given, the participant directly knows that at least one illegal input value was submitted. Still, it is possible to generate a valid checksum with cleverly arranged illegal input values. We discuss this threat in section 6.7.

6.6 Analysis of Objective Fulfillment

In this section we evaluate how well the individual objectives are met by the suggested protocol.

6.6.1 Immutability of the Referendum Proceedings

Proceedings are immutable whenever they are preserved in a way that renders retroactive tampering infeasible. Given the presented protocol, proceedings can be expressed by a complete log of participant-exchanged messages. As those messages are exchanged publicly through the ledger, the ledger content itself serves as complete transcript of referendum proceedings. Since the blockchain ensures the immutability of persisted records, we obtain an immutable log of the referendum proceedings.

6.6.2 Confidentiality of Votes

A ballot is secret if no entity other than the voter himself knows the submitted value. Our protocol applies a strong protection of votes, by first splitting them according a secret sharing scheme and then encrypting the obtained shares asymmetrically. Unless an adversary manages to break asymmetric encryption or secretly gather the private keys of t workers for a collusive ballot reconstruction, the confidentiality of submitted votes remains ensured. Though asymmetric encryption mechanisms are theoretically breakable, it is commonly assumed a computationally infeasible task. That is to say with current hardware it is infeasible for an adversary to reconstruct a secret without the required key material. As workers only know one another by their pseudo-identifiers, it would be difficult for them to secretly establish a communication side channel for collusion. Even if they are able to communicate via a side channel, they would still need at least t corrupted workers to compromise confidentiality.

6.6.3 Referendum Validation

To verify the correctness of the referendum outcome, each participant must be able to validate that two conditions are met:

1. The inputs that the outcome evaluation occurred on, are valid. This means all votes must be valid numeric options. As we will see in section 6.7, this condition restricts the range of valid parameters for the (t, n) threshold system.
2. The evaluation itself was conducted correctly. This means that the intermediate results computed by the workers must be correct for the provided inputs.

The second condition can be ensured by redundancy. The polynomial based secret sharing scheme allows to detect and ignore outliers. Imagine 10 sampling points are provided for a polynomial of degree two. Now, if nine of them match the polynomial but a single point does not, this would suggest that the 10th support is incorrect. Assuming that intermediate results are verifiable, the worker-provided checksum allows a privacy aware input validation.

6.6.4 Transparency

A referendum is considered transparent if all participants possess a correct and complete log of all actions performed throughout the entire referendum. In our model, all actions eventually result in communication. As we force all communication to run through the ledger, the trace of deposited messages provides a transparent and verifiable log of actions.

6.7 Security Analysis

In this section, we evaluate whether adversary strategies are detrimental to the suggested protocol:

- **Intentional inactivity**: Any participant can violate the protocol by intentional inactivity where interaction is expected. Eligible voters can choose not to distribute shares or only send them to a subset of workers. A worker can ignore the expected submission of intermediate result shares. Although the payload of vote-messages is encrypted, all participants can inspect the ledger content and detect if eligible voters are inactive or do not communicate with all designated workers. The default strategy by honest workers is to systematically ignore all vote-shares of voters that do not comply to the expected behavior. Inactive workers are harder to prevent, but the redundancy of the (t, n) threshold system allows a determination of the evaluation *outcome* until up to $n - t$ inactive workers. However, in terms of the referendum outcome's *checksum*, the boost of sampling points required for reconstruction, lowers the protocol's robustness to a tolerance of at most of $n - t^2$ inactive workers. [47]

- **Syntactically incorrect messages**: Participants can violate the protocol by sending syntactically incorrect messages. Syntactic errors can be easily detected with syntax-schemes. The default strategy is to ignore any syntactically incorrect message. This way, messages that are in no relation to the protocol also have no impact. If ignoring the message results in an interpreted participant inactivity, the above inactivity analysis is applicable here too.

- **Impersonation**: Participants may try to illegally send messages in the name of another participant. Impersonated messages are easy to detect, since their signature does not match the expected author. Messages with invalid signature are systematically ignored.

- **Invalid voting options**: Voters are expected to vote for either ± 1. However, as their shares are submitted in encrypted form, they might try to boost their influence with higher (or lower) numeric values. For colluding participants, it is possible to arrange invalid votes in a way that the input validation checksum is still fulfilled.[1] However, this attack is not in the interest of the adversaries, since it can only diminish the overall influence of the outcome. If parties collusively submit illegal inputs that pass the validation, the impact of those inputs is lower than the impact they would have achieved with legal input values. This is a consequence of the *Cauchy-Hölder inequality*: $\sum_{k=1}^{n} \mid x_k y_k \mid \leq (\sum_{k=1}^{n} \mid x_k \mid^{p})^{1/p} (\sum_{k=1}^{n} \mid y_k \mid^{q})^{1/q}$, with $n \in \mathbb{N}, \{x_1, ..., x_n\}, \{y_1, ..., y_n\} \in \mathbb{R}, p, q \in [1, \infty)$.[2]

- **Incorrect intermediate results**: Workers might submit incorrect intermediate results on purpose. In case of an extreme threshold system configuration with $t = n$, the existence of incorrect result shares is neither detectable nor correctable. However, with rising share redundancy, an honest majority of workers can push incorrect shares into a detectable outlier position (see section 6.6). However, massively colluding adversaries could also push an honest minority into an outlier position. Another inconvenience for worker adversaries is that they cannot predict the effects of their manipulation. Given the SMPC, an altered value can influence the result in either direction.

[1] Example: Imagine two votes $\psi_1 = \sqrt[2]{1.5}, \psi_2 = \sqrt[2]{0.5}$ are submitted, their checksum is $\psi_1^2 + \psi_2^2 = 2$, while the resulting vote impact is $\psi_1 + \psi_2 \neq 2$.

[2] If we chose $p = 2, q = 2, y_k = 1$, the inequality is reduced to $\sum_{k=1}^{n} \mid x_k \mid \leq \sqrt[2]{\sum_{k=1}^{n} \mid x_k \mid^2} \sqrt[2]{n}$. However, the client side checksum verification ensures that $\sum_{k=1}^{n} x_k^2 = n$, which further reduces the inequality to $\sum_{k=1}^{n} \mid x_k \mid \leq n$. This maximum value is obtained with valid inputs $x_k \in \{-1, +1\}$, rendering a collusive construction of illegal inputs pointless, since such inputs cannot surpass the impact of valid values, on the referendum.

- **Double voting**: Voters can repeat the generation, encryption and distribution of shares. As the encrypted vote-shares are exchanged via the public ledger and sender and recipient remain un-encrypted header attributes, double voting is easily detectable. The default strategy is to discard all but the most recent share that a specific voter submits to a specific worker. A voter can thus update her choice, but not increase the impact.

- **De-anonymization**: Participants might be interested to identify the physical entity that operates behind a participant pseudonym. This would enable outside-ledger undetected communication. As all network traffic runs over TOR connections, a de-anonymization is not feasible.

- **Communication side channel creation**: Adversaries may try to secretly establish an alternate platform for direct communication, parallel to the ledger. Though secret inter-participant communication is a severe threat to the protocol's transparency, a resilient system can counter this by setting the threshold-value reasonably high. Specifically, this means that the probability of the random workers to fall into societal cliques must be minimized. If adversaries do not already know their physical identities, they have to communicate publicly, as they do not know who to address to. Adversaries publicly declaring their will to collude could be detected.

- **Voter exclusion**: In section 6.2, we criticized the usage of an anonymous credential server. However, in our case anonymous credentials are only used for registration, not for voting. In [94], a voter cannot expose a dishonest behavior of a petition server. He cannot prove his previous interaction with the server and it would reveal his voting decision. In our case both does not apply. The registration itself can be logged in the ledger. Likewise the keys of registered voters, since they can be logged as part of the public init message, $b_{id(init)}$. Thus a legitimate voter could easily prove his exclusion by a malicious server.

6.8 Conclusion

By bringing together the potential of blockchain technology and secure multiparty computation, we constructed a highly transparent referendum protocol that allows participants to autonomously verify proceedings and outcome. Traditional (t, n) threshold based systems gain security by selection of parameters that render successful collusive attacks unlikely. In this work, we further enforce security by considering inspection of communication data protected by blockchain technology. With exception to the anonymous credential issuer, the need for trusted third parties is eliminated. We provided a realistic adversary model and analyzed how our protocol withstands corresponding attacks.

In future research we would like to further investigate a meaningful selection of referendum parameters. We would also like to explore other input validation methods that have less impact on the voter-worker ratio. Another open question is how to best select the worker subset. Given the focus of our current work on the security aspects of the protocol, we are interested in performance evaluations of a practical implementation of the protocol, particularly in a mobile scenario.

Chapter 7

Collusion-Resistant Worker Set Selection for Transparent and Verifiable Voting

Collusion occurs when multiple malicious participants of a distributed protocol work together to sabotage or spy on honest participants. Decentralized protocols often rely on a subset of participants (who may be called workers) for critical operations. Collusion between workers can be particularly harmful to the security of the protocol. We propose two protocols that select a subset of workers from the set of participants such that the probability of the workers colluding together is minimized. Our first solution is a decentralized protocol that randomly selects workers in a verifiable manner without any trusted entities. The second solution is an algorithm that uses a social graph of participants and community detection to select workers that are socially distant in order to reduce the probability of collusion. We present our solutions in the context of a decentralized voting protocol proposed by Schiedermeier et al. [256] (described in Chapter 6) that guarantees transparency and verifiability. Enabling collusion-resistance in order to ensure democratic voting is clearly of paramount importance thus the voting protocol provides a suitable use case for our solutions.

This chapter is an adapted version of the article: "Collusion-Resistant Worker Set Selection for Transparent and Verifiable Voting." M. Bettinger, L. Barbero, O. Hasan. *SN Computer Science (Springer Nature).* 2022. Vol. 3, no. 5, article 334. This work was carried out as part of the Master's project of M. Bettinger and its subsequent follow-up.

7.1 Introduction

Voting is an essential element of a robust democracy. However, traditional polling site voting, which requires the voters to physically visit designated locations, poses several problems. For example, in a sanitary context such as the COVID-19 pandemic, health concerns of in-person voting are a major concern. The requirement of in-person voting may also result in low voter turnout [1]. Mail-in ballots pose their own set of challenges. For example, delays and associated controversies [2, 8] of the 2020 presidential election in the United States demonstrated the limitations of mail-in ballots. These issues show us that we need a more transparent and verifiable method to ensure democratic votes, which relies less on a central entity or authority. In response to this problem, Schiedermeier et al. [256] propose and evaluate a secure electronic referendum protocol for users that ensures confidentiality, integrity, transparency, and verifiability.

In this work, we use this protocol as an example of a decentralized process that needs to resist against colluding malicious entities. Collusion attacks involve multiple protocol participants working together to sabotage that protocol or steal information from it. We propose solutions for selecting entities in a transparent and decentralized manner that reduce the probability of collusion. Schiedermeier's protocol indeed operates in a trustless and decentralized network environment, which implies that the participants involved in the protocol do not have to trust each other or any third parties. Some steps of the protocol are carried out by a subset of the participants of the referendum called workers, chosen by an entity called the initiator. However, we identify that the initiator may act maliciously and handpick corrupted workers who can collude together in order to compromise the security of the protocol. This may include discovering the secret votes, corrupting the final result, or preventing a result altogether. We note that the challenge of worker collusion is not specific to the protocol by Schiedermeier et al. [256]. Worker collusion is a general threat to a range of secure decentralized protocols.

We propose two approaches to the selection of a set of workers that minimize the potential of collusion. Firstly, we propose a solution where workers are designated randomly among a set of participants. Secondly, given a social graph of the referendum participants, we use graph analysis and community detection in order to choose workers among them such that the social distance between them is maximized and thus the potential for collusion is reduced.

In this work, we use the protocol for transparent and verifiable blockchain-based voting by Schiedermeier et al. [256] as a context for the problem of selecting workers while minimizing the risk of collusion. The protocol is described in Chapter 6. Nevertheless, our solutions are not limited to Schiedermeier et al.'s protocol [256]. They can be applied to other contexts with the need for collusion-resistant worker set selection.

The outline of the rest of the chapter is as follows. We discuss previous works related to worker selection, as well as their collusion resistance properties in section 7.2. Next, in section 7.3, we describe and formalize the problem of collusion-resistant worker set selection. Our solutions are detailed in section 7.4 for verifiable random worker selection and section 7.5 for verifiable social graph aware worker selection. In section 7.6, section 7.7, and section 7.8, we focus on the experimental protocol that we use, the dataset and tools used, and the analysis of the experimental results, respectively. This is followed by a discussion of the findings in section 7.9 and the conclusion in section 7.10.

7.2 Related work

The following categories of works are presented in the order of decreasing needed insight about participants, ending with random selection of workers. The first works are based on monitoring participant behavior and their interactions, while the next two categories use the network topology of participants. Finally, work on random selection of workers is presented.

7.2.1 Reputation-Based Witness Selection

Fighting potential colluding workers through decentralized reputation systems as mentioned in [236] could be an interesting idea. There are however two main limitations to this approach given our use-case.

First, participants are pseudo-anonymized for a vote, and that pseudo-identity changes between each voting event. This prevents using past behaviour of participants to choose workers among them, like it is done by Aral et al. [35]. In their work, in the context of decentralized cloud computing, they detect patterns of workers that tend to fail together (accidentally or maliciously), in order to choose a worker set that maximizes the probability of success. Because we cannot link pseudo-identities between distinct voting events, we cannot use their approach. Even without pseudo-identities, the rarity of voting events and even rarer selections of a given participant as a worker mean few events when its reputation could be evaluated. This reduces the meaningfulness of computing a reputation score.

Second, an important requirement for any reputation system used in the scenario of voting would be strong privacy preservation. Given a real person, the secrecy of whether that person voted in a given event, and more importantly the content of the vote should be preserved. These security concerns regarding the voting phase are discussed by Schiedermeier et al. Our work focuses on a transparent and verifiable setup before that vote. That setup is designed to tolerate malicious participants, meaning it does not seek to identify them (e.g. for prosecution purposes). In some cases, it is indeed difficult to differentiate between accidental and malicious actions, for example with crashes. Therefore, as did the original protocol, we should adhere to the goal of participant privacy.

7.2.2 Leader Election

Leader (s)election, for example developed in [281], uses the network topology of participants in the system to select a subset of them as leaders. However, such schemes are not viable solutions for collusion resistance in our specific use case. Indeed, the purpose of these works is to choose some nodes as leaders, so as to minimize the distance between each graph node and its closest leader. Rather than distributing voters around leaders (in our case workers), we would prefer to distance workers from one another, which is not the same goal. Moreover, in order to preserve the confidentiality of the vote protocol, the heuristic for such leader selection would focus on the node IDs, which is the only known attribute known about participants. Again, if the initiator is responsible for the input data (i.e. the social graph needed and IDs), it could easily manipulate these IDs to make sure malicious agents are elected.

7.2.3 Decentralized Random Number Generation and Worker Selection

Our first proposed solution relies on random selection of a set of workers among participants. We use a blockchain as an immutable and transparent messaging hub for participants, so they can generate a random seed number in a decentralized manner in a single phase, which determines the set of workers.

A work close to our problem is Nguyen-Van et al.'s [224], who propose a decentralized multi-step protocol based on Homomorphic Encryption, Verifiable Random Functions (VRF) and distributed ledgers (e.g. blockchains) to generate random numbers. Homomorphic encryption enables adding (or multiplying) values while they are encrypted. VRFs generate random numbers as well as proofs that these numbers were obtained by running that VRF. The initiator in Schiedermeier et al.'s work [256] could request a random number to be generated and would also obtain a proof of correct execution through the pipeline. However, a limitation is that the initiator must not share the private key it uses for requesting the number, otherwise voiding tamper-resistance on the result. As the initiator is not trusted in our case, this method cannot be used.

7.3 Problem Statement

7.3.1 Adversary Model

We will call malicious all potential workers as well as the initiator likely to work together in order to form a coalition with the intention of disrupting the expected behavior of the referendum. We can distinguish three malicious behaviors resulting from workers' collusion:

- Discovery of the secret votes of the honest participants from the intermediate values.
- Manipulation of intermediate values in order to corrupt the final result.
- Prevention of the computation of the final result due to inactivity from the malicious workers.

The initial list of participants in the voting event is trusted. Notably, each pseudo-identity in the list corresponds to a unique and real participant, for example an officially registered citizen. This means that a malicious entity cannot create multiple fake pseudo-identities to sway the vote, namely, Sybil attacks are not possible. Similarly, people expected to participate in the vote are present in the list. We accept both assumptions for the following reasons:

1. An expected participant whose pseudo-identity is missing in the list can easily detect and report it, as the list is publicly available;
2. Reason (1) also means that for a list size equal to the expected number of real participants, malicious entities cannot replace the identities of other participants by theirs;
3. A list containing more pseudo-identities than expected real participants is likewise detectable and shows the presence of malicious fake participants, given reasons (1) and (2);

Note however that this does not prevent some other actions by malicious entities that increase their power in the vote, like:

- corrupting a real participant;
- obtaining control over a pseudo-identity (e.g. through phishing). This non-consensual loss of control by the honest participant may be reported and fixed before the voting event takes place, canceling the malicious action.

The effect of such actions is an increased number of malicious entities in the protocol. In this work, we simply count m maliciously controlled pseudo-identities in the participant list during an execution of the protocol.

7.3.2 Collusion-Resistant Worker Selection

An arbitrary selection of workers by a single entity (the initiator in the case of Schiedermeier et al.'s protocol [256]) can lead to collusion if the entity performing the selection is malicious. This is why we propose to remove this arbitrary choice, replacing it with a more neutral and secure process.

We will now define which constraints need to be upheld by our solutions to successfully achieve this task. Let m be the number of malicious workers among all n workers. We place ourselves in the context of a $t - n$ threshold-based Shamir secret sharing scheme [260], which allows splitting up a secret into n shares (each held by one of the n workers) in such a way that any t of them suffice to reconstruct the secret. Malicious agents cannot gain any information about the secret if they possess strictly less than t shares.

1. To ensure the secrecy through Shamir's threshold system-based secret sharing scheme [260], the number of malicious workers m shall not be more than t;

2. If there are less than t honest active workers, then a coalition of malicious workers can prevent reconstructing the value by being inactive. Therefore, m shall not surpass $n - t$ workers. Like the first constraint, this rule considers the presence of m malicious agents in a $t - n$ threshold system;

3. A checksum is computed at the same time as the referendum's result. Its purpose is to count the total number of votes. This prevents participants from voting illegal referendum values. For the referendum's checksum reconstruction, m must be less than $n - t^2$ workers (again against inactivity) [85].

Therefore we have:

$$n \geq t > m \geq 0 \tag{7.1a}$$

$$n - t > m \geq 0 \tag{7.1b}$$

$$n - t^2 > m \geq 0 \tag{7.1c}$$

The collusion limit is when $m = t - 1$, i.e., when the malicious coalition is missing one worker to be able to discover the secret. Replacing $m = t - 1$ into (3) gives us: $t^2 + t - 1 - n < 0$ and $t > 0$. Given n workers, the maximal t is therefore given by $t_{max} = \lfloor(-1 + \sqrt{5 + 4n})/2\rfloor$, based on verifiability constraints presented above.

7.4 Proposal 1: Verifiable Random Worker Selection

Replacing arbitrary worker selection with a verifiable random selection enables quantifying the probability of collusion. For example, in section 7.8, we show that even with a third of the participants as workers in the sampled social graph, the probability of collusion is less than 2.5%. Before, that probability was unknown, because the initiator was a trusted entity. Now, however, the probability $\mathcal{P}(X = m)$ of having sampled m malicious workers among M malicious participants knowing n total workers were sampled among P total participants follows a hypergeometric law $\mathcal{H}(n, \frac{M}{P}, P)$. This probability is given by:

$$\mathcal{P}(X = m) = \frac{\binom{M}{m}\binom{P-M}{n-m}}{\binom{P}{n}} \tag{7.2}$$

For example, let us assume that we want to have a probability of collusion of less than 5% and expect to have M malicious participants out of the total P participants. Plotting the hypergeometric law's Cumulative Distribution Function $CDF(n, m)$, for given values of M and P, and taking the intersection with the plane of probability 95%, gives us the curve $\mathcal{C}(n)$ such that there is 95% probability that there are less or equal than m malicious workers in a sample of n participants.

Our protocol computes a random number in a decentralized fashion. This number is used as a seed to randomly select workers among participants. This seed will be known by all. Therefore, the whole procedure can be verified by any participant. This solution introduces an additional phase in the voting protocol. This phase harnesses the immutability and ordering properties of messages committed on the ledger to generate a verifiable seed. Indeed, we can use these properties to generate a number in a decentralized manner by soliciting the participation of voters, who will

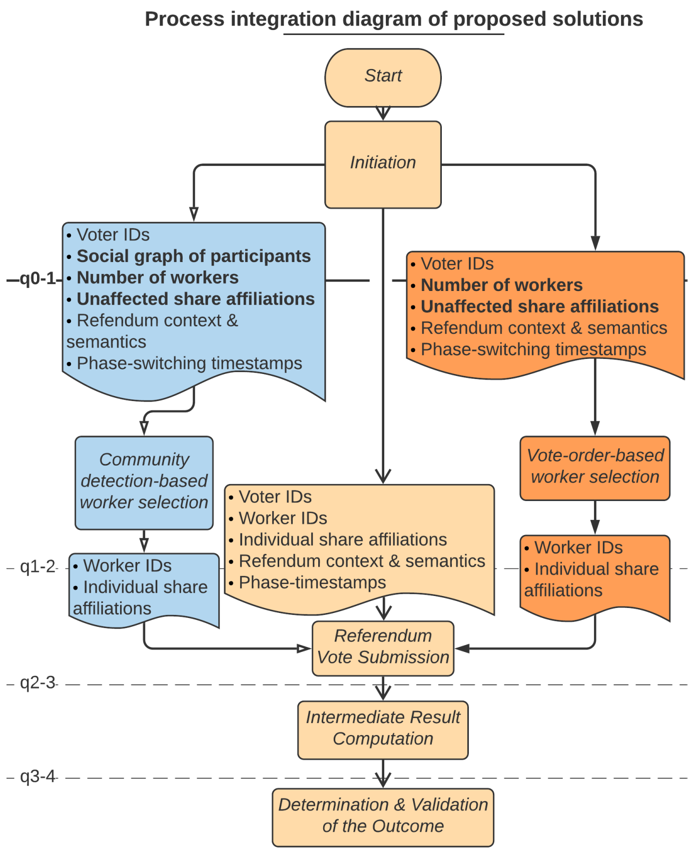

Process integration diagram of proposed solutions
Start
Initiation
• Voter IDs
• Social graph of participants
• Number of workers
• Unaffected share affiliations
• Refendum context & semantics
• Phase-switching timestamps
q0-1
• Voter IDs
• Number of workers
• Unaffected share affiliations
• Refendum context & semantics
• Phase-switching timestamps
Community detection-based worker selection
Vote-order-based worker selection
• Voter IDs
• Worker IDs
• Individual share affiliations
• Refendum context & semantics
• Phase-timestamps
q1-2
• Worker IDs
• Individual share affiliations
• Worker IDs
• Individual share affiliations
Referendum Vote Submission
q2-3
Intermediate Result Computation
q3-4
Determination & Validation of the Outcome

Figure 7.1: Process diagram of the initial protocol (in the center), with proposed solution protocols integrated onto it on the sides. q_{i-j} lines represent phase transitions as defined in [256], q_{0-1} is a new transition used only in vote-order-based worker selection (right-hand side). Rectangles represent process phases, "document"-shapes represent messages on the ledger. Document-shapes contain data items used in the protocol. Bold items are changes compared to the initial protocol.

deposit a message on the ledger. Our number is generated by comparing the order in which a subset of participants committed a message on the ledger, as well as the composition of that subset, to the ordered complete list of participants. Given the number generation algorithm and the messages on the ledger, any participant can verify the outcome of that phase.

The public ledger on the blockchain has the particularity to be immutable. It means that malicious entities cannot interfere with the ordering of messages on the ledger nor can they interfere with whether messages are published or not on the ledger. Blocks are appended sequentially to the blockchain, and each block contains a list of transactions (in our case our messages). We can use this sequential data structure to define an order of participant IDs.

7.4.1 Context

Underlying blockchain characteristics: Our solution relies on blockchain architectures that meet the following specifications:

- Consensus algorithm: should be so the malicious coalition lacks the necessary resources to manipulate or control it. For example, it can be based on hardware resources, like Proof-of-Work, or cryptocurrency assets, like Proof-of-Stake or its Delegated Proof-of-Stake variant.

- Openness/Access control: the blockchain system may either be public or permissioned. However, public blockchains are useful to have a large number of nodes maintaining them, which enables high decentralization of the consensus algorithm. The voting application logic would run on top of this consensus layer. Indeed, participants in the vote can be registered and identified in the vote's participant list by their public blockchain account addresses, or their corresponding public keys (for example with Ethereum [287], the account address is derived from a user's public key). Participants are therefore registered at the application-level (or smart contract-level, if present), while the underlying blockchain is open to the general public. This mechanism makes it easy to distinguish real from fake vote participants, in the context where the voting application coexists with other services (and their respective users) on the same blockchain. Because of the extended blockchain user base, which includes nodes maintaining the network, it also means participants are only required to be able to send transactions (containing the protocol messages) to the blockchain using their account.

- Smart contracts: the protocol only needs the blockchain as a messaging hub. As such, advanced features like smart contracts are not required. Nonetheless, smart contracts are useful to enable decentralized and easier enforcement of the protocol (e.g. for phase-switches, for considering only a voter's latest vote submission).

Inputs: Let $\mathcal{P}$ be an ordered list of P participant IDs, obtained through a trusted process (e.g. voter registration by government entities). Let n be the size of the subset of participants to select in $\mathcal{P}$. Furthermore, let $\mathcal{M} \subset \mathcal{P}$ be the set of malicious participants. Malicious participants are *a priori* indistinguishable from honest participants, i.e. no ID in $\mathcal{P}$ can be proven to be in $\mathcal{M}$ before the protocol starts. $M = |\mathcal{M}| \geq 0$ the number of malicious participants is unknown as well.

Outputs: Let $\mathcal{W} \subset \mathcal{P}$ be the set of selected workers.

Objective: Select $n = |\mathcal{W}|$ workers out of $\mathcal{P}$, such that the procedure is verifiable by all participants in $\mathcal{P}$ and all malicious participants in $\mathcal{M}$ cannot increase their chances of being chosen as workers to more than random chance (i.e. the probability of m malicious participants being selected as workers follows a hypergeometric law $\mathcal{H}(n, \frac{M}{P}, P)$).

Hypotheses:

(h_1) Malicious entities cannot interfere with the block-/transaction-ordering on the ledger. *Rationale: Property guaranteed by the Blockchain*;

(h_2) Malicious entities cannot interfere with whether transactions are published or not on the ledger. *Rationale: It is assumed that the coalition of malicious entities does not have the resources to take control over the blockchain's consensus algorithm (as required by blockchain characteristics defined before)*;

(h_3) If a Cryptographically Secure Pseudo-Random-Number Generator[1] (CSPRNG), a high entropy generator, and a numbering method with a large output space are used to generate the list of workers, malicious participants cannot increase their chances of being chosen as workers by choosing particular IDs. *Rationale: Property guaranteed by CSPRNGs.*

7.4.2 Algorithm

A new phase is added between the referendum's initiation and the referendum vote. Initiation commits a list of P participants and a number n of workers to select. This new phase will consist of the participants choosing to deposit a message on the ledger or not. This creates a sublist of participant IDs on the ledger which will be used to generate a seed number for a CSPRNG. This CSPRNG will in turn sample n workers among the P participants. It is important to note that these committed messages are distinct from the participants' actual votes, which happen in a later phase.

This protocol proceeds as follows (see Figure 7.1 above for reference):

1. The phase begins when the phase-switching deadline q_{0-1} after the referendum's initiation is reached. This happens when the initiator commits the referendum's information on the ledger. These information include:
    - IDs of participants able to vote in the referendum;
    - Number $n_{workers}$ of workers to select;
    - Unaffected share affiliations between participants and workers used in Shamir's Secret Sharing Scheme (i.e. a participant x will divide its secret referendum-vote in the next phase and respectively give each share to the $k^{th}, l^{th}, \ldots$ selected workers);

[1] Cryptographically Secure Pseudo-Random-Number Generator: high entropy generator which resists reverse-engineering, cryptanalysis and passes statistical randomness tests

    - Context and semantics for the referendum;
    - Phase-switching timestamps $\{q_{1-2}, q_{2-3}, q_{3-4}\}$.

2. Each participant decides to commit a message on the ledger or not;

3. The phase ends when the phase-switching deadline q_{1-2} is reached;

4. An ordered list of messages V will then be present on the ledger. A number is computed from V and the ordered list of all referendum participants P. Three main computations can be used to generate this number: the permutation, combination and arrangement numbers of V in regard to P. The first computation uses only the message-list V as randomness input, while both V and P are used for combination and arrangement number generation. Mathematical expressions and output spaces are presented in Table 7.1.
    - The computed number is used as a seed for a CSPRNG. Then, w workers are sampled in the list of participants with this generator;
    - Shamir's Secret Sharing Scheme's shares affiliations between participants and workers for the referendum-vote are then attributed to each selected worker;
    - The protocol can then proceed to the next phase (referendum vote).

Remarks:

- *On multiple committed messages:* only the first message deposited by each participant (if they deposited any) is considered. This prevents seed value manipulations from malicious participants as the phase goes on.

- *On merging worker selection with the voting phase:* as said earlier, the workers are selected during a distinct phase preceding the actual vote. For simplicity, and given worker selection is based on an order of messages, one would intuitively want to perform worker selection during the vote itself. However, this is impossible due to an incompatibility with the voting protocol mechanisms as proposed by Schiedermeier et al. [256]. Indeed, participants vote by committing n distinct shares, encrypted with each worker's public key (so that only that worker can decrypt it with its private key). This means that all workers must be identified prior to the voting phase. Moreover, from a security standpoint, merging the phases reduces the entropy of the number generation, because its entropy comes from both the number of messages and their order. Indeed, honest participants are likely to actually vote, which ultimately reduces the space from which the number is generated.

- *In the absence of committed messages:* a number is generated however many participants commit a message, in particular even when the ordered list of committed messages is empty. In that scenario, using the proposed numbering functions (see Table 7.1), the obtained value would be zero. The absence of commits is therefore not problematic as long as the hypothesis (h_2) defined above holds. If (h_2) is compromised, then the participant list, trusted by the protocol, could be maliciously designed such that no commits (or only malicious commits) make it so enough malicious workers are selected. This would be done by giving malicious participants specific IDs.

- *Message list traversal:* In the expressions of Table 7.1, the list of committed messages is traversed in the anti-chronological order (i.e. the latest message corresponds to index $i = 0$) for additional security. This prevents malicious participants from progressively computing the updates of the seed value as messages are committed. Indeed, using that direction for list traversal, all indices point to a different value each time a new message is committed.

7.4.3 Discussion

Malicious entities cannot interfere with the presence and order of honest message deposits (h_1, h_2). Therefore, a favorable condition for them is to wait until all honest participants have most likely made their choice whether to deposit a message on the ledger. From the malicious perspective, these honest messages constitute a sort of nonce for the seed number. If the initiator is malicious, it might also try to give specific IDs to some malicious participants, in order to have them chosen. Note that in the case of Schiedermeier's protocol [256], IDs are public keys. This renders it complicated to generate a key-pair such that it favors malicious participants. Moreover, ID assignment happens

Table 7.1: Three methods to attribute a number to a list V of comparable non-reoccurring elements (integers, character strings, etc.) knowing its superlist P.

Numbering Method	*Expression*	*Output Space*
Permutation $PN : V \rightarrow \mathbb{N}$	$\sum_{i=0}^{\lvert V\rvert-1}(i! \sum_{k=0}^{i-1} \mathbb{1}_{x_k<x_i})$	$[\![0; \lvert V\rvert! [\![$
Combination $CN : V, P \rightarrow \mathbb{N}$	$\sum_{k=0}^{j(0)-1} \binom{\lvert P\rvert-j(0)+k}{\lvert V\rvert-1} + \sum_{i=1}^{\lvert V\rvert-1} \sum_{k=0}^{j(i)-j(i-1)-2} \binom{\lvert P\rvert-j(i)+k}{\lvert V\rvert-i-1}$	$[\![0; \binom{\lvert P\rvert}{\lvert V\rvert} [\![$
Arrangement $AN : V, P \rightarrow \mathbb{N}$	$\sum_{i=0}^{\lvert V\rvert-1}(\frac{\lvert P\rvert!}{(\lvert P\rvert-i)!} + CN(V,P) * \lvert V\rvert! + PN(V))$	$[\![0; \sum_{v=0}^{\lvert P\rvert} A_{\lvert P\rvert}^{v} [\![$

With: $\mathbb{1}_F = \begin{cases} 0, F \text{ is False} \\ 1, F \text{ is True} \end{cases}$, F a boolean expression. $P = [x_0, .., x_{p-1}]$ a list of elements in ascending order, p the number of participants. x_k is the ID of the k-th participant in the referendum. $V = [y_0, .., y_{v-1}]$ ordered by the order of messages in the ledger, V a sublist of P, v the number of messages. $j : \mathbb{N} \rightarrow \mathbb{N}$ a function such that: $j(i) = k$ iff $y_i = x_k$, i.e. for an ID in V of index i, returns its index in P. $A_n^k = \frac{n!}{(n-k)!}$ is the amount of distinct arrangements of k among n non-reoccurring elements, also called k-permutations of n.

before this protocol's execution, therefore it is only possible to try to maximize chances of being selected given a set of nonces. The greater the numbering function's output space, the lower these chances. Enabling participants to cancel their previous message by committing a new one gives more freedom for malicious entities to adapt to the order of honest participant messages. The security of the seed generation is given by the size of the output space, meaning the possible entropy of the number generation, coupled with mechanisms to hinder malicious behavior adapting as the protocol advances.

Our seed generation method's output space depends on the number of participants ($[\![0; \sum_{v=0}^{|P|} A_{|P|}^{v} [\![$). Therefore, for low participant numbers, entropy might be too low to guarantee pseudo-randomness. A solution to this problem would be to introduce entropy into the system, by considering a value posted in each participant's message. However, this approach introduces possibilities for malicious entities to manipulate the seed value. One must keep in mind that the size of the output space is a sum of factorials depending on the number of participants: for 10 participants, there are 10^6 possibilities, but 10^{158} for 100 participants. To boost the number of possibilities for a low number of participants, multiple message commits per participant can be handled for number generation. If n message commits are allowed per participant, then each message (containing a number between 1 and n) would be considered as that of an artificial participant in a list of $P' = n * P$ participants. Note that the actual proportion of malicious participants (real and artificial) remains unchanged, but the output space for generating numbers greatly increases with P' (see the AN function's output space in Table 7.1).

7.5 Proposal 2: Verifiable Social-Graph-Aware Worker Selection

Instead of randomly choosing the workers among the voters, this next solution uses social graph data in order to select them. This process is done with a community detection algorithm. First, we compute the communities of the graph. We then decide how many workers will be selected from each community. Finally, we select the workers from the communities whilst trying to maximize their distances to other workers.

If we are able to distinguish several communities in a social graph representing interactions between the candidates of the protocol, we can gather new information to achieve worker selection. Indeed, two participants close to each other in the same community are likely to know each other and have social interactions [132], in other words, should they also be malicious, they are likely to collude. This assertion leads us to a new algorithm: maximizing distances between workers inside each community and in the whole graph to minimize interactions.

We assume that the participants have access to an anonymized social graph of the participants. This is a simplification of the context for experimental purposes, where privacy of participants is not required. Indeed, a main purpose of this solution is to compare our random worker selection to a selection taking advantage of the social structure of participants. If this graph is available and the number of workers to be selected is known, then this protocol is verifiable by all participants. If such a graph is not available, we propose to use our first solution (section 7.4), which is fully decentralized. In section 7.9, we also discuss some potential alternatives to the centralized availability of the social graph.

In any case, like our first solution, this approach is independent from the voting protocol that we use to contextualize them. Therefore, it can be useful in other contexts, where such a graph is available and privacy is not a concern. For example, in the context of a referendum in a company, the organization chart, which is open and not privacy-sensitive information inside the company, can be used as an approximation of the interactions between employees, and as such, as an approximation of the social graph inside the company. In this specific context, an ability to punish detected malicious activities may be wanted. Otherwise, the organization chart may be modified to reduce information identifying specific employees. Note that it is possible to enable linking workers to real identities, while preserving voter secrecy, by maintaining two lists of participant identities, one for voting and one for (potentially) working. Geographical clustering of voters, such that a sufficient k-anonymity is guaranteed (for example, in USA: federal, state, county or municipality levels), could also be used to approximate the overall social graph from which workers can be selected.

7.5.1 Context

Consider the existence of a "real" social graph representing the relationships between vote participants in real life. This abstract structure is to be distinguished with graphs obtained from social networks like Facebook, which are created based on user interactions on that service. As such, the latter are an approximated representation of the complete and theoretical "real" graph, with missing or added nodes and edges. Nonetheless, experimentally, we will tolerate this approximation and use Facebook social graphs as datasets (see section 7.7). In our context of collusion-resistance, the proximity between nodes is considered to represent their likeliness of complicity, meaning workers are more likely to collude together if they are close in the social graph. Therefore, in that context, selecting workers such that they are distant from each other in the social graph would reduce the probability of collusion.

Inputs: Let G be an undirected unweighted social graph with $G = (V, E)$. The same reasoning can be applied to a weighted graph. V a set of vertices representing each referendum participant. Each vertex v possesses an attribute which is the participant's ID. Let E be a set of edges, two vertices are connected with an edge e if the corresponding participants mutually and directly know each other. Let $n_{workers}$ be the number of workers which shall be designated among the participants.

Output: Let W be a set of participant IDs which shall act as workers in subsequent phases.

Objective: Choose W such that workers don't know each other well enough to form a coalition capable of invalidating the results of the referendum. This means determining W a subset of V, such that W contains less than t mutually cooperating malicious nodes out of $n_{workers}$ nodes, i.e. maximize the social distance between all workers.

Hypotheses: Concerning community detection, we consider the following hypotheses:

- Two participants know each other less the longer the shortest path between their respective vertices is. This hypothesis is based on heuristics used in topology methods for link prediction problems in (social) graphs, e.g. common neighbors [132];
- Participants from distinct communities know each other less, even more so if those communities don't communicate directly with each other (paths between members of the two communities go through other communities).

7.5.2 Algorithm

Our main Algorithm 7.2 can be divided in three steps:

1. *Community Detection:* From a social graph, we apply a clustering algorithm to find groups of nodes (see subsection 7.5.3);

```
Algorithm 1: Social-graph-aware worker selection
  Data: $G$: unoriented graph $(V, E)$, with $V$ a set of vertices representing each participant,
        $E$ a set of edges such that $v_i$ and $v_j$ are adjacent if participants i and j know each
        other; $n_w$: amount of workers to place in $G$
  Result: $W$: subset of vertices $W \subseteq V$, the set of chosen workers
1 $P, C \leftarrow detectCommunities(G)$;
  /* $P$ partitions $G$ in $n_c$ clusters and attributes a cluster ID
     $c_k \in V_C = \{i \in [\![1; n_c]\!] | c_i\}$ to each vertex in $V$, such that
     $P : P(v \in V) = c_k \in V_C$                                          */
  /* $C = (V_C, E_C)$ an undirected graph of $n_c$ communities.  $c_i$ and $c_j$ are
     connected with an edge $e_{c_{i,j}} \in E_C$ if one or more vertices in
     $\{v \in V | P(v) = c_i\}$ are adjacent in $G$ to one or more vertices in
     $\{v \in V | P(v) = c_j\}$.                                             */
2 $N \leftarrow quantifyWorkersPerCommunity(G, P, C, n_w)$;
  /* $N = \{i \in [\![1; n_c]\!] | n_i =$ number of workers in $c_i\}$          */
3 $W \leftarrow$ empty set;
4 for $c_i \in V_C$ do
5     $W_i \leftarrow selectWorkers(G, P, c_i, n_i)$;
6     $W \leftarrow W \cup W_i$;
7 end
8 return $W$;
```

Figure 7.2: Social-graph-aware worker selection

2. *Quantifying the number of workers to be selected in each community:* Depending on the number of communities $n_{communities}$ and total number of workers $n_{workers}$, we will compute the number of workers n_i to be selected in each community c_i (see Algorithm 7.3):

   → $n_{communities} = n_{workers}$: one worker per community;

   → $n_{communities} > n_{workers}$: one worker per community in distant communities;

   → $n_{communities} < n_{workers}$: distribute multiple workers amongst communities using a criterion.

3. *Selection of workers:* According to the number of worker n_i, we will then choose which participants will become workers for each community and maximize their distances to each other (see Algorithm 7.4).

7.5.3 Step 1: Community Detection

The choice of the community detection method will greatly impact the worker selection algorithm and the collusion resistance. Indeed, our main hypotheses for workers selection and assignment are based on inter- and intra-communities interactions. We distinguished several criteria to choose our clustering algorithm:

- deterministic outcome (no random operations to generate clusters) required for reproducibility and verifiability by participants;
- linear or near-linear time-complexity on the number of vertices or edges;
- each vertex is assigned to a unique cluster (i.e. no multi-community detection);
- the number of detected communities depends only on the graph structure (i.e. no need of parameters to set or vary the number of communities found).

Considering the above criteria as a filter for algorithms already implemented in the graph library that we use (igraph [4]), we retain the multi-level community detection by Blondel et al. [54] as our community detection algorithm.

7.5.4 Step 2: Quantifying the Number of Workers to be Selected in Each Community

This step consists of determining the number of workers which shall be selected in each community in Step 3 (subsection 7.5.5), considering the total number of workers $n_{workers}$, the number of communities n_c, and a distribution criterion.

Algorithm 2: Quantifying the number of workers to be selected in each community

```
   Data: $G$, $P$, $C$, $n_w$ as defined in algorithm 1
   Result: $N = \{i \in [\![1; n_c]\!] | n_i = \text{number of workers in } c_i \in C\}$, $n_c$ the number of
           communities in $C$
1  $N \leftarrow \{i \in [\![1; n_c]\!] | n_i = 0\}$;
2  if $n_c = n_{workers}$ then
3      $N \leftarrow \{i \in [\![1; n_c]\!] | n_i = 1\}$;
4  else if $n_c > n_w$ then
5      $C_w \leftarrow spreadVertices(C, n_w)$;                /* Algorithm 3, $C_w \subset C$ */
6      $N \leftarrow \{i \in [\![1; n_c]\!] | n_i \begin{cases} 0, c_i \notin C_w \\ 1, c_i \in C_w \end{cases} \}$;
7  else
8      $N \leftarrow criterionDistrib(G, P, C, n_w)$;
9  end
10 return $N$;
```

Figure 7.3: Quantifying the number of workers to be selected in each community

Hypotheses

We consider the following hypotheses:

1. Given a shortest path length d in the graph, two participants in the same community distanced by d know each other more than two participants from distinct communities with that same distance d between them.

2. Sparse graphs are better suited to house more workers than dense one. *Rationale: there are more shortest paths with values of 2 or more, meaning participants of that same cluster know each other in varying degrees of separation.*

3. Placing workers evenly across the graph (i.e. all workers distant by 3 more edges versus creating small clusters of less than t workers) gives better guarantees for non-collusion. *Rationale: each worker does not know its nearest worker-neighbors that well.*

Algorithm

If the ratio $\frac{n_{workers}}{n_c}$ is less than one, then workers should be placed in communities distant from each other. Otherwise, all communities shall contain a worker before assigning multiple workers to a community (to take advantage of hypothesis 1). If there are more workers than communities, the graph structure should be used to infer how many workers should be contained in each community (see algorithm 7.3). In this last case, the number of workers in each community is given by the pro-rata of the number of vertices in a community to the total number of vertices in the graph, e.g. a community comprising 10% of the vertices will receive 10% of the workers.

7.5.5 Step 3: Worker Selection

Once the number of workers per community has been determined, those numbers must be selected from each community's subgraph (see algorithm 7.4). Finding the optimal solution to maximize the distances between worker nodes is time-exponential in the number of nodes. We approximate this solution by iteratively adding nodes to the output set, using breadth first searches and maximizing different selection criteria. Time-complexity achieves $O(n_i^2 * n + n_i * e)$ this way, with n nodes and e edges in the community c_i's subgraph, n_i the number of workers to place in c_i.

Distancing Criteria

The main criterion for worker selection is maximizing the distance with the nearest worker. In case of multiple choices, a secondary criterion is used to decide: (MSR) Maximizing the Sum of a Reward function $r : d \rightarrow reward$:

$$r : d \rightarrow \begin{cases} 0, & d \in [\![0; 1]\!] \\ 1, & d = 2 \\ 2, & d \in [\![3; +\infty[\![\end{cases}$$

```
Algorithm 3: Select spread out vertices in a graph
  Data: $G$: unoriented graph $(V, E)$, $V$ the set of vertices, $E$ the set of edges; $n$: amount of
        vertices in $V$ to select; $V_{av}$: $V_{av} \subset V$, $V_{av} \leq |V| - n$, a set of vertices to avoid
        (empty by default)
  Result: $V_S \subseteq V, |V_S| = n$, the set of selected vertices
  $n_r \leftarrow n$;                                  /* remaining vertices to select */
  if $|V_{av}| = 0$ then
      $V_S \leftarrow$ {selectInitialVertex(G)};
  else
      $V_S \leftarrow V_{av}$;
      $n_r \leftarrow n_r - 1$;
  end
  $M_{sp} \leftarrow BFS(V_S, V)$;   /* Breadth-first-search distances from vertices in $V_S$ to
    vertices in $V$ into a $(|V_S|, |V|)$ matrix.  If the graph is weighted, then BFS
    can be replaced with an adequate algorithm.  */
  for $i \in [\![1; n_r]\!]$ do
      $v_s \leftarrow selectVertex(V, M_{sp})$;
      $V_S \leftarrow V_S \cup \{v_s\}$;
      $M_{sp} \leftarrow (M_{sp}|BFS(\{v_s\}, V))$ ;
  end
  $V_S \leftarrow V_S - V_{av}$;
  return $V_S$;
```

Figure 7.4: Select spread out vertices in a graph

MSR's heuristic takes into account the fact that the primary criterion (maximizing the distance to the closest worker) has operated a pre-selection on the candidate nodes. After the primary criterion, we know that there will be some workers who are close to the candidate. Knowing that, what we seek is to have the highest number of faraway workers (distances of 3 or more). Which is why workers distant by 3 or more yield the reward of 2, while those distant by 2 yield half, and adjacent workers give no reward.

Random choice of the next worker among candidate nodes also gives good results in spreading workers on the graph. However, we require using deterministic methods for verifiability purposes.

7.6 Experimental Protocol

We will now present the experimental protocol which we will use to evaluate the implementation of our worker selection methods: verifiable random worker selection (section 7.4) and community detection-based worker selection (section 7.5). The objective of this protocol is to build an environment to quantify and analyze collusion resistance in the protocols that we proposed.

7.6.1 Context

Given an execution of either method placing n workers on a social graph, the probability of collusion will be quantified by the size of the largest clique of workers, distant from each other by at most $k \geq 1$. This metric is time-exponential on n, which means experiments on high n values will have to be done with fewer protocol executions. We assume for our experiments that this largest clique of workers of size m will try to collude. We could have chosen to tag a subset of graph nodes as malicious, then measure the size of the largest clique of malicious workers. However, this second technique requires additional assumptions on how malicious workers are distributed on social graphs, which we have chosen to avoid.

We have defined in section 7.3 the threshold $t_{max}(n) = \lfloor (1+\sqrt{5+4n})/2 \rfloor$ under which a number of m colluding malicious workers still upholds our protocol's security properties. Given this metric m and n workers placed on a graph, we consider that collusion is possible if $m \geq t_{max}(n)$. Given a set of protocol executions selecting a number of n workers, we also compute a confidence interval $[0; m_{max}]$ on the distribution of obtained m values. A protocol is defined as collusion-resistant with a confidence c for a given number of workers n, if the probability that $t_{max}(n) \notin [0; m_{max}]$ is c. Our experimental protocol aims to answer the following question: does random sampling of workers among participants give a high enough confidence that no collusion will be possible between those workers?

We will fix the confidence level for our experiments to 95%. An alternative would have been to compute a p-value of each set of executions for n selected workers. In our case, this p-value would correspond to the probability that we erroneously consider that there is a collusion attack, meaning the probability that no collusion takes place. A high p-value would mean a low probability of collusion. However, due to the reduced number of executions for high n values, this p-value would not be significant, given the low number of considered experiments. Figure 7.5 describes our experimental protocol's workflow.

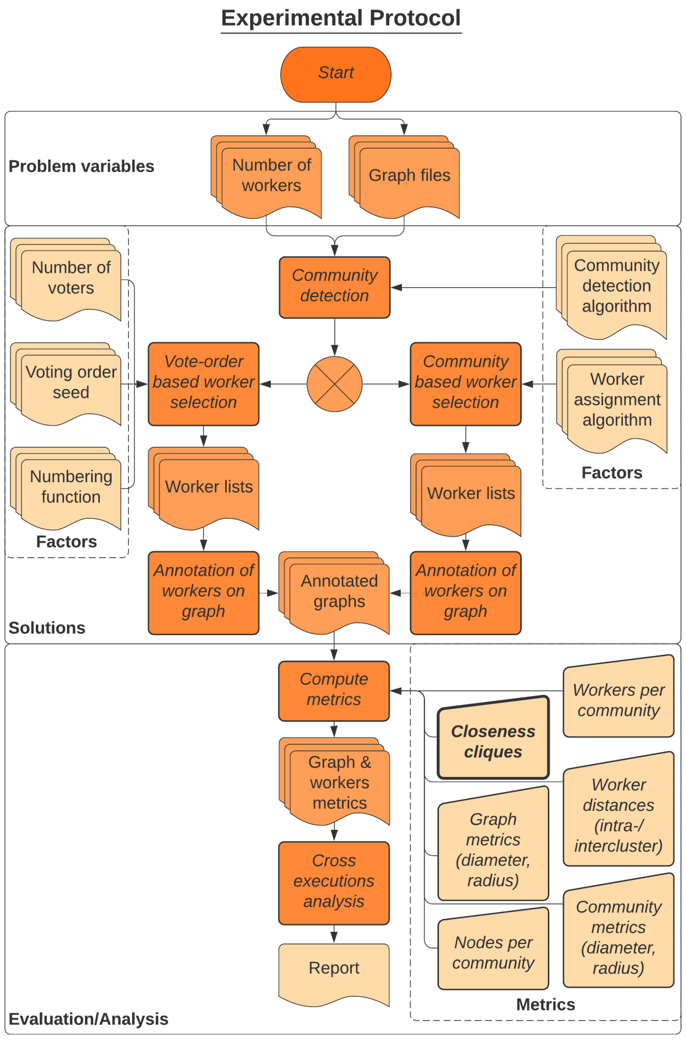


Figure 7.5: Process diagram for the experimental protocol, given a set of problem variables comprising social graphs and amounts of workers to place.
Experimentation is divided into a solutions-processing phase followed by metrics computation and evaluation. On the sides of the solution phase, factors for a given algorithm are presented. Our main metric is emphasized in bold.
This process is fully automated until "cross executions analysis" excluded.

The effective parameters which are the number of workers to select were chosen based on 10 points of an exponential growth from 10 workers to $\frac{1}{5}$ of the size of the graph (in the case of facebook_combined[2]: 807 out of 4039 nodes) with the logspace method from the Python NumPy library[3]. These parameters will be tested on 100 executions. Due to the time-exponential complexity on the number of workers, experiments for $n \in \{1000 + 200 * k | k \in [\![0; 10]\!]\}$ will be run on 5 executions.

[2]See section 7.7 for the dataset introduction
[3]Python NumPy logspace function documentation at https://numpy.org/doc/stable/reference/generated/numpy.logspace.html

Table 7.2: Problem instances or Experimentation inputs

Problem Variables	*Values*
$n_{workers}$	$\{10, 16, 43, 70, 114, 186, 304, 495, 807\}$ $\bigcup\{1000 + 200 * k \mid k \in [\![0; 10]\!]\}$

7.6.2 Randomly Sampling Nodes on a Graph: Verifiable Random Worker Selection Simulation

We substitute the Verifiable Random Worker Selection with a simulation protocol for the experiment's purpose. This does not have an impact on the results since the simulation keeps the same concept of randomness and the output remains an ordered list of participant IDs.

In lieu of this specific approach, the more general problem of random sampling of workers in a given graph will be evaluated. This gives us an expectable behavior and feasibility approximation for our random number generator protocol. Those solutions being non-deterministic, they will be considered feasible in practice for a given graph and a number of workers n, if the executions' largest clique-size 95%-confidence intervals satisfy the problem's constraints.

In practical terms, this considers a scenario where all P participants are potentially malicious, but a subset of workers can only successfully collude together if they form a clique larger than the security threshold t in the graph.

7.6.3 Graph and Worker Selection Metrics

Our main metric is the size of the largest cliques of workers distant of at most k, k in $[\![1; 3]\!]$. It gets computationally too expensive for $k > 3$, for less added insight, as 93% of communities found in the graph have diameters of 5 or less for the multilevel algorithm, the facebook_combined graph itself being of diameter 8. The best method should minimize the size of the largest clique given $n_{workers}$. For behavior analysis purposes, additional metrics will be measured: 1) Community subgraph diameter and radius; 2) Number of nodes in communities; 3) Number of workers in communities; 4) Distances between workers (intra- and inter-community), mainly on inter-worker distances less or equal to 2. The best method should minimize the amount of workers directly adjacent or separated by one other participant.

7.7 Chosen Dataset and Tools

7.7.1 Dataset

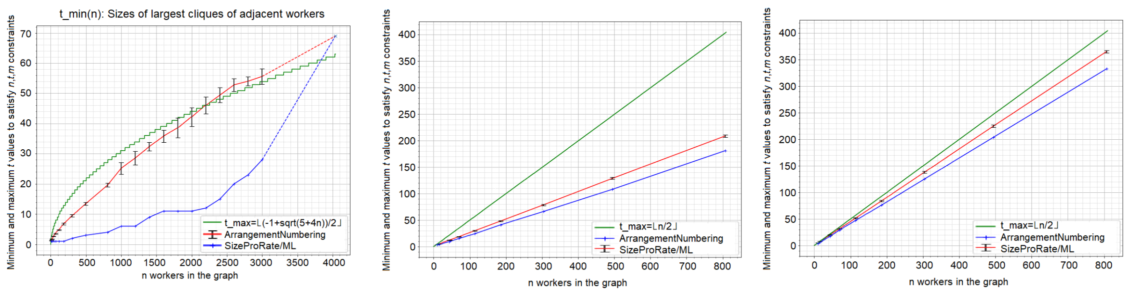
We used the facebook_combined [5] dataset for the experimental evaluation. Some of the properties of the facebook_combined dataset are as follows: 1) Undirected edges (symmetrical relationships); 2) Comprises 4,039 nodes and 88,234 edges; 3) "Community detection friendly": most clusters should be detectable by humans on the graph. Most members of these clusters should be adjacent to multiple other members of the same cluster. Counter-example: streaming website Twitch's network [6], where some individuals have very high degrees, while their adjacent nodes are mostly not connected to each other.

7.7.2 Tools

We used the following tools for the experimental evaluation. *Graphing tools:* igraph (data structures and graph methods) [4]; graph-tool (visualization) [7]. *Exploration and experimentation:* Python Jupyter [3]. *Workflow and reproducibility management:* A framework developed by Matthieu Bettinger [50] (provided with the rest of the source code [51]); Python Jupyter [3].

7.8 Evaluation and Analysis

This part aims to present and analyze the results from the execution of the workflow presented in Figure 7.5, i.e. the step "cross-executions analysis" in the process diagram.



(a) Sizes of largest cliques of adjacent workers (b) Size of largest cliques of workers distant of at most 2 (c) Size of largest cliques of workers distant of at most 3

Figure 7.6: Lower and upper bounds for t in t-n Shamir Secret Sharing Scheme. $t_{max}(n)$ is obtained when removing the problem's strictest constraint on t (Equation 7.1c, see also section 7.3).

7.8.1 Random Nodes Sampling on a Graph

We plot the lower and upper bounds for t in the $t - n$ Shamir Secret Sharing Scheme in Figure 7.6. The inequalities 7.1a through 7.1c from section 7.3 give the maximum values t_{max} given n. Meanwhile, the metric we evaluate empirically, that is the size of the largest clique of workers distant of at most k (with k increasing from 1 to 3 in Figures 7.6.a, 7.6.b and 7.6.c), gives the lower bound t_{min}. This is done for our $ArrangementNumbering$ and $SizeProRata/ML$ (Multi-Level) worker selection algorithms, presented respectively in sections 7.4 and 7.5. A solution is viable for a given n if $t_{max} > t_{min}$.

$ArrangementNumbering$'s curve is given with a 95% confidence interval (error bars). For $n \leq 807$, 100 executions were done for each point on $ArrangementNumbering$, 5 executions for $n > 807$, due to the metric's exponential complexity on n. No error bars are necessary for $SizeProRata/ML$, as the algorithm is deterministic.

Intuitively, a method which has access to the information upon which a metric is computed will perform better on that metric than a method without that information. In our case, the metric being the size of the largest clique of workers, our graph-aware proposed solution should perform better than our random worker selection protocol. This observation can be confirmed by the graph representing the cliques' size threshold for the $t - n$ Shamir Secret Sharing Scheme (see Figure 7.6). Undeniably, the $SizeProRata/ML$ curve ("ML" stands for multi-level community detection), our graph-aware solution, shows considerably better results than the random selection with $ArrangementNumbering$. Until 45% of participants are workers, $ArrangementNumbering$ is closely under the maximal threshold curve : there is a confidence of 97.5% that the size of largest cliques is under t_{max} until 1800 workers. $SizeProRata/ML$ is remarkably more efficient and its heuristics only begin to lose efficiency at around half of the voters as workers. When the number of workers reaches the number of voters, the $ArrangementNumbering$ and $SizeProRata/ML$ curves will intersect in a final point. This intersection corresponds to the largest clique size (in this case 69 for 4039 nodes) when all the voters are workers. Computations for over 3000 workers were not executed, because of the largest clique search algorithm's time-complexity. Only the final intersection point was computed. Dotted lines show the expected evolution of both curves between 3000 workers and the endpoint.

However, considering a worker threshold of $\frac{1}{5}$ of the participants (here 807 among 4039), which is already a high number of workers, the two workers selection methods stay under the t_{max} curve determined by our problem's constraints.

7.8.2 Results Under Relaxed Constraints

As defined in section 7.3, our main constraint on the upper bound of the security threshold t is due to the checksum verification which implies $n - t^2 > m$ (Inequality 7.1c). Should the constraint of Inequality 7.1c be removed or be absent (i.e. in another protocol without this checksum mechanism), only constraints linked to Shamir's Secret sharing scheme would remain (Inequalities 7.1a and 7.1b, namely $t > m$ and $n - t > m$), which give an upper bound for t: $t_{max} = \lfloor \frac{n}{2} \rfloor$. In practical terms, as $n - t \geq n - t^2 > m$ for $t \in \mathbb{N}^*$, removing Inequality 7.1c enables tolerating a greater proportion of malicious workers, for a given number n of chosen workers.

If we were to consider this new upper bound, we could compare curve behaviors of both our selection methods with that new boundary curve, for cliques of workers distant of at most 2 (see Figure 7.6.b), resp. 3 (see Figure 7.6.c). These methods' curves greatly surpass the more constrained version of t_{max}. In case of a maximal distance of 2 resp. 3, we can see that both methods representing t_{min} grow linearly with a slighter slope than the boundary curve representing t_{max} (*SizeProRata/ML*: 0.23 resp. 0.41; *ArrangementNumbering*: 0.26 resp. 0.43; $t_{max}(n)$: 0.5). Therefore, should the $n - t^2 > m$ constraint be lifted whilst maintaining the protocol's properties, better collusion-resistance insights can be obtained. Indeed, we would know that under these new constraints, it is unlikely that a clique of workers distant of at most 3 would be larger than t_{max}.

7.9 Discussion

7.9.1 Clique Sizes in Social Graphs

When a social graph increases in size, how do graph properties like the size and the number of the largest cliques scale? The denser a given graph, the larger the probable size of the largest clique and the number of cliques of a given size. If the proportion of nodes in big cliques compared to the global graph gets higher, random sampling of nodes will more and more likely occur in those cliques. The same reasoning holds true for big communities, a relaxed concept of cliques of nodes. Johan Ugander et al. described the structure of Facebook's social graph in 2011 [277].

For both the U.S. and global friends networks on Facebook, around 90% of users had less than 500 friends and 1% had more than a thousand friends (maximum number at around 5000 friends). They used the degeneracy metric in their analyses, which corresponds in "an undirected graph G [to] the largest k for which G has a non-empty k-core. Meanwhile, the k-core of a graph G is the maximal subgraph of G in which all vertices have degree at least k". A k-core corresponds to a $k+1$-clique if its size is $k+1$. This also means that k-cores are sets of nodes which may contain cliques of size less or equal to $k+1$. Therefore, degeneracy provides an upper bound for the size of the largest clique in the graph. Their findings use the degeneracy for nodes of a certain degree, i.e. for a given person, the maximum k friends which also know $k-1$ other friends of that same person. Degeneracy grew monotonously with the node degree. The maximal degeneracy for the 95th percentile was of around 200 for a degree of 5000. This would mean that 200 friends of that person knew 199 other friends of that person.

What interests us more is that a clique containing that person would be of size lower than 200. Their analysis was carried out on active Facebook users (users, with at least one friend, who logged in at least once in the last month prior to the analysis), representing a graph with 721 million nodes. This result on such a large graph gives us reassurance on the evolution of the size of the largest clique as the graph gets larger. Indeed, for such a graph size, $t_{max} = 26851$, which is two orders of magnitude bigger than the upper bound for the largest clique's size. In fact, a graph of size 39,799 (10x the size of the graph we used) would still tolerate a largest clique of 200 nodes whilst ensuring collusion-resistance.

7.9.2 The Feasibility of Random Worker Selection

A determining factor on whether random selection of workers is feasible with a low collusion probability in a given graph is the size of the graph's largest clique (see Figure 7.6 for reference). If that size is significantly lower than the upper bound for the $t - n$ Shamir Secret Sharing Scheme's t value obtained through the problem's constraints, then there is a low collusion probability, i.e. high collusion resistance, for any number of workers. However, if the size of the largest clique is close to or greater than $t_{max}(P)$, with P the total number of nodes in the graph, then there exists an upper bound n_{max} for n where, for $n \geq n_{max}$, $P(|Clq_{max}| > t_{max}) > 5\%$, with Clq_{max} the largest clique's set of nodes for n workers.

If the graph is unknown or unavailable, then this upper bound n_{max} can be approximated through other approaches. Insights about the voting population's social structure, for example insights on its density, the (expected or known) size of some communities among participants, can help in estimating n_{max}. Without such graph knowledge, then one can use system constraints (Inequalities 7.1a-c) and the hypergeometric distribution followed by this random selection (as presented in section 7.4). By estimating a certain proportion of M malicious participants among the total P, one can get all numbers of workers n such that system constraints are verified.

7.9.3 The Feasibility of Relaxed Graph-Aware Methods

Would less informed knowledge about the graph be sufficient in order to give high collusion-resistance confidence? For low values of $n_{workers}/n_{participants}$, the size of the largest clique of workers seems to increase linearly with the number of workers (see Figure 7.6). That function's slope is initially steeper than the one of the linear function passing through the point corresponding to the largest clique in the graph, then the slope gets slighter in order to end on that same point. It would be of interest to investigate if that observation still holds on other social graphs. In that case, because t_{max} follows a square root-shaped function, there exists a range of low values of n, where t_{max} is greater than t_{min} (size of largest clique with 95% confidence).

Other criteria altogether, not using a graph, could be used to help determine a number of workers n: 1) Using the curve obtained by intersecting the hypergeometric law's Cumulative Distribution Function $P(n, m)$ with a plane of probability p (e.g. 95%) as defined in subsection 7.4.3. This method requires quantifying an expected total number of malicious participants M; 2) Optimizing n on the criterion of the amount of needed messages. Indeed Shamir's Secret Sharing Scheme requires dividing each one of the P participant's vote in n shares, one per worker. This means there will be $n * P$ messages placed on the distributed ledger during the referendum-vote. For scalability purposes, the number of workers should be kept as low as security criteria permit it.

7.9.4 The Feasibility of Graph-Aware Methods

Should a social graph of participants be available for a given referendum, where should it be stored? If only the referendum initiator has access to it, should they be malicious, nothing prevents them from not using it altogether in designating workers. Forcing him to use it could be done by forcing him to provide a proof that the result was obtained through the algorithm.

Let us now consider a graph annotated with participant IDs on nodes. The initiator can use it or transmit it to other malicious entities to violate participant anonymity, through graph inference re-identification. However, knowing only the unannotated graph may provide lower and upper bounds for n in the same way our experiments did (see Figure 7.6). Anonymity could be maintained in this case. If the graph is public, then it becomes easier to ascertain whether the initiator used the algorithm, for example through Smart Contracts (Ethereum)[287]. However, an annotated graph would again be at risk of participant re-identification. If we were to divide the graph among participants in order to decentralize tasks, we would need workers and would therefore have the same problem to select those.

Another variant would be to use a decentralized social graph, with participants knowing only their "friends" on the now implicit social graph (plus some strangers to avoid re-identification if some nodes have a low degree). A decentralized algorithm should then be designed to select workers under those constraints.

7.10 Conclusion

In this work, we proposed two solutions to provide better collusion-resistance in distributed protocols where a subset of workers needs to be selected from the set of participants:a verifiable random worker selection based on decentralized computation of a random seed, as well as a selection based on community detection in social graphs. Firstly, we used the blockchain's immutability and ordering to design a collusion-resistant decentralized protocol to randomly select workers. Secondly, we considered a social graph representing participants and proposed an algorithm to distance workers from each other in the graph. Based on a social graph and our problem's constraints, we computed the size of the largest clique of workers to evaluate the number of workers' bounds for which our solutions were resistant to collusion with high confidence.

Both approaches provided ranges of numbers of workers satisfying the constraints. The decentralized random worker selection works from low numbers of workers to an upper limit which depends on the size of the graph's largest clique. As expected, the method taking advantage of the graph's structure provides better results: it distances workers better for a wider range of numbers of workers.

As discussed, an interesting topic for future work would be to analyze in depth the impact of the social graph structure on the protocol's resistance to collusion. An equally important topic would be to fully decentralize the method based on the graph structure and community detection. This solution should ensure privacy for participants, be transparent to all and verifiable by all, whilst preserving the demonstrated collusion-resistance properties.

Part III

Privacy-Preserving Message Routing

Chapter 8

A Study of the Unwillingness of Nodes to Participate in Mobile Delay Tolerant Network Routing

Message routing in mobile delay tolerant networks inherently relies on the cooperation between nodes. In most existing routing protocols, the participation of nodes in the routing process is taken as granted. However, in reality, nodes can be unwilling to participate. We first show in this work the impact of the unwillingness of nodes to participate in existing routing protocols through a set of experiments. Results show that in the presence of even a small proportion of nodes that do not forward messages, performance is heavily degraded. We then analyze two major reasons of the unwillingness of nodes to participate, i.e., their rational behavior (also called selfishness) and their wariness of disclosing private mobility information.

Our main contribution in this work is to survey the existing related research works that overcome these two issues. We provide a classification of the existing approaches for protocols that deal with selfish behavior. We then conduct experiments to compare the performance of these strategies for preventing different types of selfish behavior. For protocols that preserve the privacy of users, we classify the existing approaches and provide an analytical comparison of their security guarantees.

This chapter is an adapted version of the article: "An Investigation on the Unwillingness of Nodes to Participate in Mobile Delay Tolerant Network Routing." J. Miao, O. Hasan, S. B. Mokhtar, L. Brunie, and K. Yim. *International Journal of Information Management (Elsevier).* 2013. Vol. 33, no. 2, pp. 252-262. This work was carried out in the context of the Ph.D. of J. Miao, co-supervised with L. Brunie and S. B. Mokhtar.

8.1 Introduction

The heavy utilization of mobile devices with short-range networking interfaces, such as smart phones and personal digital assistants, has led to the emergence of a new type of *opportunistic* networks called Mobile Delay Tolerant Networks (MDTNs). MDTNs are constructed by the (intermittent) connection of co-located mobile devices. Contrary to Mobile Ad-hoc NETworks (MANETs) [252], in MDTNs, a complete routing path between two nodes that wish to communicate cannot be guaranteed [109]. The applications developed for these networks are necessarily geo-localized with no critical time constraints (e.g., advert dissemination, recommendation of points of interest, asynchronous communication). A number of networking scenarios have been categorized as MDTNs, such as Vehicular Ad-hoc NETworks (VANETs) [176], Pocket Switched Networks (PSNs) [150], etc.

Due to the frequent and long-term network partitions that characterize MDTNs, message delivery is considered as one of the major challenges in these networks. In order to deal with the lack of end-to-end connectivity between nodes (i.e., mobile devices), message routing is often performed in a "store-carry-and-forward" manner [109], in which a message is stored by intermediary nodes and forwarded to nodes closer and closer to the destination until it is eventually delivered or it expires. Therefore, message routing in MDTNs inherently relies on the cooperation between nodes.

In the literature, most of the existing routing protocols in MDTNs explicitly or implicitly assume that the nodes in a network are willing to relay messages for others. Unfortunately, reality is different. Indeed, first, as it has been previously demonstrated in the literature, collaborative systems are subject to rational behavior (also called selfish behavior). MDTNs are particularly suited for exacerbating such behavior due to the resource constraints of mobile devices (e.g., battery, memory and bandwidth) [240]. A second reason that leads to the unwillingness to participate in MDTN routing is the users' wariness of disclosing private information (e.g., identity, location, message content). The main contribution of this work is to survey the existing related research works that overcome these two issues.

The remainder of this chapter is organized as follows. We first analyze the impact of the unwillingness of nodes to participate through a set of experiments in Section 8.2. We then classify selfish behavior, and summarize the impact of selfish behavior on routing performance in Section 8.3. We then investigate different strategies for preventing selfish behavior in Section 8.4. This is followed by an experiment to compare the performance of different strategies in Section 8.5. In Section 8.6, we discuss and classify the privacy concerns that users face in MDTNs. We then investigate different privacy-preserving protocols in Section 8.7. The privacy-preserving protocols are then compared in Section 8.8. Finally, we conclude this work in Section 8.9.

8.2 Impact of the Unwillingness to Participate in MDTN routing

In order to evaluate the impact on performance of the unwillingness of a proportion of nodes in the network to participate in the routing of messages, we performed the following experiment. We ran one of the most efficient routing protocols in DTNs, i.e., the Binary Spray and Wait [268] algorithm. In this algorithm, the source node holds a given number of copies of the message it wants to send. Each time it encounters another node, it hands over half of the remaining copies it holds, until it does not have enough copies to send. Similarly, if a node has more than one copy of a given message, it hands over half of the message copies to the encountered nodes, and so forth until the message reaches the destination. In this experiment we injected a proportion of nodes that are not willing to participate in the routing process and analyze their impact on the delivery ratio. We considered two types of behaviors for non-participating nodes, i.e., nodes that explicitly refuse to participate (referred to as "Non-forwarding" in the experiment results) and nodes that accept to receive messages but eventually drop them instead of forwarding them (referred to as "Dropping" in the experiment results). The experimental settings we used for this experiment are the same as those described in Section 8.5. Results, depicted in Figure 8.1 show that in presence of non-participating nodes, the delivery ratio is heavily impacted, especially if nodes do not explicitly declare themselves as non-participating (i.e., the Dropping curve in the graph). Note that in presence of 100% of non participating nodes, the delivery ratio drops to 40%, which represents the situations where the source node directly delivers the message to the destination node.

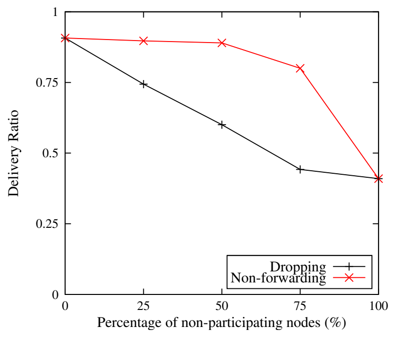


Figure 8.1: Impact of non participating nodes in MDTN routing

Our aim in this work is to understand the reasons why a node may be unwilling to participate in an MDTN routing protocol and survey the related research contributions to deal with this issue. We identify two major reasons, i.e., nodes' selfishness (presented in sections 8.3, 8.4 and 8.5) and their wariness of disclosing private information (presented in sections 8.7 and 8.8).

8.3 Selfishness

In this section, we first develop a unified view of the classification of selfish behavior. We then discuss the methodologies utilized for investigating the influence of selfish behavior on the performance of routing protocols. Finally, we highlight the performance degradation caused by selfish behavior.

8.3.1 Classification of Selfish Behavior

Recent years have seen considerable research works addressing the issue of selfish behavior in DTNs [187, 196, 299]. Traditionally, most works consider selfish behavior as the unwillingness of a single node to relay the messages of all other nodes in order to conserve its limited resources. Nevertheless, people in real life (i.e., the carriers of mobile devices) generally do not act alone, but tend to belong to communities [152]. In an alternative type of selfishness, a node that belongs to a community is willing to relay messages for the nodes within the same community but refuses to relay messages for the nodes outside its community. For this reason, selfish behavior is classified into two categories: *individual selfishness* and *social selfishness* [184].

Moreover, in the literature investigating the impact of selfish behavior on routing performance [165, 231], the authors generally consider the following two types of selfish actions: *non-forwarding of messages* and *dropping of messages*. Non-forwarding of messages means that a node refuses to relay messages for the nodes towards which it is selfish. Dropping of messages means that a node agrees to relay messages for the nodes towards which it is selfish, but it drops the messages after receiving them.

From the above description, we can see that there are two classifications of selfish behavior from different aspects. In this work, we develop a unified view of the classification of selfish behavior. We term the two aspects of the classification as *collusion* and *non-cooperation*. From the viewpoint of collusion, selfish behavior can be classified into two categories: individual selfishness and social selfishness. From the viewpoint of non-cooperation, selfish behavior can be classified into two categories as well: non-forwarding of messages and dropping of messages. The reader is requested to refer to Figure 8.2 for an illustration of the unified view of the classification. To the best of our knowledge, this is the first work to develop this unified view of the classification of selfish behavior.

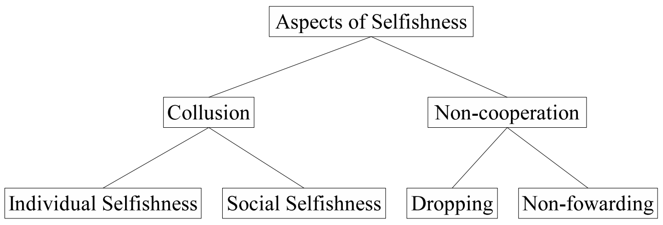


Figure 8.2: Classification of selfish behavior in DTNs

8.3.2 The Methodologies of Investigating the Impact of Selfish Behavior

Since Panagakis et al. [231] first presented their study on the performance degradation caused by selfish behavior in DTNs, researchers have shown significant interest in this field. To evaluate the impact of selfish behavior on the performance of existing routing protocols, some works utilize theoretical analysis models, such as Continuous Time Markov Chains (CTMC), whereas others utilize simulations.

To the best of our knowledge, CTMC is first exploited by Karaliopoulos et al. [165] to demonstrate the impact of selfish behavior in DTNs. Later studies [185, 186, 187, 188] explored CTMC to show the influence of selfish nodes on routing performance in the contexts of social selfishness, constrained energy and multicast routing. CTMC provides a theoretical approach of analyzing selfish behavior in DTNs.

However, the routing process modeled by CTMC is built on the assumption that the inter-contact times between nodes follow exponential distribution, which rarely holds in real-life situations [72, 150]. Moreover, CTMC can only be utilized to model the routing process of simple routing protocols, such as Epidemic [279], Spray and Wait [268]. These routing protocols are generally considered to be inefficient in reality [209]. In addition, the studies based on CTMC do not evaluate the performance of routing protocols in terms of delivery ratio, which is traditionally considered to be the most important performance metric in DTNs. Therefore, authors in [82, 168, 188] utilize simulation methods to investigate the influence of selfish behavior on the routing performance.

8.3.3 The Impact of Selfish Behavior

Since Panagakis et al. [231] first presented their study on the performance degradation caused by selfish behavior in DTNs, researchers have shown significant interest in this field. The existing research works [82, 168, 187, 188] based on theoretical analysis and experimental simulations reveal the following two characteristics of the impact of selfish behavior on the routing performance. Firstly, the routing performance (i.e., delivery ratio, delivery cost and delivery latency) is seriously degraded, if a major portion of the nodes in the network is selfish. For instance, the delivery ratio in the presence of selfish nodes can be as low as 20% compared to what can be achieved under full cooperation [263]. Secondly, the impact on the routing performance is related to the non-cooperative action of selfish behavior (i.e., non-forwarding of messages and dropping of messages). Specifically, the behavior of non-forwarding of messages reduces the delivery cost, while the behavior of dropping of messages increases the delivery cost. However, both of them decrease the delivery ratio, and prolong the delivery latency, even if messages are eventually delivered.

8.4 Strategies for Preventing Selfish Behavior

In order to reduce the impact of selfish behavior on routing performance, a number of studies focus on stimulating selfish nodes to be cooperative. The existing incentive strategies are traditionally classified into three categories [68, 75]: barter-based [67, 68, 288], credit-based [75, 196, 227, 298, 299] and reputation-based [52, 100, 183, 196, 282, 285]. In the following subsections, we will introduce the representative strategies in each category and summarize their common problems.

8.4.1 Barter-based Strategies

The simplest strategies are barter-based or pair-wise Tit-For-Tat (TFT) strategies [67, 68, 288]. The mechanism is that two encountering nodes exchange the same amount of messages. In [67, 68], the authors divide the messages into two categories: primary messages and secondary messages. For a given node, the messages in which it is interested (e.g., the messages destined for it) are primary messages. Other messages are secondary messages. When two nodes encounter each other, they first exchange the description about the messages stored in their buffers. Based on the analysis of the description, each node determines an initial list of the desired messages from the other node, and sorts the messages in order of preference (i.e, the priority of primary messages is higher than that of secondary messages). For the sake of simplicity, let us assume that the size of messages is the same. Finally, each node refines the list by keeping the top K messages in its initial list, where K is the minimum size of two initial lists.

From the above depiction of message selection under this strategy, we can see that it is entirely up to the nodes to determine the desired messages. Thus, a node may adopt selfish behavior towards the secondary messages, in order to conserve its limited resources. However, exchanging the secondary messages is also beneficial, since they can be used to exchange the primary messages in the future. In other words, each message has a potential value, which is employed to prevent selfish behavior. Moreover, the authors in [68, 288] consider the message selection process as a two-person game, and utilize the Nash Equilibrium [223] to increase the message delivery ratio.

After the message selection process, two encountered nodes exchange the messages in the lists one by one (i.e., if a node has sent a message to the other node, it would not send another message, until it receives a message from the latter). In such a manner, even if the connection is disrupted during the exchange process, the maximum difference of the number of the exchanged messages between two nodes is one. Consequently, the fairness of message exchange can be ensured by exchanging approximately the same amount of messages between two encountering nodes.

However, the requirement of exchanging the same amount of messages is a two-edged sword. It degrades routing performance dramatically in the case that one of the two encountering nodes has fewer messages. For instance, let's consider that there are two encountering nodes, called node A and B. Node A contains a message whose destination is node B. However, there is no message in the buffer of node B at the moment. In such a case, the message cannot be delivered to node A. Furthermore, if node A is the source of the message, the performance in terms of delivery ratio is even worse than that achieved by utilizing Direct Delivery [270] which is generally considered to achieve the lower bound for delivery ratio in DTNs.

8.4.2 Credit-based Strategies

Credit-based strategies are proposed to avoid the disadvantages of barter-based strategies. This kind of strategy stimulates nodes to be cooperative by utilizing the concept of virtual credit, which is utilized to pay for message forwarding. The mechanism is that if a node cooperates to forward a message for others, it receives a certain amount of credit as a reward that it can later utilize for its own benefit.

Based on which node is charged with the message forwarding, the credit-based strategies can be further sub-divided into two models [69]: 1) *Message Purse Model* and 2) *Message Trade Model*. In message purse model [75, 196, 299], the source node of a message pays credits to the intermediate nodes which participate in delivering the message to the destination. In the message trade model [227], messages are considered as valuable goods. The receiver pays credits to the sender of a message in each hop-by-hop transmission until the message reaches the destination, which finally pays for the message forwarding. Since the source nodes do not pay for the message forwarding, the message trade model is inherently vulnerable to the source nodes flooding the network. For this reason, most of the credit-based works utilize the message purse model.

In the strategies that belong to the message purse model, the common assumption is the existence of a Virtual Bank (VB), or Credit Clearance Service (CCS). The VB covers the space that the mobile nodes can reach, and can be connected by any nodes in the network. The responsibility of VB is to charge the source node of a message and reward the intermediate nodes which participate in delivering the message to the destination.

The strategies [75, 196, 299] belonging to the message purse model are suitable for different routing protocols. In [196], the proposed strategy is designed for the single-copy routing protocols (e.g., Direct Delivery and First Contact [156]) under which only one message copy exists in the routing process. Although single-copy routing protocols consume the least resources, the routing performances in terms of delivery ratio and delivery latency are generally too low to be applicable in reality [269]. Therefore, more routing protocols (e.g., Epidemic and Spray and Wait) are multi-copy based. In [75, 299], the proposed strategies are targeted to multi-copy based routing protocols in DTNs. In [299], Zhu et al. include the solution of cheating actions (i.e., credit forgery attack, nodular tontine attack and submission refusal attack) which are adopted by the selfish nodes to maximize their benefits. Detailed information about these cheating actions is given in [299].

From the above discussion, we can see that the process of charging and rewarding is invoked at the side of the VB, when (1) a message is successfully delivered to the destination and (2) there are intermediary nodes participating in the routing process. Let's consider a scenario where a major portion of the nodes is selfish and each node has enough credits to request the message forwarding service from an encountering node in a contact. In such a case, a message can only be delivered when the source node directly encounter the destination node. In addition, before the message reaches the destination node, the credits of the source node are reusable to request the message forwarding service. Therefore, a selfish node cannot be aware of the necessity of cooperation with other nodes. Due to the above two reasons, the credit-based strategies cannot efficiently stimulate the selfish nodes to be cooperative, when a major portion of the nodes is selfish.

8.4.3 Reputation-based Strategies

We first explain the concept of reputation before discussing the reputation-based strategies: "Reputation of an agent is a perception regarding its behavior norms, which is held by other agents, based on experiences and observation of its past actions" [193]. In the scope of investigating selfish behavior, the reputation value of a node indicates other nodes' perception about the cooperation of the node. For instance, if the reputation value of a node is low, it means that the node is considered to be selfish by other nodes. If the reputation value of a node is high, it means that the node is considered to be cooperative by other nodes.

The mechanism of this kind of strategy is that a message generated by a given node is forwarded only if the node has forwarded messages originating from others. Therefore, the observation about the behavior of other nodes plays a significant role in this kind of strategy. Based on the feasibility of observation by other nodes, we further divide the existing strategies into two models: 1) *detection-based model* and 2) *non-detection model*.

In the detection-based model, each node monitors the behavior of the intermediary nodes. In [52, 183, 285], the authors utilize different methods to detect selfish behavior in DTNs. In [285], each intermediate node receives a receipt after forwarding a message to another node. The receipt is a proof about the cooperation of the intermediate node. The cooperation of an encountering node is assessed by Beta distribution, which is parameterized by the number of cooperative and selfish actions taken by the node. However, the strategy cannot prevent collusion cheating, which means that some nodes together cheat other nodes in order to increase their reputation. Detailed information about this cheating action is given in [299]. Similar to [285], the behavior of intermediary nodes is proved by the return of a receipt. The difference is that a receiver floods the receipt instead of sending the receipt to the sender. In [52], selfish behavior is detected in a different way. In [52], the sender of a message (including the source and intermediate nodes) keeps the records of the encountered nodes and the forwarding records which contain the identifier of the message, the destination of the message and the forwarding time. When two nodes encounter each other, they check the forwarding records and received messages since last encountered time, in order to detect the cooperative nodes and selfish nodes.

However, due to the unique features of DTNs (e.g., the lack of an end-to-end continuous path and high variation in network conditions), the detection of selfish behavior is considered to be difficult by some authors. The alternatives belonging to reputation-based strategies are not based on the detection of selfish nodes [100, 196]. In [100], Dini et al. decrease the reputation of all nodes periodically, and only increase the reputation of the intermediate nodes who participate in the successful message delivery. Similar to [100], the proposed strategy in [196] decreases the reputation of all nodes periodically. The differences between them are twofold. First, it involves the credit-based incentive strategy to reward the intermediate nodes which participate in the successful message delivery. Second, no matter whether the message delivery succeeds or not, all cooperative nodes can get good reputation values by sending the proofs of collaboration to a Trusted Authority (TA), which is responsible for credit and reputation clearance.

From the above description, we can see that the reputation-based strategies can work well even if a major portion of the nodes takes the selfish behavior of dropping messages. However, this kind of strategy mistakenly considers the collaboration of intermediate nodes as selfish behavior, if the reason causing the failure of message delivery is the message expiration other than selfish behavior of intermediate nodes. It is unfair to the cooperative nodes. Furthermore, it results in the decrement of delivery probability of the message generated by this kind of cooperative node, since they are mistakenly considered as selfish nodes by other nodes. Moreover, since the reputation-based strategies only check whether an intermediate node forwards the message to other nodes or not, it cannot tackle the selfish behavior of non-forwarding messages.

8.5 Analysis of Strategies for Preventing Selfish Behavior

In this section, we first introduce representative strategies in the categories discussed above. We then present the experiment settings. The routing algorithm and performance metrics are subsequently depicted. Finally, we compare the performance of the different strategies for preventing selfish behavior.

8.5.1 Compared Strategies for Preventing Selfish Behavior

In the experiment, we compare the performance of preventing selfish behavior of the following strategies against a basic routing protocol (i.e., Binary Spray and Wait), called *Non-strategy*, which does not cope with the selfish behavior of nodes. The detailed settings of the selected strategies are depicted in Table 8.1.

Barter: In [67], when two nodes encounter each other, they exchange the same amount of messages.

MobiCent: Due to the selected routing algorithm, which will be presented later, is multi-copy based, we choose the MobiCent as the representative strategy in the category of credit-based. In [75], the charging and rewarding processes are performed at the side of Virtual Bank (VB), when a message is firstly delivered to the destination. A constant credit is charged from the account of the source node in VB. The charged credit is equally divided, and distributed to the intermediate nodes in the message delivery path as a reward.

IRONMAN: Compared to barter-based and credit-based strategies, IRONMAN [52] includes the detection of selfish behavior. Therefore, it is selected as the representative strategy in reputation-based strategy. As depicted in Section 8.4 Part C, when two nodes encounter each other, they firstly check the forwarding records and the received messages, in order to detect the selfish nodes. The two encountering nodes then update the opinion about others' behavior with each other.

8.5.2 Simulation Setup

In this experiment, there are 50 nodes in each simulation. To simulate the social relationships, we equally divide the nodes into two groups. Two nodes that belong to the same group are considered to have a social relationship; otherwise, the nodes are considered to not have a social relationship. During the simulation, a message with a random source and destination is generated at every 5 seconds. Since the message generation process lasts for 12 hours, there are 8640 messages generated in each simulation. The detailed settings of the simulation are listed in Table 8.2

Table 8.1: Simulation Parameters for Strategies

Strategy Name	**Parameter Name**	**Value**
MobiCent	Initial Credit for Each Node	1
	Payment for Each Message	1
IRONMAN	Initial Trust for Each Node	0.5
	Trust Increment	0.5
	Trust Decrement	0.5
	Threshold	0.49

Table 8.2: Simulation Parameters

Parameter Name	**Value**
Simulation Area	500 m x 500 m
Simulation Length	13.5 hours
Mobility Model	Random WayPoint (RWP)
Number of Mobile Nodes	50
Number of Groups	2
Number of Nodes in Each Group	25
Transmission Range	10 m
Node Speed	1 m/s
Warm-up Period	0.5 hour
Duration of Message Generation	12 hours
Message Generation Rate	1 message per 5 seconds
Time-To-Live (TTL)	1 hour

8.5.3 Routing Algorithm

Based on the above settings, we conducted our experiment with an efficient multi-copy routing algorithm in MDTNs, called Binary Spray and Wait [268]. The Binary Spray and Wait routing algorithm provides a platform for the selected strategies. The routing process is elaborated below.

Binary Spray and Wait: In [268], each message is associated with an attribute L, which indicates the maximum copies of the message that a message carrier can make. For each message, there are two phases: *spray* phase and *wait* phase. In the spray phase (i.e., $L > 1$), a message carrier hands over half of its message copies to an encountering node without the message. In the wait phase, the message can only be forwarded to the destination node. In the experiment, L is set to 5.

8.5.4 Performance Metrics

We observe the following metrics to assess the impact of selfish behavior in DTNs:

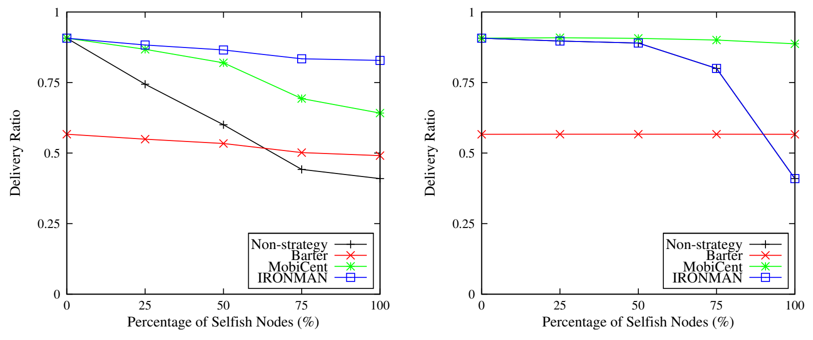
Delivery Ratio: The proportion of messages that have been delivered out of the total unique messages created.

Delivery Cost: The total number of messages (including duplicates) transmitted in the simulation. To normalize this, we divide it by the total number of unique messages created.

8.5.5 Simulation Results

In Figure 8.3(a), the performance of the strategies for preventing dropping messages is shown. When there is no selfish node, the performance of Non-strategy, MobiCent and IRONMAN is the same, since the cooperative nodes always cooperate with other nodes. However, the performance of barter is lower than those of other strategies, due to the requirement of exchanging the same amount of messages. As the percentage of selfish nodes increases, the performance of all strategies is degraded. The performance of IRONMAN and MobiCent is always better than that of Non-strategy. The performance of barter exceeds that of Non-strategy, when the percentage of selfish nodes is about 60%. The performance of IRONMAN is much better than those of other strategies even if all nodes are selfish, since it can detect the dropping of messages of a selfish node.

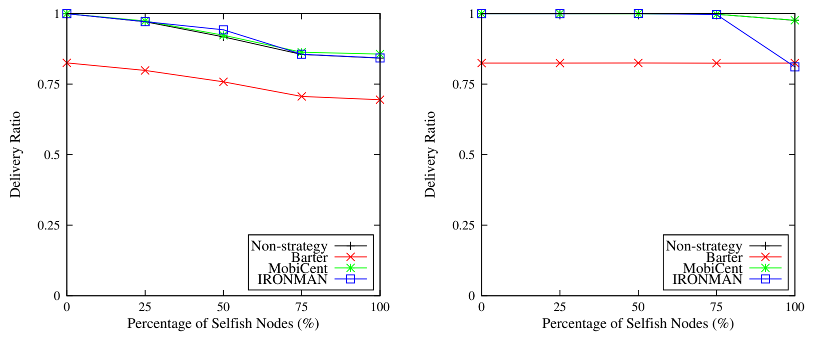
Figure 8.3(b) illustrates the performance of the strategies for preventing non-forwarding messages. The performance of all strategies decreases, as the percentage of selfish nodes increases. The performance of MobiCent is always better than other strategies. MobiCent can stimulate selfish nodes to be cooperative, as the number of the messages generated by selfish nodes increases. IRONMAN always achieves the same performance as Non-strategy, since it cannot detect the selfish behavior of non-forwarding of messages. The performance of Barter only exceeds than that of Non-strategy, when the percentage of selfish nodes is about 82%.



(a) Dropping (b) Non-forwarding

Figure 8.3: The routing performance in terms of delivery ratio under individual selfishness. The selfish actions of dropping and non-forwarding messages are illustrated in (a) and (b) respectively.

In Figure 8.4, the performance of delivery ratio of the four strategies under the social selfishness is investigated. From the figures, we can see that all the strategies cannot work well under social selfishness. Specially, the performance of barter is even worse than that of Non-strategy, due to the requirement of exchanging the same amount of messages. For the selfish behavior of non-forwarding of messages, the performance of MobiCent is much better than those of other strategies, when the selfish nodes are 75% percentage.

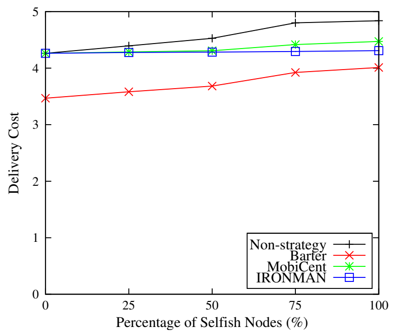


(a) Dropping (b) Non-forwarding

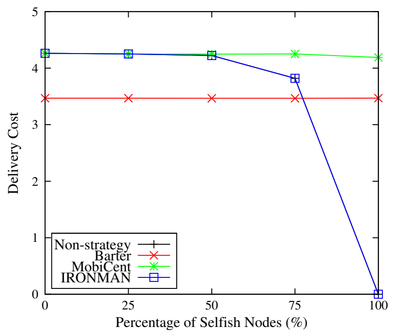
Figure 8.4: The routing performance in terms of delivery ratio under social selfishness. The selfish actions of dropping and non-forwarding messages are illustrated in (a) and (b) respectively.

The performance of delivery cost is demonstrated in Figure 8.5 and 8.6. In Figure 8.5(a), due to the characteristics of dropping of messages, the delivery cost of all strategies increases, as the percentage of selfish nodes increases. Meanwhile, the delivery cost of all incentive strategies is lower than that of Non-strategy, since they stimulate selfish nodes to be cooperative. However, in Figure 8.6(b), the delivery cost of Non-strategy, IRONMAN, and MobiCent decreases, as the percentage of selfish nodes increases, since they cannot deal with social selfishness. In Figure 8.5(b) and Figure 8.6(b), due to the characteristics of non-forwarding of messages, the delivery cost of all strategies decreases, as the percentage of selfish nodes increases. The delivery cost of MobiCent is higher than that of Non-strategy when the percentage of selfish nodes is high, due to the stimulation of selfish nodes to be cooperative.

From the above analysis of the simulation results, we can see that, for individual selfishness, the reputation-based strategies cannot prevent the selfish behavior of non-forwarding of messages. For

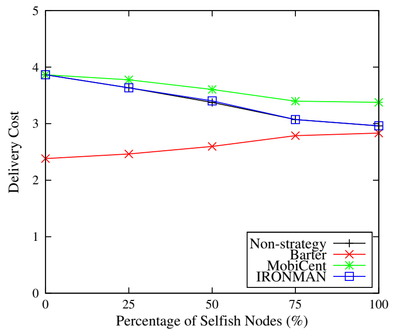


(a) Dropping

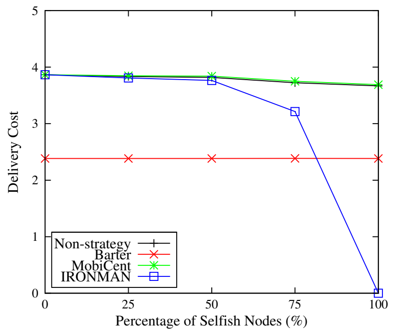


(b) Non-forwarding

Figure 8.5: The routing performance in terms of delivery cost under individual selfishness. The selfish actions of dropping and non-forwarding messages are illustrated in (a) and (b) respectively.



(a) Dropping



(b) Non-forwarding

Figure 8.6: The routing performance in terms of delivery cost under social selfishness. The selfish actions of dropping and non-forwarding messages are illustrated in (a) and (b) respectively.

social selfishness, there is no strategy that can efficiently prevent the selfish behavior of dropping of messages, and credit-based strategies can prevent the selfish behavior of non-forwarding of messages. The performance of barter-based strategies is always worse than that of credit-based and reputation-based strategies.

8.5.6 Comparison of Strategies

According to the above simulation results, we utilize three types of circles to indicate the performance of the selected strategies compared with that of Non-strategy: (1) ● indicates that the performance of a given strategy is always better than that of Non-strategy; (2) ◑ indicates that the performance of a given strategy is better than that of Non-strategy, only when the percentage of selfish nodes is high; and (3) ○ indicates that the performance of a given strategy always cannot exceed that of Non-strategy. The performance of the representative strategy in each category is listed in Table 8.3.

Table 8.3: Performance comparison of the selected strategies

Strategy	**Individual selfishness**		**Social selfishness**	
	Dropping	**Non-forwarding**	**Dropping**	**Non-forwarding**
Barter	◑	◑	○	○
MobiCent	●	●	○	◑
IRONMAN	●	○	○	○

8.6 Privacy

In this section, we will classify privacy preserving protocols for MDTN routing according to their specific privacy objectives.

8.6.1 Classification of Privacy Objectives

As mentioned in the introduction, messages in MDTNs are relayed by intermediary nodes. Apart from selfishness, mobile device carriers can be unwilling to participate in the routing process due to the concern of privacy. Recent years have seen considerable research works addressing the issues of privacy in MDTNs. The protocols in the literature are mainly concerned with preserving the privacy of one or more of the following sensitive user aspects: (1) identity, (2) location, (3) message content, and (4) relationships. We can thus classify the existing privacy preserving protocols according to their privacy objectives. Please refer to Figure 8.7 for an illustration of this classification. We discuss each of these privacy objectives in the following section along with some solutions proposed in the literature for achieving these objectives.

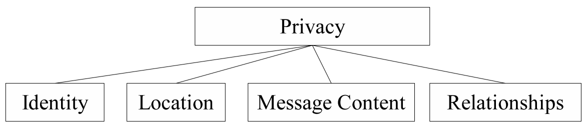


Figure 8.7: Classification of privacy from the aspect of privacy objective

8.7 Strategies for Preserving Privacy

8.7.1 Identity Privacy

In the category of identity privacy, the identity of nodes participating in message delivery is considered as private information.

Kate et al. [166] presented an anonymous communication architecture for DTNs using Identity-Based Cryptography (IBC) [261]. This is one of the first anonymous communication solutions specifically for DTNs. Kate et al. use a construct called DTN gateways, which are entities assumed to be trusted and to be aware of user identities. In the routing process, a DTN gateway replaces the identity of a source node with a pseudonym unlinkable to the identity. The advantage of the protocol is that there is not much overhead for routing. However, the protocol relies on the assumption that trusted DTN gateways are present, which is a strong assumption for MDTNs.

Le et al. [181] proposed a privacy preserving infrastructure called Privacy-Enhanced Opportunistic Networks (PEON) based on onion routing [246]. In PEON, nodes are clustered into groups. Nodes in the same group share public keys. Before sending a message, a source node determines the routing path, which contains a certain number of node groups. The message is then encrypted by the public keys of the destination node and the determined groups in an inverse order. Thus, each relay node can only be aware of the next hop (i.e., a node group) in the routing path and remains unaware of the identity of the source node. Compared to classic onion routing, the routing performance of PEON in terms of delivery ratio and delivery latency is enhanced due to the utilization of multicasting inside a group. However, node groups are randomly clustered, which may result in the inefficient dissemination of messages inside a group. In addition, the assumption of a Public Key Infrastructure (PKI) rarely holds in MDTNs [166].

Lu et al. [195] presented a social-based privacy-preserving packet forwarding protocol (named SPRING) for Vehicular DTNs. In SPRING, Road Side Units (RSUs) are assumed to be trusted and uncompromisable. Similar to [297], RSUs are strategically deployed at some highly-social intersections to temporarily buffer the messages as relays. Due to the utilization of RSUs, an adversary cannot find out the identity of the source and the destination nodes. However, the private information of nodes is disclosed, if any RSU in the network is compromised. Additionally, all RSUs in SPRING are managed by a single management authority, which results in inflexibility.

8.7.2 Location Privacy

In the category of location privacy in MDTNs, the discovery of the user location by the adversary is considered as the main privacy threat. In an untrusted network, the mobile device owners do not want others to know their positions for personal security reasons [197].

In [197], Lu et al. proposed the Anti-Localization Anonymous Routing (ALAR) protocol for MDTNs. In ALAR, each message is divided into k segments and each segment is then encrypted and sent to n different neighbors. Therefore, an adversary may receive several copies of a segment at different times from different relay nodes. Even if the adversary collects these segments, they cannot localize the source node with high probability. The disadvantage is that the routing performance is influenced by the setting of the parameters k and n. Specifically, the routing performance in terms of delivery ratio and delivery latency is degraded as the two parameters increase.

Zakhary and Radenkovic [291] presented a location privacy protocol that is based on the utilization of social information of nodes. In this protocol, each node maintains a social profile, which includes n profile attributes. The social relationship between nodes are inferred by the matching of profile attributes. For each message, the forwarding is guided by the obfuscated attributes in the first k hops. After that, the message can be routed by any routing protocols. Therefore, an adversary cannot distinguish the location of the source node from the other k relay nodes. However, nodes that have strong social relationships are generally considered to be frequently co-located. Thus, the adversary can still detect the approximate location of the source node. Moreover, the routing performance is degraded, due to the extra k forwarding hops.

8.7.3 Message Content Privacy

Since messages are relayed by intermediary nodes in MDTNs, the content of messages can be unintentionally disclosed to these nodes in the routing process. Thus, in the category of message content privacy, the content of messages is considered as private information.

Jansen and Beverly [157] proposed a Threshold Pivot Scheme (TPS) based on the technique of secret sharing [226]. In TPS, a message, considered as the secret, is divided into multiple shares by the technique of secret sharing. The shares are delivered to the destination node via multiple independent paths. The content of a message is thus protected from individual intermediary nodes. At the destination node, the message can be reconstructed by the knowledge of any τ shares. The disadvantage of this protocol is that if an adversary successes n monitoring a sybil attack, it can create multiple pseudonymous nodes and then intercept sufficient number of shares.

Shi and Luo [264] proposed an anonymous communication mechanism called ARDEN based on onion routing [246], multicast dissemination and Attribute-Based Encryption (ABE) [124]. In ARDEN, before sending a message, the source node determines a path of disjoint groups, one of which includes the destination node. The message is then encrypted by the keys of the destination node and the grouping keys. Compared with the traditional onion routing, the advantage of ARDEN is that it encrypts messages with the keys of groups rather than the keys of individual intermediate nodes. The performance in terms of delivery ratio and delivery latency can be improved, since all nodes in the same group can participate in message forwarding. On the other hand, the arbitrary group partitioning manner may result in performance degradation in terms of delivery ratio and delivery latency.

8.7.4 Relationships Privacy

As mentioned in the introduction, the mobility pattern of nodes plays an important role in the routing process. A number of proposed routing protocols exploit the encounter probability [89, 191] and social relationship of nodes [89, 151] to guide the message forwarding decision. However, such information is considered as personal and private [234] thus users may hesitate in participating in such protocols.

Hasan et al. [140] proposed a Privacy Preserving Prediction-based Routing (3PR) protocol for MDTNs. A prediction-based routing protocol for MDTNs works by forwarding a message from one intermediate node to another if the latter has higher probability of encountering the destination node. However, this process compromises the privacy of the nodes by revealing their mobility patterns. 3PR forwards messages by comparing information about communities of nodes instead of individual nodes. Specifically, it compares the maximum probability that a node in the community of a potential intermediate node will encounter the destination node. Simulations on a community-based mobility model demonstrate that the protocol has comparable performance to existing prediction-based protocols.

Parris and Henderson [234] presented the Privacy-enhanced Social-network Routing protocol. This protocol takes advantage of obfuscated social information rather than accurate social information to guide the message forwarding. The original social information of a node is obfuscated by the following two approaches: (1) modifying the friend list, i.e., adding or removing some items into or from the friend list, or (2) using a Bloom filter [55] to hash the friend list. The advantage of the protocol is that the presence of a public key infrastructure is not necessary. However, message routing may be guided erroneously due to the utilization of obfuscated social information. Moreover, in the case of modifying the friend list of a source node, an adversary can approximately determine the source node's friends by collecting the messages from the source node. In the second approach, the probability of false positives increases as the Bloom filter becomes more full, due to the characteristics of Bloom filter.

8.8 Analysis of Strategies for Preserving Privacy

8.8.1 Criteria for Comparison

The criteria for comparison of the above privacy preserving protocols are described in the following sections.

Adversarial models

We identify two adversarial models, which characterize the behavior of dishonest users. The models are: Semi-Honest, and Malicious. A privacy preserving protocol is considered secure under one of these models if it can show correctness and meet its privacy requirements under the given model.

Collusion

A dishonest user may act alone or multiple dishonest users may act in agreement to achieve their ulterior motives. When multiple dishonest users work together, it is referred to as collusion. Privacy preserving protocols either consider that collusion can take place between users or consider that collusion does not take place.

Security Building Blocks

The privacy preserving protocols for MDTN routing are generally built using security building blocks such as Identity-Based Cryptography (IBC) [261], Public Key Infrastructure (PKI), Onion routing [246], Secret Sharing, Attribute-Based Encryption (ABE) [124], and Bloom filter [55].

8.8.2 Comparison of Strategies for Preserving Privacy

A comparison of the above privacy preserving strategies is given in Table 8.4 according to the established criteria.

Table 8.4: Comparison of Strategies for Preserving Privacy

Protocol	Privacy Objective	Collusion	Attack Model	Building Blocks
Kate et al. [166]	Identity	Group	Semi-Host	IBC
Le et al. [181]	Identity Content	Group	Semi-Host	Onion Routing PKI
Lu et al. [195]	Identity	Group	Semi-Host	
Lu et al. [197]	Location	Individual	Semi-Host	Secret Sharing
Zakhary and Radenkovic [291]	Location	Individual	Semi-Host	
Jansen and Beverly [157]	Content	Individual	Semi-Host	Secret Sharing
Shi and Luo [264]	Identity Content	Group	Semi-Host	Onion Routing ABE
Hasan et al. [140]	Relationships	Group	Semi-Host	Secret Sharing
Parris and Henderson [234]	Relationships	Individual	Semi-Host	Bloom Filter

8.9 Conclusion

In this work, we investigated the existing research works that address the unwillingness of nodes to participate in MDTN routing. We identified the factors of selfishness and privacy as the two primary reasons why nodes are unwilling to participate. For selfishness, we first developed a classification of the aspects of selfish behavior. We then classified the existing strategies for preventing selfish behavior into three categories: barter-based, credit-based and reputation-based. We subsequently analyzed the mechanisms of the proposed strategies and pointed out the problems in each category. We then conducted an experiment to investigate the performance of the representative strategies for preventing different types of selfish behavior. For privacy, we classified the existing privacy preserving protocols for MDTNs according to their specific privacy objectives: identity, location, message content, and relationships. We reviewed the various strategies proposed in the literature for preserving the privacy of nodes under each of these categories. We also presented an analytical comparison of the privacy preserving protocols.

Chapter 9

Privacy-Preserving Routing in Mobile Delay Tolerant Networks

Message routing is one of the major challenges in Mobile Delay Tolerant Networks (MDTNs) due to frequent and long-term network partitions. A number of routing protocols for MDTNs belong to the category of prediction-based routing protocols, which utilize the social encounter probability of nodes to guide message forwarding. However, these prediction-based routing protocols compromise the privacy of the nodes by revealing their mobility patterns. In this work, we propose the Privacy Preserving Probabilistic Prediction-based Routing (4PR) protocol that forwards messages by comparing aggregated information about communities instead of individual nodes. Specifically, it compares the probability that at least one node in a community will encounter the destination node. We present theoretical security analyses as well as practical performance evaluations. Our simulations on a well established community-based mobility model demonstrate that our routing protocol has comparable performance to existing prediction-based protocols. Additionally, the community information is computed efficiently and independently of the routing protocol.

This chapter is an adapted version of the article: “4PR: Privacy Preserving Routing in Mobile Delay Tolerant Networks.” J. Miao, O. Hasan, S. B. Mokhtar, L. Brunie, and A. Hasan. *Computer Networks (Elsevier).* 2016. Vol. 111, pp. 17-28.

9.1 Introduction

Mobile Delay Tolerant Networks (MDTNs) (also referred to as Mobile Opportunistic Networks) are constructed by the intermittent connection of co-located mobile devices. The MDTN architecture caters to the rapidly expanding cyber-physical space where mobile and socially connected human users are coupled with smart portable devices forming mobile network nodes. The short range networking interfaces (e.g., Bluetooth) of these devices enable Mobile Networking in Proximity (MNP), where neighboring devices interact through short-range communications. However, routing messages between two nodes that are not within communication range is a challenge in MDTNs since an end-to-end routing path cannot be guaranteed. The applications developed in these networks are often geo-localized with no critical time constraint, e.g., advertisement dissemination, recommendation of points of interest, and asynchronous communication.

In order to deal with the lack of end-to-end connectivity between nodes, message routing in MDTNs is often performed in a “store-carry-and-forward” manner [109], in which a node may store and carry a message for some time before opportunistically forwarding it to another node [271]. In order to better choose intermediary nodes, a number of routing protocols [289, 294] forward a message from one intermediate node to another if the latter has higher probability of encountering the destination node. Such routing protocols are called prediction-based routing protocols. It has been shown that these protocols perform better than other protocols when nodes exhibit well-known mobility patterns [289, 294]. However, prediction-based routing protocols implicitly assume that nodes accept revealing their mobility patterns to other nodes. In practice, the disclosure of mobility patterns can result in the unwillingness of nodes to participate in MDTNs due to privacy concerns [210].

In this work, we present the Privacy Preserving Probabilistic Prediction-based Routing (4PR) protocol for MDTNs. For routing a message, 4PR distinguishes the routing inside a community from the routing between communities. A community is defined as a set of nodes that frequently encounter each other (see Section 9.3). For disseminating a message inside a community, 4PR relies on the epidemic protocol [279], which by construction preserves the privacy of nodes and is efficient as communities are small. The main challenge addressed by 4PR is thus the routing of a message between communities in a privacy preserving manner. To do so, each node in the network calculates the probability that at least one of the nodes in its community will encounter the destination. When two nodes from different communities encounter, instead of comparing their respective probabilities to encounter the destination node, they compare the aforementioned probabilities to determine the message forwarding decision. The probability that at least one node in a community will encounter a given node in the network is computed in a privacy preserving manner within the community using the MDTN-Private-Probability protocol, presented in Section 9.5.

To the best of our knowledge, only our previous work (the 3PR protocol [141]) has addressed the privacy issue of prediction-based routing protocols. In 3PR, message routing is guided by the maximum probability that nodes in a community will encounter a destination node. In contrast, in the 4PR protocol, message routing is guided by the probability that at least one node in a community will encounter the destination node. This fundamental difference in how messages are forwarded provides 4PR some significant advantages over 3PR. As we discuss in further detail in Section 9.2.1, the advantages of 4PR over 3PR include: 1) better privacy preservation since the true upper bound of encounter probabilities is not revealed; 2) private computation of probability is more efficient than private computation of maximum; 3) the probability that at least one node in a community will encounter the destination node is a more accurate measure for routing path prediction than the maximum probability in the community.

We evaluate 4PR both theoretically by providing security analyses (Sections 9.4 and 9.5) and practically through extensive simulations (Section 9.6). We have conducted our simulations based on a well established community-based mobility model [89, 267]. We compare the performance of 4PR against five state-of-the-art protocols, i.e., epidemic [279], Direct [270], PRoPHET [191], Bubble [151], and the 3PR protocol [141]. Epidemic and Direct are traditionally considered to achieve the upper and lower bounds of routing performance. PRoPHET and Bubble are representatives in prediction-based and community-based routing protocols respectively. Results show that 4PR has comparable performance to existing prediction-based protocols while preserving the privacy of the nodes.

The remainder of this chapter is structured as follows. Section 9.2 discusses related work on privacy preserving protocols in MDTNs. The system model is described in Section 9.3. We describe the 4PR protocol in Section 9.4 followed by the MDTN-Private-Probability protocol presented in Section 9.5. The performance evaluation is subsequently presented in Section 9.6. We conclude in Section 9.7.

9.2 Related Work

Recent years have seen considerable research addressing the issues of privacy in delay tolerant networks. The protocols in the literature are mainly concerned with preserving the privacy of one or more of the following sensitive user aspects [210]: (1) identity, (2) location, (3) message content, and (4) relationships. In contrast, our protocol 4PR is a novel type of protocol, which has the specific goal of hiding the encounter probabilities of nodes. Therefore, 4PR differs fundamentally from other existing privacy preserving routing protocols for MDTNs due to the difference in objectives.

Hasan et al. [141] proposed the Privacy Preserving Prediction-based Routing (3PR) protocol for MDTNs, which is the predecessor of the 4PR protocol presented in this work. This is the only other work that we are aware of that has the same objective as 4PR, i.e., hiding the encounter probabilities of nodes. In 3PR, when two nodes from different communities encounter, they compare the *maximum probability in their community* that a given node will encounter the destination. However, compared with 4PR, the forwarding decision mechanism of 3PR has the following two shortcomings. Firstly, 3PR consumes much more resources in terms of the number of message copies to compute the maximum probability in the community (see Section 9.6.1). Secondly, the maximum probability in the community cannot accurately measure the probability of all nodes in the community delivering the message to the destination node, due to the message being flooded inside the community.

We note some protocols that attempt to preserve privacy in the other aforementioned categories. In the category of identity privacy, the identity of nodes participating in message delivery is considered as private information. Papapetrou et al. [232] propose the SimBet-BF routing protocol for MDTNs. This anonymized routing protocol represents all node identities using Bloom filters. The desired effect is that two nodes can exchange information while maintaining the privacy of their identity and their past encounters. However, the protocol described by Papapetrou et al.

is not a prediction-based protocol. In fact, a direction of future work described by the authors is to use encounter information to enhance the routing protocol.

Kate et al. [166] presented an anonymous communication architecture for MDTNs using Identity-Based Cryptography (IBC). This is one of the first anonymous communication solutions specifically for MDTNs. Kate et al. use a construct called MDTN gateways, which are entities assumed to be trusted and to be aware of user identities. In the routing process, a MDTN gateway replaces the identity of a source node with a pseudonym unlinkable to the identity. The advantage of the protocol is that there is not much overhead for routing. However, the protocol relies on the assumption that trusted MDTN gateways are present, which is a strong assumption for MDTNs.

In the category of location privacy in MDTNs, the discovery of the user location by the adversary is considered as the main privacy threat. Zakhary and Radenkovic [291] presented a location privacy protocol that is based on the utilization of social information of nodes. In this protocol, each node maintains a social profile, which includes n profile attributes. The social relationship between nodes are inferred by the matching of profile attributes. For each message, the forwarding is guided by the obfuscated attributes in the first k hops. After that, the message can be routed by any routing protocols. Therefore, an adversary cannot distinguish the location of the source node from the other k relay nodes. However, nodes that have strong social relationships are generally considered to be frequently co-located. Thus, the adversary can still detect the approximate location of the source node. Moreover, the routing performance is degraded, due to the extra k forwarding hops.

Since messages are relayed by intermediary nodes in MDTNs, the content of messages can be unintentionally disclosed to these nodes in the routing process. Thus, in the category of message content privacy, the content of messages is considered as private information. Shi and Luo [264] proposed an anonymous communication mechanism called ARDEN based on onion routing [246], multicast dissemination and Attribute-Based Encryption (ABE) [124]. In ARDEN, before sending a message, the source node determines a path of disjoint groups, one of which includes the destination node. The message is then encrypted by the keys of the destination node and the grouping keys. Compared with the traditional onion routing, the advantage of ARDEN is that it encrypts messages with the keys of groups rather than the keys of individual intermediate nodes. The performance in terms of delivery ratio and delivery latency can be improved, since all nodes in the same group can participate in message forwarding. On the other hand, the arbitrary group partitioning manner may result in performance degradation in terms of delivery ratio and delivery latency.

In the category of relationships privacy in MDTNs, the social relationships of nodes is considered as personal and private thus users may hesitate in participating in such protocols. Parris and Henderson [234] presented the Privacy-enhanced Social-network Routing protocol. This protocol takes advantage of obfuscated social information rather than accurate social information to guide the message forwarding. The original social information of a node is obfuscated by modifying the friend list, i.e., adding or removing some items into or from the friend list. The advantage of the protocol is that the presence of a public key infrastructure is not necessary. However, message routing may be guided less accurately due to the utilization of obfuscated social information.

9.2.1 4PR vs. 3PR

In [141], we presented our Privacy Preserving Prediction based Routing protocol, abbreviated as the 3PR protocol. In this work, we propose the Privacy Preserving Probabilistic Prediction based Routing protocol, which we abbreviate as the 4PR protocol. The original 3PR protocol provided significant advantages over state of the art routing protocols, notably preservation of user privacy while maintaining comparable routing performance. Our newer 4PR protocol proposes further improvements to privacy preserving prediction based routing. In this section we will present an architectural comparison between 4PR and 3PR, describe the shortcomings of the 3PR protocol, and an overview of how the 4PR protocol overcomes those shortcomings and implements an even stronger privacy preserving routing protocol.

4PR and 3PR share some commonalities, which include routing a message inside a community using the epidemic protocol [279], which by construction preserves the privacy of nodes and is efficient as communities are assumed to be small. The main difference between 4PR and 3PR is how the routing of a message between communities in a privacy preserving manner is handled.

In 3PR, when two nodes from different communities encounter, they compare the *maximum probability in their community* that a given node will encounter the destination. However, in 4PR, when two nodes from different communities encounter, instead they compare the *probability of at least one node in their community* encountering the destination node.

There are a number of advantages to comparing the *probability of at least one node in the community* as in 4PR over comparing the *maximum probability in the community* as in 3PR.

Firstly, in 3PR, although the maximum probability in the community is an aggregate value and does not reveal the precise private probability $P_{a_i,d}$ of an individual node a_i encountering the destination node d, it still divulges some undesirable information about the private probability. Specifically, the maximum value reveals the upper bound on the private value. For example, if the maximum is given as 0.4, then the adversary learns that the private value $P_{a_i,d}$ is no higher than 0.4. On the other hand, the 4PR protocol demonstrates the probability of at least one node in the community encountering the destination. This aggregate value does not reveal the true upper bound or any lower bound on the private value of an individual node.

Secondly, the protocol for computing the probability of at least one node in the community encountering the destination is much more efficient than the protocol for computing maximum probability in the community. As described in Section 9.5, the protocol for the former requires one round of multiplication, whereas as described in [141], the protocol for the latter requires several rounds of summation depending on the number of bits that represent the private number. The network resources required for multiplication and summation required in the two protocols being equal, the protocol for 4PR is much more resource efficient.

Thirdly, the probability of at least one node in the community encountering the destination (as in 4PR) is a more accurate measure for the likelihood of some node in the community encountering the destination than the maximum probability in the community (as in 3PR). Let's take an example to demonstrate this difference. Let's say that there are two communities C_1 and C_2. Community C_1 has three nodes each of which has a probability of 0.8 of encountering the destination node, whereas community C_2 has three nodes, one with probability of 0.8, and the remaining two with probability 0 of encountering the destination node. The maximum probability in both communities is 0.8 (as in 3PR), whereas the probability of at least one node in the community encountering the destination is 0.99 and 0.8 in C_1 and C_2 respectively, according to Equation (9.3) (as in 4PR). Clearly, 4PR provides a more accurate measure of the likelihood.

The above stated advantages offered by 4PR over 3PR make 4PR a major improvement over 3PR, which was to the best of our knowledge, the first privacy preserving prediction based routing protocol for mobile delay tolerant networks in the literature.

9.3 System Model

9.3.1 A Mobile Delay Tolerant Network Model

We consider a set $\mathbb{A}$ of N nodes with communication facilities that can freely roam in a physical environment. The communication facilities consist of a short range wireless connection. Two nodes can communicate only if they are adjacent to each other, i.e, if they are physically within each other's transmission range. We assume that the communication is unreliable, i.e., a message sent from a node to an adjacent node may not arrive. However, we assume that a node knows whether the transmission of a message has been interrupted by a network failure or whether the message correctly reached the intended recipient.

To send a message to a destination node that is not within the transmission range of the source node, the latter uses a routing protocol. The routing strategy that we consider in this work is prediction-based routing [194]. We generalize prediction-based routing protocols as follows: Consider a node a that has a message for a destination node d. When the node a encounters another node b, it forwards a copy of the message to the node b if the probability of b encountering d (given as $P_{b,d}$) is higher than the probability of a encountering d (given as $P_{a,d}$). Thus the probability that a node with a copy of the message will encounter the destination node continues to rise until the message is delivered or the Time To Live (TTL) of the message expires.

As demonstrated in many studies of real human mobility traces, we assume that nodes belong to communities [151]. We define a community C as a set of nodes such that $C \subset \mathbb{A}$. We assume that the nodes in a community are frequently physically collocated and thus a high probability exists of successful message delivery from any source node in a community to any destination node in the community. A node $l \in C$ is designated as the leader of the community. A consensus protocol may be used for the election of the leader node within a community. The leader node maintains the list of the nodes in the community. Let the set of nodes in a community $C = \{a_1, a_2, \ldots, a_n\}$, where $n = |C|$. We consider a community to comprise of at least three nodes, that is, $n \geq 3$. The topic of community management has been discussed in detail in the literature by several authors including Hui et al. [152], Dang and Wu [89], and Miao et al. [211].

We consider the probability that a node a will encounter a node d as private information. Nodes are willing to let this private information be used for routing of messages. However, nodes require that their private information is not revealed to any other node in the network, which includes fellow nodes in a community.

In this work, we consider the semi-honest adversarial model [121]. The nodes in this model always execute the protocol according to the specification. However, the adversary passively attempts to learn the private information of nodes by using intermediate information gleaned during the execution of the protocols.

9.3.2 Computation of Encounter Probabilities of Nodes

In this work, the encounter probabilities of nodes are computed according to the method proposed by Lindgren et al. [191]. The computation of the encounter probabilities of nodes is driven by events. There are two kinds of events: (1) *Connect Event*, and (2) *Update Event*.

(1) *Connect Event*. It happens at the moment when two nodes, e.g., nodes a and b, encounter each other. When a connect event takes place, two encountering nodes compute their encounter probabilities for a given time window according to Equation (9.1), where $P_{init} \in [0, 1]$ is an initialization constant, and $P^{'}_{a,b}$ is the previous probability that node a may encounter node b.

$$P_{a,b} = P^{'}_{a,b} + (1 - P^{'}_{a,b}) \times P_{init} \tag{9.1}$$

(2) *Update Event*. The update event is periodically invoked by all nodes every δ time units. When an update event happens, each node in the network utilizes an aging equation to reduce the probabilities of encountering the other nodes. The intuition behind such a strategy is that a pair of nodes are less likely to encounter each other in the future if they have not encountered in a while. The aging equation is expressed in Equation (9.2), where $\alpha \in [0, 1)$ is an aging constant.

$$P_{a,b} = P^{'}_{a,b} \times \alpha \tag{9.2}$$

It is worth pointing out that the computation of the encounter probabilities of nodes are based on their own histories. This implies that nodes compute the encounter probabilities locally. Therefore, the computation of these probabilities is carried out in a privacy preserving manner.

9.4 4PR: Privacy Preserving Probabilistic Prediction-based Routing

9.4.1 Protocol Description

As stated in Section 9.3, C is a community, such that $C = \{a_1, a_2, \ldots, a_n\}$, and $n = |C|$. Let $P_{C,d} = prob(C, d)$ be the probability that at least one node a_i in community C will encounter the destination node d, given as Equation (9.3).

$$P_{C,d} = 1 - \prod_{i=1}^{n}(1 - P_{a_i,d}) \tag{9.3}$$

We now present an overview of 4PR, our Privacy Preserving Probabilistic Prediction-based Routing protocol. A routing example is depicted in Figure 9.1. This figure shows a number of nodes belonging to three communities C_1, C_2 and C_x. A source node s that belongs to the community C_1 wants to send a message to a node d that belongs to the community C_x.

In 4PR, we distinguish the routing inside a community from the routing between communities. Specifically, when two nodes that belong to the same community encounter each other, they exchange all the messages that they each have. On the other hand, if two nodes a_{11} and a_{21} that belong to different communities C_1 and C_2 respectively encounter each other, node a_{11} forwards a message intended for a destination node d to node a_{21}, only if the probability of at least one node in community C_2 encountering d (given as $P_{C_2,d}$) is higher than that in community (given as $P_{C_1,d}$). In Figure 9.1, when node a_{11} encounters node a_{21}, node a_{11} forwards the message intended for d to node a_{21} because $prob(C_2, d) > prob(C_1, d)$.

In other words, to route a message m from s to d, m is first disseminated in an epidemic manner inside the community C_1. Message m then moves from a community to another such that: (1) at each forwarding step, the probability of at least one node in the next community to reach the

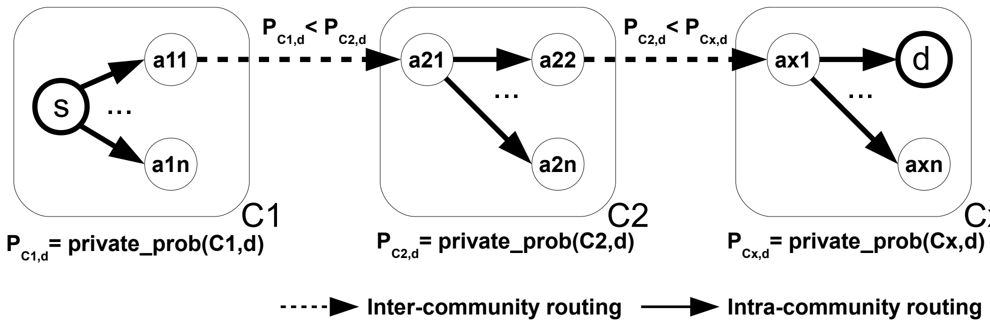


Figure 9.1: 4PR Protocol Overview

Protocol: MDTN-4PR
Participants: Node a and node b, where $a, b \in \mathbb{A}$.
Input: (1) m, a message. (2) d, the destination node of message m. (3) C_a, the set which denotes the community of node a. (4) C_b. (5) $P_{C_a,d} = prob(C_a, d)$, that is the probability that at least one node in community C_a will encounter the destination node d. (6) $P_{C_b,d}$.
Output: Message m is delivered to the node b if $b = d$, or $b \in C_a$, or $P_{C_b,d} > P_{C_a,d}$.
Setup: Node a has a message m whose destination is node d.
Events and Associated Actions:

node a encounters a node b

```
1  if $b = d$
2      then node $a$ sends message $m$ to node $b$
3  elseif $b \in C_a$
4      then node $a$ sends a copy of the message $m$ to node $b$
5  elseif $P_{C_b,d} > P_{C_a,d}$
6      then node $a$ sends a copy of the message $m$ to node $b$
```

Figure 9.2: Protocol: MDTN-4PR

destination is higher than that in the previous community, (2) as soon as it reaches a community, m is disseminated in an epidemic manner within the community.

A key characteristic of 4PR is that $P_{C_a,d} = prob(C_a, d)$, the probability that at least one node in community C_a will encounter the destination node d, is computed in a privacy preserving manner, that is without revealing the individual probabilities of the nodes in the community. $prob(C_a, d)$ is therefore denoted as $private\_prob(C_a, d)$ in Figure 9.1.

Our protocol 4PR for Privacy Preserving Probabilistic Prediction-based Routing in Mobile DTNs is specified in Figure 9.2. The computation of $private\_prob(C_a, d)$ is performed using a decentralized protocol for privately computing the function over a set of values in a delay tolerant manner without revealing the individual values, i.e., MDTN-Private-Probability, further described in Section 9.5.

The probability is computed periodically in the community independently from the routing protocol. Therefore, the complexity of the MDTN-Private-Probability protocol has no direct impact on the performance of the routing protocol.

9.4.2 Security Analysis: Correctness

With each forwarding of the message, the conventional prediction-based routing strategy delivers a copy of the message to a node that has a higher probability of encountering the destination node. We consider our protocol 4PR to be correct if it achieves the same effect as the conventional prediction-based routing strategy.

In 4PR, a node a in community C_a sends message m to node b in a community C_b if a and b encounter and $P_{C_b,d} > P_{C_a,d}$, i.e., if the probability of at least one node in C_b encountering the destination node d is higher than that in C_a (lines 7 and 8). Upon receiving the message m, node b floods the message to all nodes in C_b (lines 4 and 5). In Section 9.3, we stated the assumption that a high probability exists of successful message delivery from any source node in a community to any destination node in the community. Given this assumption, the message m reaches all nodes in C_b with high probability. As $P_{C_b,d} > P_{C_a,d}$, the protocol succeeds (with high probability) in delivering the message m to a node that has a higher probability of encountering the destination node than the node a.

9.4.3 Security Analysis: Privacy

A node a reveals to an outsider node only the probability that at least one node in its community C_a will encounter the destination node. This probability is a function of the community as a whole and thus hides the probability of any individual node in the community encountering the destination. Moreover, the probability is computed within the community in a privacy preserving manner using the MDTN-Private-Probability protocol, thus individual probabilities also remain confidential from the nodes inside the community.

The reader may refer to Section 9.5 for the security analyses of the MDTN-Private-Probability protocol.

9.5 Privacy Preserving Computation of Probability

9.5.1 Protocol Description

We describe a protocol for computing the probability $P_{C,d}$, that at least one node a_i in community C in a mobile delay tolerant network will encounter the destination node d, as given in Equation (9.3).

Our protocol for private computation of probability is inspired by the protocols by Kreitz et al. [177], Sheikh and Mishra [262], and Hasan et al. [134, 138, 141]. However, our protocol addresses specific challenges in MDTNs listed below that the protocols by Kreitz et al. and Sheikh and Mishra do not. Moreover, unlike the protocol by Sheikh and Mishra, our protocol does not require Trusted Third Parties (TTPs).

The mobile delay tolerant network environment presents the following challenges: (1) Mobility implies that the nodes a node will encounter (neighbor nodes in the terminology of graph theory) are not known beforehand. (2) Connectivity is intermittent, messages arrive after long and variable delays, and message transmission is asynchronous.

Each node a_i in the set C participates in the protocol with a private number p_i as an input, where $p_i = 1 - P_{a_i,d}$, that is the complement of the probability that node a_i will encounter the destination node d. The nodes participating in the protocol learn the probability $P_{C,d}$, that at least one node a_i in community C in a mobile delay tolerant network will encounter the destination node d, as given in Equation (9.3). The protocol is specified in Figure 9.3.

The protocol is initiated by the leader node of a community given as the set of nodes C. The leader node floods an *init* message (Figure 9.3: protocol initiation: line 3) to all nodes. After a node receives the *init* message, it sends and receives a random number from each node belonging to C that it encounters (PROBINIT: lines 5 and 6). A node can send the *init* message to an encountered node if it has not received it yet (PROBINIT: lines 3 and 4). After a node has encountered k nodes (PROBINIT: lines 1 and 2), where k is a constant, the node sends a partial product to the leader node (PROBINIT: line 8). A node computes the partial product as the product of its private number and all random numbers received divided by the product of all random numbers sent (PROBINIT: line 7). The leader node maintains a running product of all partial products received (PROBPARTIAL: line 2). When the partial products are received from all nodes in C (PROBPARTIAL: line 3), the leader node computes the final product γ_C and floods $1 - \gamma_C$ to all nodes (PROBPARTIAL: line 4). $1 - \gamma_C = P_{C,d}$ is the required probability that at least one node a_i in community C will encounter the destination node d, as given in Equation (9.3).

9.5.2 The Value of Constant k

The choice of the value of constant k depends on the value of n, where $n = |C| \geq 3$. As stated in Section 9.3, we consider a community C to comprise of at least three nodes. Since a node in the protocol can exchange random numbers with at most all other nodes in its community, the interval of the constant k can be given as $[2, n)$, i.e., $2 \leq k < n$.

Additionally, when $k = 2$, whatever the value of n, these n nodes can always make a pair. Therefore, k can always be set as 2. When $2 < k < n$, according to the mechanism of our protocol, each node should exchange random numbers with k distinct nodes in its community. Hence, there are nk random numbers generated in each execution of our protocol. These nk random numbers should be divisible by $k + 1$. That is $n(k + 1 - 1) = n(k + 1) - n$ is divisible by $k + 1$. Therefore, the value of the constant k should also be compatible with: $n\%(k + 1) = 0$.

Summarizing, the value of the constant k should meet the following two requirements: 1) $2 \leq k < n$, and 2) $k = 2$ or $n\%(k + 1) = 0$.

Protocol: MDTN-Private-Probability
Participants: Nodes in a community denoted by the set C. One node in C is the leader node denoted by l.
Input: Each node a_i has a private input $p_i = 1 - P_{a_i,d}$.
Output: The nodes in C learn $P_{C,d} = 1 - \prod_{a_i \in C} p_i$.
Setup: (l, g) uniquely identifies a session of the protocol, where g is an integer. k is a constant such that $2 \leq k < n$, and $n\%(k+1) = 0$, where $n = |C|$. Nodes are not ordered, that is, a_i denotes any given node in C.
Events and Associated Actions:

leader node l initiates the protocol

```
1  $R \leftarrow \phi$
2  $\gamma_C \leftarrow 1$
3  $l$ floods $\langle$PROBINIT, $l, g\rangle$ to all nodes in $C$
```

node $a_i \in C$ receives $\langle$PROBINIT, $l, g\rangle$

```
1  for $j \leftarrow 1$ to $k$
2       do $a_i$ encounters node $a_j \in C$
3           if $a_j$ has not received $\langle$PROBINIT, $l, g\rangle$
4             then $a_i$ sends $\langle$PROBINIT, $l, g\rangle$ to $a_j$
5           $a_i$ sends a random number $r_{ij}$ to $a_j$
6           $a_i$ receives a random number $r_{ji}$ from $a_j$
7  $\gamma_i \leftarrow p_i(\prod_{j=1}^{k} r_{ji})/(\prod_{j=1}^{k} r_{ij})$
8  $a_i$ sends $\langle$PROBPARTIAL, $l, g, \gamma_i\rangle$ to $l$
```

leader node l receives $\langle$PROBPARTIAL, $l, g, \gamma_i\rangle$ from a_i

```
1  $R \leftarrow R \cup \{a_i\}$
2  $\gamma_C \leftarrow \gamma_C \times \gamma_i$
3  if $R = C$
4     then $P_{C,d} = 1 - \gamma_C$
5           $l$ floods $\langle$PROBFINAL, $l, g, P_{C,d}\rangle$ to all nodes in $C$
```

Figure 9.3: Protocol: MDTN-Private-Probability

9.5.3 Security Analysis: Correctness

The first challenge for the protocol due to the mobile delay tolerant network environment is that the nodes a node will encounter (neighbor nodes) are not known beforehand. To address this challenge, the protocol allows a node $a_i \in C$ to encounter any other k nodes in C (PROBINIT: lines 1 and 2). The encountered nodes, given as a_j, where $j \in \{1, 2, \ldots, k\}$, are considered as the neighbors of node a_i.

Each node $a_i \in C$ sends a random number r_{ij} to each encountered node a_j (PROBINIT: lines 5 and 6). Node a_i divides its product γ_i by r_{ij}, whereas node a_j multiplies its product γ_j by r_{ij} (PROBINIT: line 7). Each node a_i also multiplies its private value p_i to its product γ_i (PROBINIT: line 7). When the leader node computes $\gamma_C = \prod_{a_i \in C} \gamma_i$, the product γ_C is the required product $\prod_{a_i \in C} p_i$ because γ_i and γ_j are divided by and multiplied by r_{ij} respectively which results in being multiplied by the multiplicative identity 1 (PROBPARTIAL: lines 1 – 4).

The second set of related challenges of mobile delay tolerant network environments are as follows: connectivity is intermittent, messages arrive after long and variable delays, and message transmission is asynchronous. The following two elements of the protocol address this set of challenges: (1) The *init* message (PROBINIT) reaches all nodes in C with high probability and thus they all participate in the protocol. This is due to the assumption that a high probability exists of successful message delivery from any source node to any destination node in a community. (2) If a node $a_i \in C$ that has received the *init* message encounters a node $a_j \in C$ that has not yet received the *init* message then a_i sends a copy of the message to a_j to initiate it to the protocol (PROBINIT: lines 3 and 4). Nodes consider an encounter successful only if they exchange all messages according to the specification during their period of contact. Otherwise, they ignore any partial messages sent and received.

9.5.4 Security Analysis: Privacy

Let's consider a node $a_i \in C$. In an ideal protocol [121], the node would submit its private value p_i to a TTP. The TTP is considered trustworthy therefore it would not disclose the private value p_i of node a_i to any other party. It would only reveal the output of the protocol, which is the product of the private values received from all nodes in C, and consequently the probability as defined in Equation (9.3).

In the MDTN-Private-Probability protocol, node a_i discloses the following information: (1) One random number to each of the k nodes that it encounters after receiving the PROBINIT message. (2) The value γ_i to the leader node l as part of the PROBPARTIAL message. The value

γ_i is also revealed to the intermediate nodes that participate in the delivery of the message to the leader node.

The random numbers r_{ij}, where $j \in \{1, 2, \ldots, k\}$, are independent of p_i therefore they reveal no information about p_i.

$\gamma_i = p_i\theta_i$, where $\theta_i = (\prod_{j=1}^{k} r_{ji})/(\prod_{j=1}^{k} r_{ij})$. Let's assume that the interval of the random numbers is large compared to the interval of p_i and that the random numbers are distributed uniformly. This implies that the interval of θ_i is also large and that it is distributed uniformly. Thus there is high probability that the adversary can learn no information about p_i from γ_i.

The adversary can learn p_i if it learns θ_i in addition to γ_i. To learn θ_i, the adversary must learn all values r_{ij} and r_{ji}. This is possible only if all k nodes a_j that encountered node a_i are dishonest and collude to reveal all of their individual r_{ij} and r_{ji} values and consequently the value of θ_i.

As in the ideal protocol, the output of the protocol is the product of the private values of all nodes in C, and consequently the probability as defined in Equation (9.3). The MDTN-Private-Probability protocol thus does not reveal any more information about the private value p_i of node a_i than the ideal protocol if the following assumptions hold true: (1) the interval of the random numbers r_{ij} and r_{ji} is large compared to the interval of p_i and the random numbers are distributed uniformly, and (2) at least one of the k nodes that encountered node a_i is honest.

9.5.5 Security Analysis: Probability of Privacy Breach

As we described in the previous section, the adversary can learn p_i if all k nodes a_j that encountered node a_i and the leader node l are dishonest. Let P_D denote the probability that the private value of a node a_i is disclosed by the collusion of dishonest nodes. Let P_l denote the probability that the leader node l is dishonest. Let P_k denote the probability that the k encountered nodes of node a_i are dishonest. According to the above analysis, we can see that $P_D = P_l \times P_k$. Hence, P_D depends on the number of nodes in community C, the value of k, and the number of dishonest nodes in community C. Let's denote the number of dishonest nodes as h, where $0 \leq h \leq n-1$.

Hence, if we assume that the leader l is randomly chosen from the community, then P_l can be expressed as Equation (9.4).

$$P_l = \frac{h}{n-1} \tag{9.4}$$

Moreover, due to the random mobility model, we can assume that the encounters are random and cannot be scripted by the adversary. According to the values of k, h, and n, the analysis of P_k can be divided into the following two cases: (1) $0 < h < k$; (2) $k \leq h \leq n-1$. In the first case, the private information of node a_i cannot be learned by the adversary node, i.e., $P_k = 0$. In the second case, there are C_h^k combinations that all the k encountered nodes met by node a_i are dishonest, while there are C_{n-1}^k combinations that node a_i encounters k distinguish nodes inside community C, i.e., $P_k = C_h^k / C_{n-1}^k$. Hence,

$$P_k = \begin{cases} 0, & \text{if } 0 \leq h < k \\ \frac{C_h^k}{C_{n-1}^k}, & \text{if } k \leq h \leq n-1 \end{cases} \tag{9.5}$$

Combining (9.4) and (9.5), the probability P_D can then be expressed as Equation (9.6).

$$P_D = \begin{cases} 0, & \text{if } 0 \leq h < k \\ \frac{h}{n-1} \times \frac{C_h^k}{C_{n-1}^k}, & \text{if } k \leq h \leq n-1 \end{cases} \tag{9.6}$$

In addition, one unavoidable side-effect of the protocol is that the adversary learns that node a_i's probability (i.e., $P_{a_i,d}$) of encountering the destination node d is not higher than $P_{C,d}$, since $P_{C,d} = P(\bigcup_{x=1}^{n} P_{a_x,d}) \geq P_{a_i,d}$, where $n = |C|$, $1 \leq i \leq n$. However, in contrast to the previous protocol (3PR), the 4PR protocol does not reveal the true upper bound of any individual node.

9.5.6 Complexity Analysis

In this section, we discuss the complexity or the overhead of computing $P_{C,d}$ using the MDTN-Private-Probability protocol. According to the mechanism of MDTN-Private-Probability protocol (see Figure 9.3), the information, which is utilized to compute $P_{C,d}$ in a given community C, is transmitted between nodes in the following four sub-processes: (1) the leader node floods the *PROBINIT* message to all other nodes in community C; (2) each node in community C exchanges

Table 9.1: Protocol MDTN-Private-Probability – Complexity

Sub-process	**(1)**	**(2)**	**(3)**	**(4)**
No. of bits	$3\beta(n-1)$	$k\beta n$	$4\beta(n-1)$	$4\beta(n-1)$
Complexity	$O(\beta n)$	$O(k\beta n)$	$O(\beta n)$	$O(\beta n)$

k random values (with the first k distinct nodes in community C; (3) each node directly sends the mixed value ($PROBPARTIAL$) to the leader node; and (4) the leader node floods the final result ($PROBFINAL$) to all other nodes.

Let's consider that each field (i.e., an integer or a real) of each message occupies β bits (i.e., of the same size). In the sub-processes (1), all the nodes in community C (except the leader node) get a copy of the message which contains three fields. That is, $n-1$ messages exchanged between nodes. In the sub-process (2), each of the nodes in community C sends k messages which contains only one field to the first k community members. Therefore, there are kn messages exchanged in this sub-process. In sub-process (3), all the nodes in community C (except the leader node) sends a message with four fields to the leader node. That is, $n-1$ messages exchanged in the sub-process (3). Since sub-process (4) utilizes the same method as in sub-process (1) to disseminate messages which contain four fields, the amount of messages transmitted between nodes in this sub-process is $n-1$. Consequently, the overhead of computing $P_{C,d}$ in community C is $\beta((k+11)n--11)$. That is, the protocol requires $O(k\beta n)$ bits to be exchanged, where k and β are constants, and $n=|C|$. Table 9.1 represents an analysis of the communication complexity of the MDTN-Private-Probability protocol.

9.6 Performance Evaluation

In this section, we first present a comparison between private probability and private maximum, the background protocols employed by 4PR and 3PR, respectively. We then present the simulation settings and the utilized mobility model for our experimental performance evaluation in Sections 9.6.2 and 9.6.3, respectively. Next, we introduce the routing protocols against which we compare the performance of 4PR and the performance metrics that we use in Sections 9.6.4 and 9.6.5, respectively. Finally, we present the results of our experiments in Section 9.6.6.

9.6.1 Private Probability vs. Private Maximum

Private probability and private maximum are computed for 4PR and 3PR respectively by the nodes in a community in the background independently of the routing protocols. In this section, we compare the efficiency of these background protocols. We observe that computing private probability, as presented in this work, is significantly more efficient than computing private maximum, as was proposed previously for the 3PR protocol [141].

In order to compute the value of maximum in a privacy preserving manner in [141], the protocol needs to run $2+\lambda$ (where $\lambda \geq 7$) rounds of another privacy preserving protocol (named private_sum), which computes the sum of the probability that nodes in a community will encounter a destination node. In each round of the private_sum protocol, kN messages are exchanged among the nodes in a community, where N is the number of nodes in the community, and k is a constant with $2 \leq k < N$.

In comparison, the protocol presented in the previous section in this work for computing probability in a privacy preserving manner requires only one round of kN messages to be exchanged among the nodes in a community. The order of the size of the messages being similar in the two protocols, the private probability protocol used for 4PR is at least 9 times more efficient than the private maximum protocol used for 3PR in terms of messages exchanged and the bandwidth utilized.

Since the private probability and private maximum protocols are executed in the background, they do not have a direct impact on the performance of the two routing protocols (as evident in the subsequent experimental evaluation). However, considering that these background protocols need to be executed regularly, and that private probability is significantly more efficient than private maximum, the 4PR approach has the potential to globally conserve substantial network resources.

9.6.2 Simulation Settings

We have implemented 4PR as a module of the Opportunistic Network Environment simulator (ONE) [167]. We summarize the simulation parameters that we used in Table 9.2.

Table 9.2: Parameter settings

Parameter Name	**Value**
Simulation area	2000 m × 1500 m
Transmission range	10 m
Simulation duration	13 h + TTL
Warm-up period	1 hour
Message generation rate	1 message per 30 seconds
Number of communities	12
Number of nodes in a community	from 10 to 50
Node speed	1.34 m/s
p_l	0.8
p_r	0.2

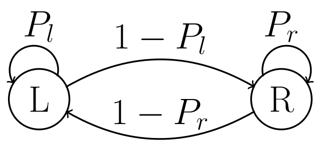


Figure 9.4: Community-based Mobility Model

We have used a simulation area of 2000 m × 1500 m. This area is equally divided into twelve regions each measuring 500 m × 500 m. In each region we initially deploy a varying number of nodes (from ten to fifty). Each node considers the region in which it has been deployed as its *local region*. According to the mobility that model we used, further described below, a node is more likely to visit its local region than other places. Nodes associated to a region constitute a community. This simulation scenario is very similar to the one used in PRoPHET [191].

The communication between nodes is performed using the Bluetooth protocol since modern mobile devices are commonly equipped with this technology. According to the specification of Bluetooth version 2.0 [167], the transmission range and bandwidth are set as 10 m and 2 Mb/s, respectively. Furthermore, the speed of nodes is set to 1.34 m/s, since this is an average human walking speed [171]. Each experiment that we run lasts approximately thirteen hours (simulation time). The first hour is a warm up period during which no message is generated. After this period, every thirty seconds, a random node sends a message to a random destination node. We have considered only messages for which the source and the destination belong to different communities.

9.6.3 Mobility Model

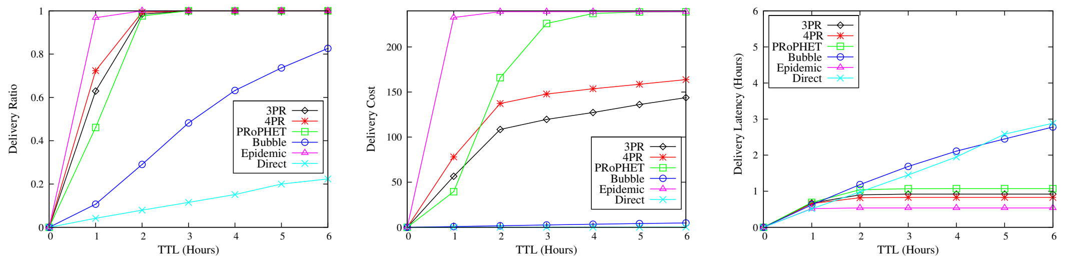
In our evaluation, we adopt the community-based mobility model proposed in [267], which has been widely utilized for the evaluation of community-based routing protocols [89, 269]. In this mobility model, each community is associated with a geographical area. The movement of node i, which belongs to the community C_i consists of a sequence of *local* and *roaming* epochs. A local epoch is a random direction movement restricted inside the area associated with the community C_i. A roaming epoch is a random direction movement inside the entire network. If the previous epoch of a node i was a local one, the next epoch is a local one with probability p_l, or a roaming epoch with probability $1 - p_l$. Similarly, if the previous epoch of node i was a roaming one, the next epoch is a roaming one with probability p_r, or a local one with probability $1 - p_r$. The state transition between local and roaming epochs is shown in Figure 9.4. In our simulations, we adopt the same values for p_l and p_r as in [191], i.e., p_l=0.8 and p_r=0.2.

9.6.4 Routing Protocols

We have compared the performance of 4PR against the following protocols:

Epidemic: in this protocol, a node forwards a copy of each unexpired message it holds to every node it encounters, which does not already have a copy of the message. Epidemic routing achieves the upper bounds of delivery ratio and delivery cost, and achieves the lower bound of delivery latency.

Direct: in this protocol, the source node only forwards the message to the destination node. Contrary to Epidemic, Direct routing achieves the lower bounds of delivery ratio and delivery cost, and achieves the upper bound of delivery latency.



(a) Delivery Ratio (b) Delivery Cost (c) Delivery Latency

Figure 9.5: (a) delivery ratio, (b) delivery cost, and (c) delivery latency w.r.t. the increasing TTL of messages.

PRoPHET: in this protocol, a node forwards a copy of a message that it holds to a node that it encounters, only if the latter has a higher probability of encountering the destination node of the message. The parameters of the protocol are set as described in [191]. PRoPHET is a well known prediction-based routing protocol.

Bubble: this is a community-based protocol that utilizes social information about nodes, such as their centrality and the community to which they belong.

3PR: in this protocol, the message forwarding decision is made by comparing the maximum probability that a node in the community of a potential intermediate node will encounter the destination node. The parameters of the protocol are set as described in [141].

9.6.5 Performance Metrics

To evaluate 4PR we used three well known metrics: the delivery ratio, the delivery cost and the delivery latency defined as follows.

Delivery ratio: is the proportion of messages that have been delivered out of the total unique messages created.

Delivery cost: is the total number of messages transmitted in the simulation. To normalize this, we divide it by the total number of unique messages created.

Delivery latency: is the average time needed to finish transmitting messages to their destinations.

9.6.6 Performance Results

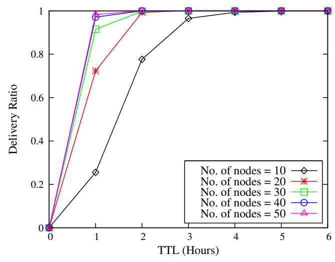
We performed two experiments. First, we compared the performance of 4PR against the protocols introduced above, with respect to the above three performance metrics. We then analyze the impact of the community size on the performance of 4PR.

Performance Comparison of Routing Protocols

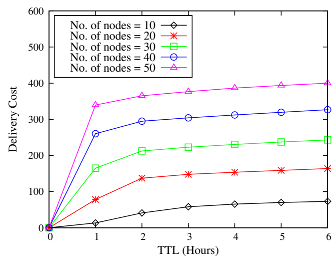
Figure 9.5a shows the delivery ratio of the compared protocols as a function of the Time-To-Live (TTL) of the generated messages. As expected, Epidemic and Direct achieve the best and worse delivery ratio, respectively, for all values of TTL. We also observe that 4PR achieves a better delivery ratio than PRoPHET and 3PR when the TTL is less then 2 hours, and achieves a similar delivery ratio to that of PRoPHET and 3PR when the TTL is greater than 2 hours. Finally, 4PR has a much higher delivery ratio than Bubble. The difference between the performance of the two protocols rises up to 70.29% for a TTL of 2 hours. This is because 4PR floods a message inside the communities which are on the path from the community of its source node to the community of its destination node.

Figure 9.5b, shows the delivery cost of the compared routing protocols. As expected, Epidemic and Direct have the highest and lowest delivery cost, respectively, whatever the value of TTL. Compared to other protocols, Bubble has a low delivery cost, which remains stable when the TTL increases. The delivery cost of 4PR is higher than that of Bubble and 3PR, but much lower than that of PRoPHET.

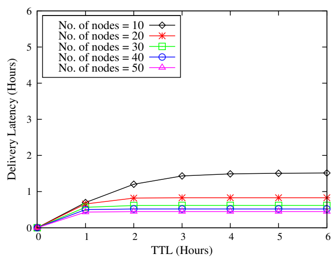
Figure 9.5c shows the delivery latency of the compared routing protocols. Epidemic has the lowest delivery latency, whatever the TTL. Further, 4PR follows the same trend as Epidemic with



(a) Delivery Ratio

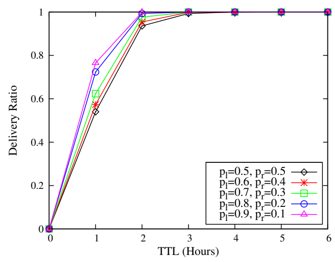


(b) Delivery Cost

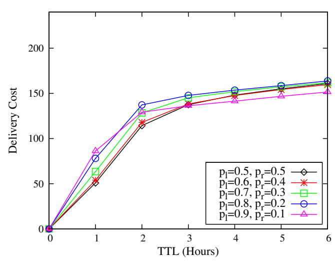


(c) Delivery Latency

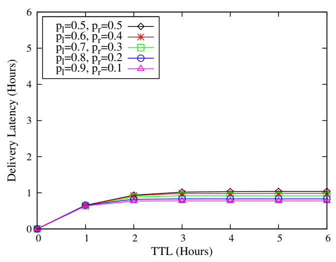
Figure 9.6: (a) delivery ratio, (b) delivery cost, and (c) delivery latency w.r.t. the increasing size of communities.



(a) Delivery Ratio



(b) Delivery Cost



(c) Delivery Latency

Figure 9.7: The impact of the settings of the mobility model on the (a) delivery ratio, (b) delivery cost, and (c) delivery latency of 4PR

higher latencies (around 0.29 hour). 3PR and PRoPHET achieve a little higher delivery latency than 4PR. The performance of Bubble and Direct increases linearly with the increase of the TTL.

Influence of the Number of Nodes in a Community

In order to investigate the impact of the number of nodes in each community on the routing performance of our protocol, we run an experiment in which we vary the number of nodes in each community from 10 to 50.

Figure 9.6a, 9.6b and 9.6c show the impact of the increasing community size on the delivery ratio, the delivery cost and the delivery latency, respectively of the 4PR protocol. The results show that the larger the communities, the higher the delivery ratio and cost and the lower the delivery latency. Since 4PR floods a message inside the community of the message carriers, the delivery cost increases as the communities become larger. However, more message copies increase the delivery probability and reduce the delivery latency.

Impact of the Settings of the Mobility Model

In this section, we investigate the impact of the settings of the adopted mobility model on the routing performance of 4PR. We run an experiment in which we vary the value of p_l from 0.5 to 0.9 and set the value of p_r as $1 - p_l$.

First, we look at the impact of the settings of the adopted mobility model on the delivery ratio. As shown in Figure 9.7a, we can observe that 4PR achieves similar results with different settings of p_l and p_r. The performance of delivery ratio increases as the increment of the value of p_l when the TTL is not greater than 3 hours. The performance of delivery ratio with different settings is the same, when the TTL is greater than 3 hours. Since 4PR floods messages inside a community, under the pre-condition that messages can be transferred among communities, the higher the probability that a node stays inside its community, the higher probability that the node gets a message flooded inside its community.

Next, we compare the delivery cost of 4PR with different settings of the adopted mobility model. From the results illustrated in Figure 9.7b, we can observe that the performance of delivery cost increases as the value of p_l increases when the TTL is not greater than 3 hours. When the TTL is greater than 3 hours, the performance of delivery cost decreases as the increment of the value of p_l. This is because that the higher probability that a node stays inside its community, the higher probability that the node gets a message flooded inside its community. In our case, for a given message, most of the nodes on the routing path from the community of its source node to the community of its destination node can get a copy of the message within 3 hours. Therefore, when the TTL is greater than 3 hours, the delivery cost increases slowly for the simulations with high values of p_l. This is consistent with the results of the delivery ratio.

Lastly, we investigate the results of delivery latency of 4PR with different settings of the adopted mobility model. As shown in Figure 9.7c, we can see that the delivery latency decreases as the increment of p_l. For each setting, the delivery latency increases as the TTL increases, when the TTL is less than 3 hours; the delivery latency stays the same as the TTL increase, when the TTL is greater than 3 hours. For the case that the TTL is less than 3 hours, the messages that need more time can be delivered as the TTL increases. As for the case where the TTL is greater than 3 hours, the latency stays the same, since the messages are delivered within 3 hours. Note that this is consistent with the results of the delivery ratio.

9.7 Conclusion

This work describes the 4PR protocol, which provides privacy preserving probabilistic prediction-based routing in mobile delay tolerant networks. 4PR is similar to prior prediction-based protocols (e.g., PRoPHET and Bubble), which take advantage of the mobility patterns of nodes to route messages. Our experimental evaluation using a well established community-based mobility model demonstrates that 4PR is comparable to the above noted protocols in terms of performance. Yet, 4PR preserves the privacy of nodes by hiding their individual mobility patterns, whereas the prior protocols do not.

The 4PR protocol is the successor of our 3PR protocol, which to the best of our knowledge, was the first protocol to hide the encounter probabilities of nodes in MDTNs. However, 4PR differs fundamentally from 3PR in how messages are exchanged and the protocols that execute in the background. 4PR's approach gains multiple advantages over 3PR, which include 1) the upper bound of encounter probabilities is not divulged, thus better privacy preservation; 2) private computation of probability requires a single round of computation, whereas private maximum in 3PR required multiple rounds; 3) the probability of at least one node in the community encountering the destination (as in 4PR) is a more accurate measure for routing path prediction than the maximum probability in the community (as in 3PR).

We foresee three opportunities for future work. First, we would like to reinforce the protocol for preservation of privacy in the malicious adversarial model, where nodes may take disruptive actions such as dropping messages, modifying the protocol, etc. Second, we would like to study the effect of conditions such as network churn and overlapping communities on the protocol. Third, we would like to analyze the energy consumption of privacy preserving routing protocols in MDTNs.

Part IV

Privacy Preservation in Financial Networks

Chapter 10

Privacy Considerations for a Decentralized Lending Platform

There are many Decentralized Finance (DeFi) Peer-to-Peer (P2P) lending platforms that offer users to obtain a loan by committing a collateral or by calculating a "credit score", which is based on factors such as the users' credit history. However, the requirements of collateral and credit history are quite burdensome for some users. Nowadays, with more than 55% of the global population using social media, there is a lot of publicly available personal data [15]. This data could be used as an alternative risk mitigator for lending. There are many inferences that can be drawn from the users' social media accounts about their professional behavior and reliability, allowing us to derive the users' social trustworthiness. We propose to calculate a "social score" based on the social media data of a user. Our contribution is to develop an Ethereum blockchain-enabled fully decentralized lending platform that relies on this score. This platform could give users a chance for a loan even if they do not have a collateral or a sufficient credit score. Furthermore, we discuss privacy considerations for our platform and present an enhanced version that protects the borrower's privacy.

This chapter is an adapted version of the article: "Privacy Considerations for a Decentralized Finance (DeFi) Loans Platform." J. Hartmann and O. Hasan. *Cluster Computing (Springer)*. 2022. https://doi.org/10.1007/s10586-022-03772-3. This work was carried out as part of the Master's research project of J. Hartmann and its subsequent follow-up.

The Ph.D. work of W. Uriawan was co-supervised with L. Brunie and Y. Badr on the related topic of "Trustworthiness-based Personal Lending on Blockchain". The work by J. Hartmann took place independently and particularly addressed the issue of privacy.

10.1 Introduction

The concept of Decentralized Finance (DeFi) promotes an open financial system where financial services can be provided and accessed without dependence on intermediaries or central authorities. Peer-to-Peer (P2P) lending is one of the main financial services that can be enabled by DeFi technologies. In this work, we look at how social network data can be used to support peer-to-peer lending while taking privacy considerations into account [15, 19].

Peer-to-peer lending allows a borrower to receive a loan directly from a single or multiple individual lenders. The first peer-to-peer lending platform Zopa [20] went online in 2005. The peer-to-peer lending market has been continuously growing since then and is predicted to keep growing in the future [9]. Some other well known P2P lending platforms are CoinLoan [11], Inlock [16], Prosper [18], and Lending Club [17]. Peer-to-peer lending has several advantages in contrast to traditional lending. It dispenses with middlemen such as financial institutions. The lending platform itself sets the conditions and enables the transactions. Not having middlemen saves time and money, which often allows the platform to offer better rates.

An additional development in recent years is that peer-to-peer lending platforms are based on blockchain and are using smart contracts. This development brings more transparency. This type of lending is fully in line with the concept of decentralized finance. However, since there are no middlemen verifying a potential borrowers' financial situation, they need to prove to the lenders that they are credit worthy. Loans can be secured or unsecured. They are received by either depositing a collateral or by calculating a credit score to prove one's creditworthiness.

10.1.1 Secured Lending

The term “secured lending” describes a way of lending, where the loan is secured with a collateral. A collateral is a valuable asset (for example, a mortgage on the borrower’s house, investments in cryptocurrencies, etc.) which the borrower has to give as insurance for the loan. After receiving the loan, the borrower has to pay back the money within a certain time. If the borrower is unable to pay, the debt is deducted from his collateral. Secured lending carries an element of risk for the borrower: If he cannot pay back the money, he loses his asset. Moreover, the borrower needs to be in possession of a suitable asset in order to qualify for a loan. The lender on the other hand is promised to get his money back. He is therefore often willing to offer better interest rates, which can be an advantage for the borrower in this kind of lending.

10.1.2 Unsecured Lending

Unsecured loans or personal loans work without a collateral. Collateral has two main problems: Firstly, people may not trust in the lending platform enough to deposit their asset. Secondly, people do not always have the money or property required for a collateral. They can have a good income and be a reliable person but without a collateral they might still not be qualified for a secured loan. Instead of taking a collateral as insurance, unsecured loans rely on a creditworthiness system, which is mostly based on a “credit score”. A well-known credit score is the FICO (Fair Isaac Corporation) score [14]. It determines the creditworthiness of a potential borrower by using a fixed formula, which takes into account aspects such as the borrower’s payment history, available credit and age. A penalty has to be paid if the monthly payment is not balanced on time. When the borrower defaults paying back his loan, he loses points of his credit score, but not a collateral. Therefore, from the borrower’s point of view, unsecured loans are less risky, but can be linked to higher interest rates. A common example for unsecured loans are student loans.

10.1.3 Peer-to-Peer Lending

In peer-to-peer lending, there exist secured loans as well as unsecured loans. However, while in traditional lending there is mostly some kind of financial institution participating in the process, peer-to-peer lending offers borrowers and lenders to connect directly without such an intermediary. This can translate into lower or no fees and there is no longer a single point of failure. However, since there is no borrower creditworthiness evaluation carried out by a third party, the individual lender himself is responsible for determining whether a person can be trusted to pay back their debts. Peer-to-peer lending platforms are online platforms that offer to match people that want to lend money as a form of investment with people who want to borrow money. A “peer” can also be a company or a group that is in need of a loan. An example of a loan to an individual could be a payday loan, whereas companies might need a loan for commercial reasons or to expand their business.

10.1.4 Our Approach

The goal of this work is to develop a new approach to calculate the borrowers’ trustworthiness based on their social capital, which does not depend on a collateral or the credit history of the user. This approach can stand on its own or it can be used in addition to traditional concepts such as collateral and credit score. The objective is to minimize the risk that the borrower defaults on the loan. It would give those users a chance for a loan who do not have a high enough credit score or the resources to deposit a sufficiently valuable collateral. This approach is based on the “social capital” theory. In this work, we develop a prototype of a decentralized finance peer-to-peer lending platform built on the Ethereum blockchain. The creditworthiness of the users on this platform is represented by a social score, which is calculated by analyzing the users’ social media accounts.

There is a large amount of personal data that is available on social network accounts of many people. In the subsequent sections, we discuss how this data can be used to compute the social score of a user. The available data includes biographical information such as name, email address, date of birth, etc., as well as social information such as the number of followers or friends, shared posts, etc. Moreover, in case a user has multiple social network accounts, the information available on different accounts can be compared and co-related to derive inferences about the user as well. In view of “social capital” theory, this multitude of information could portray the trustworthiness of a user, which we could use as an indicator of the user’s disposition to repay loans.

We note that using personal information leads to privacy concerns. Therefore, in this work, one of our main objectives is to develop a privacy-preserving loans platform. This privacy-preserving platform aims to prevent disclosure of personal information such as the user's name, email address, date of birth, precise number of friends, etc. In order to achieve privacy, we develop mechanisms using cryptographic building blocks such as homomorphic cryptosystems, zero-knowledge proofs, and cryptographic hash functions.

10.1.5 Contributions

This work makes the following contributions:

- A new approach to calculate the users' trustworthiness on peer-to-peer lending platforms using the information that can be retrieved from the users' social network accounts instead of relying on collateral and credit scores that can be problematic for the users.
- A brief analysis of the various types of lending and a look at their advantages and disadvantages.
- A description of some of the popular existing online peer-to-peer lending platforms and discussions on how they operate with or without collateral.
- The proposal of an enhanced version of the lending platform that takes privacy considerations into account by using cryptographic building blocks such as zero-knowledge proofs and cryptographic hash functions.
- The development of a prototype of an Ethereum decentralized application that implements the proposed social score formula in a smart contract.
- Quantification of the amount of Ethereum gas that is consumed by the deployment of the smart contract.
- Experiments on a real social network dataset that demonstrate how we can use analysis of social network data to determine optimal thresholds for a platform in production.

10.1.6 Outline

In Section 10.2, an introduction to three successful peer-to-peer lending platforms and their methods to ensure the borrowers' creditworthiness is given. In Section 10.3, we present the fundamentals of the social capital theory. In Section 10.4, the idea developed in this work based on the calculation of a social score is presented. The implementation details of a prototype are discussed in Section 10.5. In Section 10.6, we discuss privacy considerations for our lending platform and propose a solution for preserving the privacy of borrowers. Section 10.7 comprises of the evaluation details. This is followed by the conclusion.

10.2 Related Work

In this section, we describe three existing online peer-to-peer lending platforms. CoinLoan [11] and Inlock [16] both offer secured loans whereas Prosper [18] offers personal loans. The fundamental principle is that one peer lends a loan and another peer borrows a loan. All three platforms include interest fees, which the borrower has to pay to the lender and an origination fee, which the borrower pays to the platform for using the service.

10.2.1 CoinLoan

CoinLoan [11] is a peer-to-peer platform founded in 2017. The advantage of borrowing money with CoinLoan is to get a loan right away without having to provide anything except for a collateral in cryptocurrency. The collateral amount, the interest rate and the origination fee are calculated from the user's inputs. The borrowing money function on CoinLoan is only useful for people owning cryptomoney. Moreover, once a user receives a loan, his collateral is blocked until he has paid off his debts. During this time, the user is not able to sell the deposited cryptocurrency, which they might want to do in case the cryptocurrency is facing heavy depreciation. If the user does not pay back on time, the owed amount will be taken from his deposit of cryptomoney.

10.2.2 Inlock

Inlock [16] is another peer-to-peer lending platform that was founded in 2017. Again, there are no options to prove one's trustworthiness other than to give a collateral. The collateral has to be paid in the form of cryptomoney. Inlock currently supports only four cryptocurrencies: Bitcoin, Ether, Litecoin and Binance Coin. There is a 110% over-collateralization rate along with a universal collateral termination level. Once the collateral decreases below that level, the debts will automatically be paid off by Inlock using the deposited collateral. Thus, the user has to be careful and keep an eye on falling market values of the cryptocurrency.

10.2.3 Prosper

Prosper [18] is a peer-to-peer lending platform that was founded in 2005. Unlike Inlock and CoinLoan, it does not offer secured, but only personal and hence unsecured loans. Although the user has to give personal data, applying for a loan is a simple and quick process. Since there is no collateral, users do not need to deposit anything and therefore they do not risk to lose their collateral. On the other hand, penalty fees can rise quickly. For not paying back on time, the borrower has to pay USD 15 or 5% of the outstanding debts. They also lose credit score points. The origination fee is significantly higher than it is on other lending platforms such as CoinLoan. Prosper is also restricted in the kinds of loan they offer. As an example, they exclude student loans and other educational loans.

10.3 Social Capital

We now introduce a concept on which our proposed score is based. According to Rene Dubos in his book "Social Capital: Theory and Research" the "premise behind the notion of social capital is rather simple and straightforward: investment in social relations with expected returns" [106]. He gives four reasons for why "embedded resources in social networks will enhance the outcome of actions" [106]. Firstly, connections can help to get information and information can translate into opportunities. A good example where social connections are often useful is job hunting. Secondly, Dubos claims that having social connections may also have a positive impact on decisions involving the individual, such as discussions about promotions. The decision making process can be strongly influenced by a person putting in a good word for the individual. Thirdly, he claims that companies might value a person's social capital on top of his personal capital. "The individual can provide "added" resources beyond his / her personal capital, some of which may be useful to the organization" [106]. The fourth reason Dubos states is that being well connected provides both "emotional support" [106] and "public acknowledgment of one's claim to certain resources" [106].

Robert Putnam claimed that "economic performance as a whole is better in well-connected societies than in poorly connected ones" [111]. This claim triggered many studies on the topic. A Swedish study on unemployed Swedes resulted in the conclusion that "network size had a considerable positive impact on the likelihood of finding work, far outweighing the official employment agency" [111]. Another study, performed in Germany, revealed "that engagement in a range of social activities is positively linked with job-finding among the unemployed" [111].

In this work, we aim to use social media network information of a person to determine their social capital and to draw conclusions about possible connections in other areas like financial behavior. "The central idea of social capital is that social networks are a valuable asset" [111]. Being well-connected on social media brings advantages similar to real world connections. It helps the user to stay informed and to find possible opportunities. Further, a social media entry, such as a picture about the individual participating in a certain event, could help the individual to receive recognition. It could be a conversation opener and enable them to establish new contacts. The contacts might be useful and might put in a good word for the individual at some point.

Another widely investigated and supported claim of Putnam is that "higher levels of social capital ... translate into lower levels of crime" [111]. Further studies have demonstrated that there is higher criminality in neighborhoods where people live rather anonymously and do not maintain contact with their neighbors [111]. Social connections have a large impact on people's well-being, but there seem to be more benefits. "There appear to be clear and often strong positive links between social capital and educational attainment, economic success, health and freedom from crime." [111].

10.4 Our Decentralized Lending Platform

In this section, we propose a new approach for decentralized lending: unsecured loans based on the users' social score, instead of their credit score.

10.4.1 Creditworthiness Depending on a Social Score

The social score presented in this work is calculated based on one or several social media accounts of the user. Our algorithm analyzes the user's accounts to determine his personal social score. The algorithm is based on six hypotheses that estimate the trustworthiness of a user.

Hypothesis 1: Users who add their social media account have less to hide Our social media account says a lot about us: who are our friends, what are our likes and dislikes, what are our ambitions, etc. The social media account may also provide some personal data, such as our age, current location, and our profession. Most people are aware of the fact that their social media account can reveal a lot of information. Therefore, if they have something to hide, for example a bad habit like gambling, they will hesitate to connect their social media to a peer-to-peer lending platform, that calculates a score derived from their social media information. As we saw in the neighborhood example [111] given in the subsection about social capital: Criminality is lower when anonymity is absent and people are part of a community. Based on this theory, the risk to default may be lower for users who add their social media accounts and therefore lose their anonymity. As a result, one may assume that users who disclose their social media account have less to hide and may therefore be considered trustworthy.

Hypothesis 2: The more the user is willing to disclose about himself, the more trustworthy he is There are a lot of social media platforms these days. Currently, three important platforms are Facebook, Instagram and LinkedIn. All three platforms together cover a wide range of information about a user, which include both private and professional information. The more accounts from different platforms a user is willing to disclose, the more information about himself he is ready to provide. By giving this information about himself, the user proves once again his willingness to give up his anonymity. As already mentioned in hypothesis 1, we assume that users who give up their anonymity have less to hide and may therefore be considered trustworthy.

Hypothesis 3: Trustworthy users have authentic social media profiles To avoid that users simply disclose some accounts they just created or some fake account for the pure purpose of improving their social score, the authenticity needs to be checked. There are many indicators when it comes to identifying fake accounts. These include posting original pictures, having a significant number of mutual friends and followers, and having a non-recent date of creation of the account. For example, on Instagram, fake accounts do not have a lot of mutual friends and typically follow more people than they are followed by. According to [130], fake profiles have around 30 times as many friends as followers.

Hypothesis 4: The bigger the social network and activity, the more credit worthy is the person The Swedish study concluded that "network size had a considerable positive impact on the likelihood of finding work" [111]. Therefore, users with more social contacts and thus more followers and friends on social media, are less likely to get stuck in unemployment. Also, the well-being of people strongly depends on their social network and on how connected they are. The more social activity a user has, the higher is his social capital. "There appear to be clear and often strong positive links between social capital and educational attainment, economic success, health and freedom from crime." [111]. Therefore, more friends and connections as well as posts lead to a higher social score.

Hypothesis 5: People who make truthful statements are trustworthy "Interpersonal trust is fundamental for the effective functioning of social interactions as well as of society as a whole. It has been found to be related to many societal outcomes such as lower corruption perception" [259]. Trust is fundamental in peer-to-peer lending as well. A lender needs to trust in the borrower's good will to pay back the loan. To be trusted to get a loan, a user's honesty is tested. Therefore, before connecting with the social media accounts, the user will be asked to give three personal information: the full name, date of birth, and email address. When connecting the social media accounts, this information will be compared. If the information are corroborated by multiple social media accounts, this is taken as an indication of the user's honesty and openness.

Hypothesis 6: Consistency is a sign for stability If a user agrees to connect multiple social media accounts, the data of these accounts can be checked for consistency. Being friends with the same people and showing a similar profile on different social media platforms indicates that these accounts represent one and the same person.

10.4.2 Social Scoring in Peer-to-Peer Lending

The mentioned hypotheses need to be converted into variables and formulas that we can calculate our social score with. Each of those variables will have an impact on the final social score. The final social score will be received by calculating the average of the individual social media platforms accounts plus a bonus for disclosing more social media accounts.

$$DISC_x = OPEN_x * AUTH_x \tag{10.1}$$

The first variable is $DISC_x$, which stands for disclosure of the user's account from the platform x. Here, x could be for Facebook, Instagram, or LinkedIn. The variable can vary between zero and one, depending on the user's openness to disclose his social media account x and that account's authenticity. The $OPEN_x$ variable can only take the value one or zero. If the account x is disclosed by the user, $OPEN_x$ equals 1, whereas if the user doesn't disclose his account x, the $OPEN_x$ variable is zero and therefore $DISC_x$ equals 0. The $AUTH_x$ variable describes the authenticity of the disclosed account x. It varies between 0 and 1. It is zero, if an account contains no information at all or if it is classified as fake. It is one when an account is authentic. The variation between 0 and 1 is incremental according to a value pre-defined by the platform operator. For example, one could assign 0.2 for the first 100 followers / friends, 0.4 for the first 1000, and so on. The maximum value of $DISC_x$ is therefore one if the user discloses his account x ($OPEN_x = 1$) and the account is classified as authentic ($AUTH_x = 1$).

The precise function for computing $AUTH_x$ is to be defined by the platform operator. The value of $AUTH_x$ is computed based on the number of friends of a user. In Equation 10.2, we give an example of the function that could be used, where f = the number of friends of the user.

$$AUTH_x = \begin{cases} 0 & \text{if } f < 100 \\ 0.2 & \text{if } f \geq 100 \text{ and } f < 1000 \\ 0.4 & \text{if } f \geq 1000 \text{ and } f < 10000 \\ 0.6 & \text{if } f \geq 10000 \text{ and } f < 100000 \\ 0.8 & \text{if } f \geq 100000 \text{ and } f < 1000000 \\ 1 & \text{if } f \geq 1000000 \end{cases} \tag{10.2}$$

We present a function below to calculate the Social Score of a person, which is abbreviated by SC_x.

$$SC_x = DISC_x * HON_x \tag{10.3}$$

HON_x is short for honesty. The entered user data on our peer-to-peer lending platform consists of the name, the email address and the date of birth. This data is compared to the data available on the social media platform x. If none of the information match, HON_x equals zero. If only the email address matches, the honesty-value is 20. The same applies if only the name matches. A matching birth date adds 10 to HON_x. The maximum value for HON_x is 50 when all three information match.

$$bonus = n * 2 * CONS \tag{10.4}$$

A bonus is given on the final social score depending on the number of disclosed accounts n. The maximal number of disclosed accounts is 5. The variable $CONS$ stands for consistency. It varies between 0 and 5 and conveys the concordance between the different disclosed accounts. When the same username is used among the social media accounts of the different portals, $CONS$ increases by 2. We increase the bonus by 2 again if the email address matches. For the same birthday information, the bonus is increased by 1 for consistency. The maximal number of points reached by the bonus is therefore 50.

$$SC = \frac{(SC_1 + SC_2 + ... + SC_n)}{n} + bonus \tag{10.5}$$

The final equation is composed of the sum of the single social score's average and the bonus. The complete formula would look like this:

$$SC = \frac{(O_1 * A_1 * H_1 + ... + O_n * A_n * H_n)}{n} + n * 2 * C \tag{10.6}$$

In this equation, O is short for open [OPEN], A stands for authenticity [AUTH], H for honesty [HON] and C for consistency [CONS]. If the user does not disclose any accounts, every $OPEN$ variable and n is zero. No social media based observation can be made as no data can be accessed. Since all the partial terms $SC_1, SC_2, ..., SC_n$ contain a multiplication by zero, they all end up having the value 0. For $n = 0$, the bonus equals zero, and as a consequence, the final social score (SC) will add up to zero as well. Table 10.1 summarizes the correspondence of the variables in the social score formula to the hypotheses listed in Section 10.4.1.

Hypothesis	**Variable**	**Description**
1	$OPEN_x$	Whether a social media account on the platform x is disclosed or not.
2	n	The number of accounts disclosed.
3	$AUTH_x$	The authenticity of the disclosed account on platform x.
4	$AUTH_x$	The value of the authenticity variable reflects the size of the user's network on platform x.
5	HON_x	The honesty of the user, determined by comparing the information declared by the user and the information retrieved from the user's account on platform x.
6	$CONS$	The consistency of information between the different disclosed accounts.

Table 10.1: Correspondence of the variables in the social score formula to the hypotheses listed in Section 10.4.1.

The highest social score that can be achieved this way is 100 and the lowest is 0. Note that a low social score in this system does not mean that the user is guaranteed to have bad intentions or cannot be trusted. It could also mean that the user is not very active on social media. In any case, it means that the user did not reveal much about himself and that his social media accounts do not give us sufficient reason to trust him. In this situation, the user may obtain loans through traditional lending mechanisms such as using a collateral or his credit score.

10.5 Implementation

The information that is entered by the user and the calculated social score are saved on the blockchain. This information includes the user's name, email address, date of birth, loan amount, and the social score. The clear disadvantage of saving this information on a public blockchain is lack of the user's privacy. As we discussed in our prior conference paper [131] that described only the non-privacy-preserving version of our lending plaform, future work should address this problem of user privacy. In Section 10.6 of this work, we do indeed present an enhanced version of our platform that takes privacy considerations into account. For the current prototype, we consider that the data will be publicly accessible. In a future iteration of the prototype, we may store only the less sensitive information on the blockchain, for example, only the loan amount and the social score. The changes required in the code would be minimal since $setInfos$ is the only function that would need to be adapted.

10.5.1 Tools and Technologies

We decided to develop this implementation on an Ethereum test network. As stated in [143], the Ethereum test network ("testnet") simulates Ethereum, which gives the developers a chance to deploy and test Ethereum projects without getting any real assets involved. The testnet allows developers to easily obtain tokens and Ether for test purposes, which carry no financial value. This makes it possible to test a project with simulated tokens and Ether instead of using expensive valuable assets. A guide to using an Ethereum test network is provided by Hayes [143].

We connect to the Ethereum test network using Ganache (www.trufflesuite.com/ganache). Ganache is a tool that is used for setting up a local Ethereum blockchain. The smart contracts for this project are written in the programming language Solidity on the Ethereum IDE Remix (remix.ethereum.org). Solidity is a statically-typed programming language that allows developing smart contracts for Ethereum. Remix IDE is an open source web-based platform, which has a plugin architecture that promotes extensibility. Remix IDE provides several tools for all the steps required for smart contract development with the Solidity language.

To connect our smart contract with the Ganache blockchain, we use MetaMask (metamask.io). MetaMask is an Ethereum wallet. The wallet tries to simplify user experience for accessing decentralized applications (dApps) deployed on Ethereum. MetaMask can be installed as a browser extension. This allows it to provide a user-friendly interface for sending and receiving Ether.

For the frontend of the implementation, we use HTML and JavaScript. The frontend connects to the backend and therefore we test our blockchain by using the library Web3.js (web3js.readthedocs.io). Web3.js enables interaction with a local or remote Ethereum node using HTTP as well as some other protocols.

10.5.2 Smart Contract Implementation and Ethereum Gas Usage

According to [153], a smart contract is a program that is stored on a blockchain and runs when certain predetermined conditions are met. A smart contract is used to automate and enforce the execution of an agreement that has been made beforehand between the participants of the smart contract. The smart contract guarantees that the outcome of the execution will satisfy the agreed upon conditions. Moreover, the execution of the smart contract does not require the involvement of any intermediaries or trusted third parties.

On the Ehtereum platform, “gas” is a measure of the computational resources that are needed for the execution of a function of a smart contract. Executing a function requires payment of the gas fees determined for that function according to the computational resources that it would use.

Once the smart contract is deployed, one can use the functions within this smart contract while interacting with the blockchain. By calling a function in the smart contract that writes on the blockchain, for example the “setInfos” function, a block is mined and therefore gas must be payed. However, when one of the view-functions is called, no gas needs to be payed and no new block is mined on the blockchain. The “setInfos” functions is called only once when a person registers for the first time.

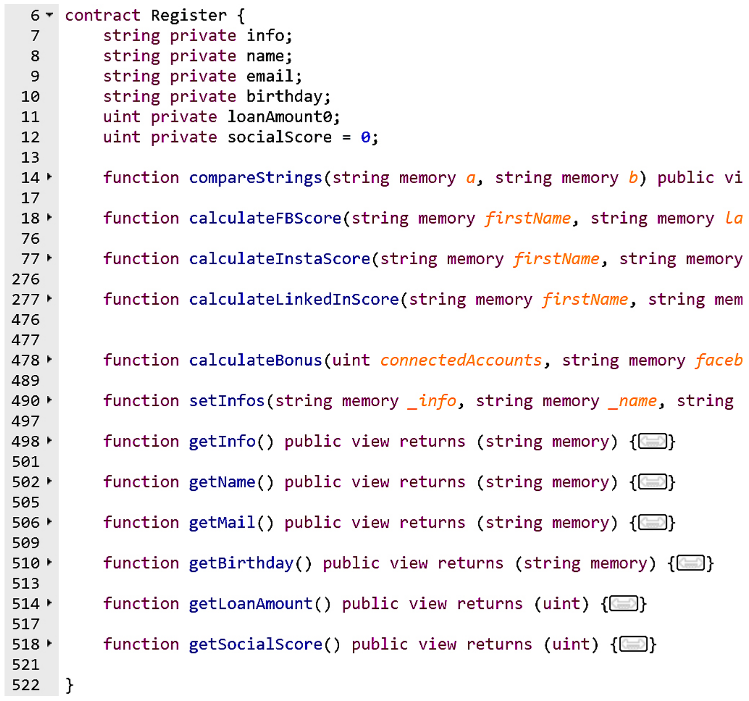
One main advantage of a smart contract is that the calculations performed are transparent and everybody can have trust in the calculated score. The smart contract calculates the score and then stores it in the blockchain by itself, without the intervention of any outside code. This way, everybody can trust in the process, including the user himself, and nobody can manipulate it. Figure 10.1 shows the listing of the functions in the smart contract implemented in Solidity.

Five of the functions in the smart contract write to the blockchain and therefore cost gas. The other functions are read- or view-only functions. The smart contract is deployed and written on the blockchain, which costs ether. The gas cost is 2141686. However, the smart contract only has to be deployed once. The functions within the smart contract, which cost ether because they write on the blockchain, are also called only once. This is done during the registration process. Not all of the functions are necessarily called. If the user does not connect his Facebook account, the function “calculateFBScore” is not called. The same is true for connecting the Instagram account and the LinkedIn account. If only one account is connected, there is no bonus added on top of the social score and as a consequence, the “calculateBonus” function is not called. The only gas costing function that is always called, is the “setInfos” function.

We also employ a helper function to compare strings. Further, we have one calculation function for each social media network and one function to calculate the bonus. None of the functions has a return value. They all work directly on the globally saved “socialScore” variable.

10.6 Privacy Considerations

As we have seen in Section 10.4.2, the initial version of our lending platform does not take the privacy of the borrower into account. A significant amount of personal information needs to be disclosed by the borrower in order for the lender to compute his social score. In this section, we discuss privacy considerations for our lending platform. We begin by looking at the private details that are divulged on the initial version of the platform. We then state our objectives for an enhanced privacy-preserving version of the platform. After that, we describe some cryptographic building blocks that we use. We then introduce our proposed measures based on the cryptographic building blocks to ensure the privacy of the borrower on the platform. We end this section with an overview of the security of our proposed privacy-preserving measures.



```
contract Register {
    string private info;
    string private name;
    string private email;
    string private birthday;
    uint private loanAmount0;
    uint private socialScore = 0;

    function compareStrings(string memory a, string memory b) public vi

    function calculateFBScore(string memory firstName, string memory la

    function calculateInstaScore(string memory firstName, string memory

    function calculateLinkedInScore(string memory firstName, string mem


    function calculateBonus(uint connectedAccounts, string memory faceb

    function setInfos(string memory _info, string memory _name, string

    function getInfo() public view returns (string memory) {⋯}

    function getName() public view returns (string memory) {⋯}

    function getMail() public view returns (string memory) {⋯}

    function getBirthday() public view returns (string memory) {⋯}

    function getLoanAmount() public view returns (uint) {⋯}

    function getSocialScore() public view returns (uint) {⋯}

}
```

Figure 10.1: Overview of the smart contract functions used in our implementation of the proposed system.

10.6.1 Analysis of the Disclosure of Private Information

We analyze what private information is revealed about the borrower for the computation of the various variables by the lender.

- $DISC_x$ is computed as a function of $OPEN_x$ and $AUTH_x$. Let's first look at the information revealed for $OPEN_x$. In order for the lender to equate $OPEN_x = 1$, the borrower gives access to his account on the social network x. This reveals the identity of the account as well as the content of the account to the lender.
- The $AUTH_x$ variable is computed as a function of the number of friends or followers of the borrower on the social network account x. Therefore, the borrower is expected to divulge the precise number of his friends or followers to the lender.
- SC_x is computed as a function of $DISC_x$ and HON_x. In order for the lender to compute HON_x, the borrower must provide personal information such as his name, email address, and date of birth on the lending platform. The lender then also accesses the corresponding information on the user's social network account x for the purpose of matching the two sets of information.
- The variable *bonus* is computed as a function of the variables n and $CONS$, where n is the number of social network accounts disclosed. In order to compute $CONS$, the lender accesses the content of n social network accounts of the borrower for checking the consistency of information among those accounts.
- SC is a function of the variables $SC_1 \ldots SC_n$, n, and $bonus$, which have all been discussed above.

10.6.2 Privacy Objectives

We state the objectives of our proposal for the preservation of the privacy of the borrower.

- **Non-disclosure of social network accounts.** The lending platform should not require the borrower to disclose his accounts. This includes his social network accounts as well as his email accounts. Yet, the borrower should be able to communicate and transact with the lender. Moreover, the lender should be able to compute the social capital score based on the information contained in the social network accounts of the borrower. This information is required by the lender to compute $DISC_x$ and $OPEN_x$. However, he will need to make the computations without learning the information. The borrower should reveal neither the identity of his social network accounts nor their content. Non-disclosure of social network accounts also implies non-disclosure of the biographical identity of the borrower. Otherwise, a simple search on the borrower's name could lead to the discovery of his social network accounts.

- **Non-disclosure of the number of friends.** The borrower will not be required to divulge the precise number of his friends or followers, which is required for the computation of $AUTH_x$.
- **Non-disclosure of personal information.** The borrower will not disclose personal information on the public blockchain or to the lender. This information, which is required for the computation of HON_x and subsequently SC_x, includes the borrower's name, email address, date of birth, and potentially other personal information. Moreover, as mentioned before, the lender will not have access to the content of the borrower's social network accounts. The lender will also have to compute the variables $bonus$, $CONS$, and SC, without access to the personal information.
- **Disclosure of a subset of friends.** We will tolerate the disclosure of a subset of the friends of the borrower for a given social network account x. We will assume that the subset is indistinguishable enough to not allow the discovery of the identity of the borrower.

10.6.3 Building Blocks

We describe the building blocks that we use for the preservation of the privacy of the borrower.

- **Homomorphic Cryptosystem.** As stated in [139], let $E_s(.)$ denote the encryption function with the public key PK_s of agent s in an asymmetric cryptosystem C. The cryptosystem C is said to be additive homomorphic if we can compute $E_s(x+y)$, given only $E_s(x)$, $E_s(y)$, and PK_s. For detailed information about homomorphic cryptosystems and their applications, we refer the reader to the thesis [243] of Doerte K. Rappe on this topic.
- **Zero-Knowledge Proof of Set Membership.** As stated in [139], let $F = \{m_1, \ldots, m_p\}$ be a public set of p messages, and $E(m_i)$ be an encryption of m_i with a prover's public key, where m_i is secret. A zero-knowledge proof (ZKP) of set membership allows the prover to convince a verifier that $E(m_i)$ encrypts a message in F.
- **Zero-Knowledge Proof of Plaintext Equality.** As stated in [139], let $E_u(m)$ and $E_v(m)$ be the encryptions of a message m with the public key of agents u and v respectively. A zero-knowledge proof of plaintext equality allows a prover to convince a verifier that $E_u(m)$ and $E_v(m)$ encrypt the same message.
- **Cryptographic Hash Function.** As described in [12] and [13], a cryptographic hash function takes an input of variable size and produces an output of fixed length. For example, for a 256-bit hash function, the output would always be 256 bits in size, if the input is 1 bit, 100 bits, or even 1 gigabits. A cryptographic hash function is efficient to compute. However, for $H(x) = h$, it is computationally infeasible to find x, where $H(x)$ is the function that takes x as input and h is the resulting hash of x. A hash function may be termed as a hiding function or a one-way function due to this property. Additionally, a hash function H is considered to be collision-resistant if it is infeasible to find a pair (x, y), such that $x \neq y$, yet $H(x) = H(y)$. This property allows $H(x) = h$ to be used as a message digest for x. Examples of cryptographic hash functions include SHA-2, SHA-3, and RIPEMD-160.

10.6.4 Our Proposed Privacy-Preserving Measures

In this section, we discuss our proposed measures that help preserve the privacy of the borrower on our lending platform.

Table 10.2 summarizes the correspondence between the privacy objectives stated in Section 10.6.2 and the the privacy-preserving measures that are taken to achieve them.

Privacy Objective	**Privacy-Preserving Measures**
Non-disclosure of social network accounts	The borrower does not disclose his biographical or his social network identity at any step during the loan request process. The borrower interacts with the lender only with a new, unique, pseudonymous, and unlinked identity that is created by the borrower on the lending platform.
Non-disclosure of the number of friends	The borrower and the certifiers publish the value of the exact number of friends only in encrypted form. The lender is only able to learn a general set that the value belongs to.
Non-disclosure of personal information	The borrower never reveals any personal information such as email addresses, date of birth, etc. to the lender in cleartext form. This information is published by the borrower and the certifiers only in encrypted or hashed form.
Disclosure of a subset of friends	The borrower appoints a subset of his friends as certifiers and discloses this information to the lender. However, this disclosure does not link the borrower to his social network account as long as the assumption that the subset is indistinguishable enough holds true.

Table 10.2: Correspondence between the privacy objectives stated in Section 10.6.2 and the the privacy-preserving measures that are taken to achieve them.

- **Borrower's pseudonymous identity.** A pseudonym is a fictitious identity of a user that hides his or her true identity. The borrower creates a new unique pseudonymous identity or address on the platform specifically for each new loan request. This identity is considered to have no link to his previous transactions as well as his social network accounts. Creating this identity may be as simple as generating a new public-private key pair. The borrower communicates with the lender using this new identity. Let's denote this pseudonymous identity of the user as b.

  Let's outline how generating a public-private key pair may be used to create a new unique pseudonymous identity for the borrower. This method is described in further detail in the book on Bitcoin and Cryptocurrency Technologies by Narayanan et al. [222]. The method uses a digital signature scheme that has a *generateKeys* operation. This operation generates a public key pk and a corresponding secret key sk. The public key pk is then considered as the pseudonymous identity of its owner. Any message that originates from the owner of the public key pk needs to be digitally signed by the corresponding secret key sk. All users can verify the authenticity of the message as originating from the owner, since only the true owner knows the corresponding secret key sk and thus only they can emit a correctly signed message. Due to the fact that in practice the public key pk would be a large random and unique number, nobody would be able to associate it with the true identity of the owner. Yet only the owner can use this identity since no other user knows the corresponding secret key.

- **Certifiers.** A borrower b appoints some of his friends from his account on the social network account x as *certifiers*. These are trusted friends who already have access to the borrower's social network content due to their friend status. The certifiers will publish the information gleaned from the borrower's social network account in homomorphic encrypted form. The lender or another verifier will be able to match the information published by the certifiers to the information published by the borrower in order to verify the latter's veracity. In case of inconsistency between the information published by the certifiers, the verifier may choose to believe the information that is certified by a certain majority. The threshold of the majority can be defined by the lender himself, for example, 50%, 75%, or even 100%.

  As stated earlier, we tolerate the disclosure of the identities of the certifiers to the lender. However, the lender does not gain any access to the social network content of the certifiers.

  Please note that the set of certifiers could alternatively be replaced by a single Trusted Third Party (TTP) certifier. This TTP could be the social network operator itself, for example, Facebook, or it could be a certification service provider that has been given access to the social network account by the borrower. However, this alternative solution would require the existence of such TTPs. In our proposed measures, we do not assume the existence of this third party infrastructure. Moreover, this alternative solution moves the platform towards centralization of trust, which may not be desirable.

- **Verifying the certifiers.** We use a challenge-response authentication protocol to verify the certifiers. In challenge-response authentication protocols, the verifying entity, let's say Alice, presents a challenge to the entity to be verified, let's say Bob. The challenge must be answered by Bob with a correct response that will be checked and validated by Alice. If the response is correct, that is, it satisfies the challenge that was presented, Alice is able to authenticate Bob. The absence of a correct response implies that Bob fails the authentication. The reader may see [10] for further information on challenge-response authentication mechanisms and protocols.

  The lender challenges the certifiers to demonstrate that they are valid account holders on the social network x. The lender sends a different challenge to every certifier on the certifier's presumed account on the social network x. Each challenge comprises of a nonce (number used only once) and the identity of the borrower. A certifier must reply with the unique nonce and a statement such as *"I am indeed a certifier for the borrower b"* from his presumed social network account. The ownership is verified if the identity of the replying account, the nonce, and the statement are correct.

- **How to link the borrower's pseudonymous identity with his social network account?** Let's denote the identity of the borrower b on the social network x as b_x. The borrower publishes the identity b_x in hashed form and links it with his borrower identity b. The information may be published as the tuple $< \ borrower\_identity, H(social\_network\_account) \ >$, where $H()$ is the cryptographic hash function. For example, $< \ b, H(b_x) \ >$. The borrower then sends the tuple to his certifiers in clear non-hashed form. For example, $< \ b, b_x \ >$. The certifiers already know the social network account of the borrower due to their friend status. The certifiers hash the social network account identity on their own and publish the tuple. The hashed data hides the social network account identity of the borrower. However, the hashed data allows comparison of the information published by the borrower and the information published by the certifiers.

  The lender can now verify the equivalence of the information published by the borrower and the certifiers. For example, let's say that a certifier c published $< \ b, H(b'_x) \ >$, then the lender can check whether $H(b_x) = H(b'_x)$. Please note that the lender does not learn the social network account identity of the borrower. However, the lender gains confidence through the certifiers that the borrower indeed owns an account on the social network x. Moreover, the borrower's anonymous unique identity b for the loan transaction is permanently linked to his social network account b_x while keeping it anonymous as well.

- **How to disclose the number of friends in a privacy-preserving manner (in order to enable the lender to compute $AUTH_x$ and $DISC_x$)?** The borrower b publishes the number of his friends f in homomorphic encrypted form, that is, $E_b(f)$. The borrower then asks his certifiers to publish the number of his friends in encrypted form as well. The certifiers can retrieve this information independently since they have access to the borrower's social network content. Let's say that a certifier c observes that the number of friends is f'. The certifier then publishes $E_c(f')$.

  After publication of the number of friends in encrypted form by the certifiers, the borrower b and each certifier c jointly generate and publish a plaintext equality zero-knowledge proof to demonstrate the equality of f and f'. The lender will verify the plaintext equality ZKP to gain confidence that both borrower b and certifier c published the same number of friends, that is, $f = f'$, even though the lender does not learn the value.

  Moreover, the borrower b publishes a set membership zero-knowledge proof to demonstrate that the number of friends f belongs to a certain set F_i that is known publicly. These sets can be pre-defined, for example, $F_0 = \{0, 1, \ldots, 99\}$, $F_1 = \{100, 101, \ldots, 999\}$, $F_2 = \{1000, 1001, \ldots, 9999\}$, $F_3 = \{10000, 10001, \ldots, 99999\}$, $F_4 = \{100000, 100001, \ldots, 999999\}$, $F_5 = \{1000000, \ldots\}$, etc. The lender will verify the set membership ZKP to learn the range. Please note that the lender will not learn the precise number of friends. The disclosure of the range places the borrower in a k-anonymous set of other users whose number of friends belong to the same range. In this example, the value of $AUTH_x$ could be calculated as given in Equation 10.7, where the function $smzkp(E_b(f), F_0)$ returns true if $f \in F_0$.

$$AUTH_x = \begin{cases} 0 & \text{if } smzkp(E_b(f), F_0) = true \\ 0.2 & \text{if } smzkp(E_b(f), F_1) = true \\ 0.4 & \text{if } smzkp(E_b(f), F_2) = true \\ 0.6 & \text{if } smzkp(E_b(f), F_3) = true \\ 0.8 & \text{if } smzkp(E_b(f), F_4) = true \\ 1 & \text{if } smzkp(E_b(f), F_5) = true \end{cases} \tag{10.7}$$

The above steps enable the lender to compute $AUTH_x$ and $DISC_x$. The lender does not learn the value of the number of friends because it is published in encrypted form by both the borrower and the certifier. However, the lender is still able to verify the validity of the value and learn a general set that the value belongs to. This is possible due to the properties of the homomorphic cryptosystem that is used and the zero-knowledge proofs published by the borrower and the certifier in conjunction with the encrypted values. The homomorphic cryptosystem enables computation on the encrypted values without the need for their decryption.

- **How to publish name, email address, date of birth, etc. in a privacy-preserving manner (in order to enable the lender to compute HON_x)?** The borrower b publishes information such as his name, email address, and date of birth in hashed form along with tags that describe the hidden data. For example, the information may be published as tuples in the format $< data\_tag, H(data\_value) >$, where $H()$ is the cryptographic hash function. Even further information, such as location, interests, groups, etc., may also be published in this manner. Some examples of the published information include: $< name, H(Alice) >$, $< email, H(alice@alice.mail) >$, $< date\_of\_birth, H(01/01/2001) >$, $< location, H(Lyon) >$, $< interest\_01, H(Tennis) >$, $< interest\_02, H(Golf) >$. An agreed upon nonce may be concatenated to a value to be hashed by the borrower and a certifier to further protect the confidentiality of the value.

  The borrower then sends the published information in clear non-hashed form to his certifiers. The certifiers are assumed to already have access to this information due to their friend status. The certifiers then look up the information on their own on the borrower's social network account; compute the hashes of the information that they find; and then publish the information so that it is visible to the lender in hashed form. The hashed data hides the data values. However, the hashed data allows comparison of the information published by the borrower and the information published by the certifiers.

  The lender can now verify their equivalence. The lender does not need to learn the data values. The lender is interested in whether the borrower is truthful about the information of his social network account. He can achieve this by verifying the equivalence. The lender is able to compute HON_x for each social network account b_x for which the borrower and his certifiers publish the information.

10.6.5 Security Overview

We first discuss whether the lender is able to compute a correct social score for the borrower on the privacy-preserving platform, despite not being able to learn personal and social network information about the borrower.

The lender is still able to correctly assign a value between 0 and 1 to $AUTH_x$. Instead of using the precise number of friends, which is no longer known, the lender uses the range of friends. For example, $AUTH_x$ could be 0 for the range F_0, 0.1 for the range F_1, and so on. $OPEN_x$ can also still be correctly assigned the value 0 or 1 by the lender. This is because the lender can determine through the certifier verification and the identity linking processes whether the borrower is providing information in encrypted form from his account on the social network x. $DISC_x$ can be derived from $AUTH_x$ and $OPEN_x$ computed above.

Regarding HON_x, as we discussed in the previous section, the lender can correctly assign its value depending on the extent to which the information published by the borrower and his certifiers matches. This can be done without the need to learn the information itself. The other variables such as SC_x, *bonus*, n, $CONS$, etc. can be either derived from the variables discussed above or calculated in a similar manner.

An underlying assumption for the correctness of the social score is clearly the honesty of the certifiers. The score may be manipulated if the borrower and a majority of the certifiers collude and cheat. However, as we discussed, the lender himself can set the threshold of the majority. For example, the lender could require that at least 75% of the certifiers provide consistent information that validates the information provided by the borrower. Moreover, the lender may also set the threshold for the minimum number of participating certifiers. If these criteria are not met, the lender may consider a borrower as untrustworthy and reject the loan request of the borrower.

We now discuss how effectively the privacy of the borrower is protected in the enhanced privacy-preserving version of our lending platform.

The borrower does not disclose his biographical or his social network identity at any step during the loan request process. The borrower interacts with the lender only with a new, unique, pseudonymous, and unlinked identity that is created by the borrower on the lending platform. The borrower appoints a subset of his friends as certifiers and discloses this information to the lender. However, this disclosure does not link the borrower to his social network account as long as the assumption that the subset is indistinguishable enough holds true.

In the privacy-preserving version of our lending platform, the borrower also never reveals any personal information such as the exact number of friends, email address, date of birth, etc. to the lender in cleartext form. This information is published by the borrower and the certifiers only in encrypted or hashed form. The confidentiality of the information is maintained as long as the security of the cryptosystem and the cryptographic hash function is not breached.

Another obvious assumption for the privacy of the borrower is that none of the certifiers will collude with the lender to reveal his identity and his private information. However, the probability of a certifier acting maliciously is considered low by the borrower since the certifiers are trusted friends who already have access to the borrower's information. Moreover, in case of ambiguity regarding the trustworthiness of a certain certifier, the borrower should be able to ascertain the risk of breach of privacy before appointing them as a certifier.

10.7 Evaluation

In this section, we evaluate the execution of the social score function on the "Social circles: Facebook" real user dataset. The objective is to determine whether the analysis of real social datasets can help the operator set the parameters of a platform in production.

10.7.1 Setup of the Test Environment

To evaluate the function itself, we use real user data from the Stanford Network Analysis Project (SNAP). The dataset is called "Social circles: Facebook" [19]. To interpret the dataset and to make calculations based on it, we work with the Anaconda Prompt (docs.anaconda.com) and the python environment Jupyter Notebook (jupyter-notebook.readthedocs.io). Within jupyter notebook, we imported the libraries pandas (pandas.pydata.org) and networkx (networkx.org) as well as matplotlib (matplotlib.org).

10.7.2 Results and Observations

We used the 107.edges file from the SNAP dataset [19]. It contains 53498 edges (signifying friend relationships) and the corresponding nodes. The dataset is anonymized. As discussed earlier, the number of friends influences the authenticity of a person. Evaluating the data shows that more than a third of all users have more than 50 friends and therefore have a chance to get the best social score if we set the threshold to this value. On the other hand, there are also almost 15% users who do not have more than ten friends and consequently get the worst result in this category. The results are shown in figure 10.2. In our evaluation, we only differentiate between three steps concerning the number of friends: 10, 30 and 50. Since more than a third of the users have enough Facebook friends to get the best result possible in the "amount of friends" category, we evaluate if a higher limit would be more suitable. For testing reasons we will adapt these limits. We will start with 200 and then go down to 150, 100 and lastly to 50. The amount of users tested in this experiment is 1034. When we set the threshold to 200, which means that the user needs more than 200 Facebook friends to get the best result in this category, only 12 of the 1034 users qualify for the best result. Subsequently, we obtain the result of 50 users qualified for a threshold of 150, 163 users for a threshold of 100, and finally 389 users for a threshold of 50. The results of this last experiment are plotted in the figure 10.3. In this experiment, we see that by analyzing real datasets, we can set the thresholds for the platform in order to correspond to the desired rate of users who should qualify. The number of friends is only a small part of the final social score. Other factors include the number of accounts connected, number of pictures posted, account creation date, the bonus

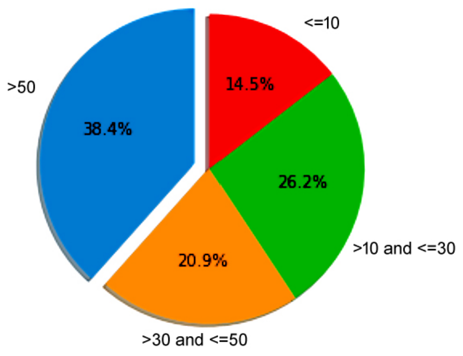


Figure 10.2: The biggest part has more than 50 Facebook friends and gets the best result in this category.

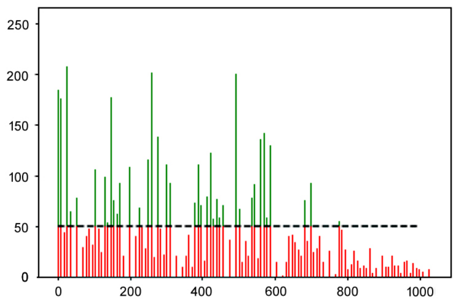


Figure 10.3: The x-axis shows the users and the y-axis shows how many Facebook friends each of these users has. 389 out of 1034 users exceed the threshold of 50.

for connecting 5 accounts, etc. These factors could not be considered in this experiment due to the absence of this information in the dataset.

10.8 Conclusion

In this work, we have presented a new approach to calculate the users' trustworthiness on peer-to-peer lending platforms. This approach is neither collateral nor credit score based. The formula that we use to calculate a user's social score relies on the social capital theory and consequently the information retrieved from the user's social network accounts. It considers how well connected a user is, since connections may help the user achieve professional and personal success. The social score is implemented in a smart contract running on the Ethereum blockchain. The whole lending process is automated and does not need any input from a middleman.

The presented approach offers a new method to verify users' trustworthiness regarding lending, where even users with a non-sufficient credit score and no valuable assets as collateral could get a chance on a loan. We quantified the amount of gas that is consumed by the deployment of the smart contract. Moreover, we also evaluated the execution of the function on a real social network dataset. This experiment demonstrated how we can use analysis of social network data to determine optimal thresholds for a platform in production.

Furthermore, we presented an enhanced version of our lending platform that takes privacy considerations into account. The proposed measures, based on cryptographic building blocks such as zero-knowledge proofs and cryptographic hash functions, enable the platform to preserve the borrower's privacy. We observe that the lender is still able to compute the borrower's social score despite being unable to learn any personal information or even the identity of the borrower.

Chapter 11

Analyzing Flow of Payments in a Business-To-Business Network to Detect Supplier Impersonation

Supplier Impersonation Fraud (SIF) is an increasingly significant issue for Business-to-Business (B2B) companies. The use of remote and quick digital transactions has made the task of identifying fraudsters more difficult. In this work, we propose a data-driven fraud detection system whose goal is to provide an accurate estimation of financial transaction legitimacy by using the knowledge contained in the network of transactions created by the interaction of a company with its suppliers. We consider the real dataset collected by SIS-ID for this work. We propose to use a graph-based approach to design an Anomaly Detection System (ADS) based on a Self-Organizing Map (SOM) allowing us to label a suspicious transaction as either legitimate or fraudulent based on its similarity with frequently occurring transactions for a given company. Experiments demonstrate that our approach shows high consistency with expert knowledge on a real-life dataset, while performing faster than the expert system.

We note that this work proposes a solution where data is analyzed in a centralized and trusted manner. However, this work is still relevant to the topic of privacy preservation and decentralization for a couple of reasons. Firstly, the main challenge lies in analyzing data that has been heavily anonymized in order to preserve the privacy of the participating companies. Secondly, the graph of transactions between companies is decentralized in nature. That is, transactions take place in a direct B2B manner without intermediaries. The assumption of a trusted central transaction handling entity (such as a clearinghouse) would essentially eliminate the problem. Reliance on such a central trusted entity is not assumed for the B2B transactions.

This chapter is an adapted version of the article: "GraphSIF: Analyzing Flow of Payments in a Business-to-Business Network to Detect Supplier Impersonation." R. Canillas, O. Hasan, L. Sarrat, and L. Brunie. *Applied Network Science (Springer).* 2020. Vol. 5, article 40. This work was carried out in the context of the Ph.D. of R. Canillas, co-supervised with L. Brunie.

Privacy-preserving fraud detection techniques were surveyed in the following paper (not included in this habiliation thesis): "Exploratory Study of Privacy Preserving Fraud Detection." R. Canillas, R. Talbi, S. Bouchenak, O. Hasan, L. Brunie, and L. Sarrat. *19th ACM/IFIP International Middleware Conference (Middleware 2018).* December 2018. Pp. 25-31.

11.1 Introduction

Fraud is a recurring issue in many domains such as credit card transactions, insurance, telecommunication, and finance. Supplier Impersonation Fraud (SIF) is a specific case of identity theft, targeting a company rather than an individual. This type of financial fraud is widespread, resulting in the loss of hundreds of thousands Euros in 2018, and ranked 1st most frequent fraud affecting French companies in the latest survey about cyber-criminality conducted in 2019 by Euler Hermes and DFCG [93]. Supplier impersonation consists of a fraudster impersonating a member of a company providing goods and services to another, in order to trigger a payment on an account controlled by the fraudster [23].

We can illustrate SIF with a toy example. Let C be a company that produces computers. In order to acquire the necessary components for the fabrication, C buys electronic chips from S.

Let F be a fraudster. A SIF takes place when F diverts a payment from C originally destined to S, usually by impersonating S in C's eyes. Such an impersonation can take several forms: the tampering of an invoice from S (similar to phishing attacks), impersonating a high-ranking employee of S and requesting the next payments to be paid on an account controlled by the fraudster, or even requesting the payment of imaginary goods and services on behalf of S.

As more and more companies are using digital tools to process transactions due to numerous advantages provided by digitalization, the risk of supplier impersonation has never been higher. However, the tools and systems required to detect SIF have not evolved at the same pace, and still mostly rely on expert-based input and monitoring. This approach might not be the most suitable to handle the increasing volume of digital transactions. Thus, the need for automated, data-driven SIF detection system arises.

In this work, we present GraphSIF, a SIF detection system that uses a B2B transactions dataset to construct a graph modeling the relationships between client and supplier companies in a B2B ecosystem, and describe how these relationships can be used to derive useful knowledge in order to assert the legitimacy of transactions.

We use the transaction network to create a time-evolving behavior sequence summing up the evolution of the graph through time. We then compare the new graph created by adding a suspicious transaction to the behavior sequence and investigate the potential discrepancy it introduces. If this discrepancy is low then the transaction is considered as legitimate, and if the discrepancy is high then the transaction is considered as likely fraudulent. In order to quantify this discrepancy, a Self-Organizing Map (SOM) is trained on the behavior sequence, and a clustering algorithm is used to quantify the similarity of the tested graph with the ones in the behavior sequence.

Finally, we analyze the results of GraphSIF using a set of transactions labeled by experts from the SiS-id company, in order to evaluate its performance, and investigate its potential shortcomings. We found that GraphSIF with the selected parameters shows high consistency with the expert system when focusing on low-legitimacy transactions.

11.2 Related Work

Due to the sensitivity of the data linked to supplier fraud detection for victim companies, we have not been able to find any publicly available research work directly related to SIF. However, we can find several systems designed for fraud detection that also use a network-based approach:

In [253] several network-based fraud detection use-cases are introduced, showing examples of successful use of graph theory to detect bank fraud, insurance fraud and e-commerce fraud. However, the authors focus on specific industrial examples without proposing a formalized evaluation of their solution.

Akoglu, Tong and Koutra [25] propose a survey on anomaly detection using graphs, notably in the domain of telecommunication fraud detection. An approach closely related to our own is found in [24] where an egonet (1-step neighborhood graph) is used to derive features describing a node. However, this approach is applied to a static graph that does not evolve through time, contrary to our approach. The analysis of dynamic graphs through the use of windows, as it is the case in our work, is akin to the ideas developed in [239] and [217] where the graphs are analyzed using a moving window and detecting anomalous connectivity variation [239] or edges p-value variation [217]. To the best of our knowledge there is no previous attempt at combining a window-based analysis as seen in [239] with the feature approach used in [25].

Van Vlasselaer et al. [280] proposes an approach somewhat similar to ours, using graphs to represent interactions between companies in order to detect social security frauds. However, their work focuses on the application of social network algorithms on large graphs of interconnected entities, whereas our work considers smaller neighborhood graphs, focused on the behavior of a single company. In addition, the system proposed in [280] bases itself on a propagation algorithm and Random Logistic Forests to perform their analysis, and does not make use of Self-Organizing Maps.

Furthermore, most of the aforementioned research uses supervised algorithms as they rely on the existence of known legitimate and fraudulent neighboring nodes to conduct their analysis. Our work proposes an unsupervised approach that does not take into account the legitimacy of neighboring nodes to propose a label, but instead uses the topology of the ego network.

Finally, as our approach makes use of significant patterns found in the 2-step egonet (the set of vertices found at most 2 edges away from the considered vertex) of a company, a parallel can be drawn with the discovery of network motifs, for example, as presented in [214]. However, the discovery of network motifs relies on the observation of patterns occurring with a statistically significant number of occurrences compared with random graphs showing similar topology. Our approach uses a different technique to create patterns from the egonet, by investigating the remaining connected sub-graphs when the ego node is removed. The techniques used in network motif analysis could however be used to discover the occurrences of payment motifs across for a number of targeted companies and thus assess if recurring payments behavior are shared in the payment ecosystem.

To the best of our knowledge, our work is the first to use a graph-based approach paired with Self-Organizing Maps in order to address the issue of SIF detection.

11.3 SiS-id Fraud Detection Platform

While SIF is a widespread fraud, it is mostly dealt with internally by companies victim of the fraud. There are three main reasons that motivate the lack of collaboration in SIF detection: firstly, being public about being victim of a fraud can cause a breach of trust and bad publicity for the company. Secondly, due to the competitive nature of the Business-to-Business ecosystem, information about the relationship between a client and its suppliers is a sensitive knowledge that could lead to economic attacks if divulged. Finally, having a successful in-house fraud detection system provides an edge for the company that owns it, and thus such a system would not be willingly shared with competitors. However, the cost of creating and maintaining complex fraud detection systems is sometimes prohibitive for a large number of companies.

In this context, the company SiS-id [1] proposes to focus on Supplier Impersonation Fraud detection and mitigation. The company, started in 2016, develops several SIF mitigation tools that other companies can use “as a service”. SiS-id emphasizes strongly on the privacy and security of the data shared by their clients that they use to develop detection techniques. Currently, SiS-id proposes two SIF mitigation systems: firstly a fraud detection system based on the relationship network created in the data shared by each of its clients, and secondly a secure repository for trusted bank accounts corresponding to verified suppliers. In this work, we propose a data-driven system that improves upon SiS-id’s current expert-based system to perform SIF detection.

The remainder of this section describes the dataset available to SiS-id to perform SIF detection, along with the system they implemented on the platform.

11.3.1 History Dataset

The SiS-id SIF detection system uses a set of historical transactions performed by SiS-id’s clients. In this section, we describe this dataset in detail.

The set of B2B transactions used by the SiS-id detection system is an aggregation of the payments performed by SiS-id’s client companies between July 2016 and July 2019. These transactions consist of a feature vector of 4 features : client identification number, supplier identification number, target account identification number, and date of the payment. For the sake of storage, all of the transactions involving the same client, supplier and destination account during a single month are aggregated, resulting in the creation of a fifth feature representing the number of similar transactions issued during the month. The time granularity of the transaction is thus a month. Table 11.1 shows an overview of the features.

The amount of payment is a feature found in most financial fraud detection systems (as seen in [58]). However, this data is very critical for companies (as described in 11.3), this feature was not shared with SiS-id by the companies that agreed to collaborate on the creation of the History dataset. Indeed, disclosing the amount of the transactions issued to their suppliers might divulge economic insights regarding their financial well-being, as well as provide a useful baseline for potential frauds. Thus, GraphSIF relies only on payments without their amounts.

In order to preserve the confidentiality of the data, a secure hash function is applied to the three distinct identifiers (client, supplier, account) so that no link can be established between the data in the history and real-life companies. While this mitigates the risks of damage in case of data leak, it also means that the same company will have a different identifier whether it has issued a transaction, or received a transaction.

At the time of writing, 950,929 transaction records are available in the History dataset. These transactions are issued by 6,063 unique companies. This number is more than the number of SiS-id’s client companies. This is explained by the fact that SiS-id’s client companies can represent a group of several branches such as a multinational group. In this case, each firm possesses its own identification number, but only a global entity will be SiS-id’s client. 215,056 unique

[1]https://www.sis-id.com/

Feature	Type	Description
Client	Nominal (ID)	Identification number of the client issuing the transaction.
Supplier	Nominal (ID)	Identification number of the supplier receiving the transaction.
Account	Nominal (ID)	Identification number of the bank account to which the money is transferred.
Date	Timestamp (Month)	Timestamp indicating the date when the transaction took place.

Table 11.1: Structure of the History data record. The History dataset is composed of 950,929 records collected from 6,063 companies over two years.

supplier companies are also found in the dataset, along with 262,157 unique bank accounts to which a payment was transferred. The fact that more bank accounts than supplier companies exist indicates that some suppliers use more than one bank account to be paid.

This dataset represents all the transactions performed by all of SiS-id clients for two years. However, there is no available information about the legitimacy of these data, and thus no knowledge of which transactions are fraudulent and which are legitimate.

Due to the economic sensitivity of the data for the companies (as discussed in 11.3 , no sample can be made publicly available at the moment. However, a request for data samples can be submitted directly to SiS-id through their website [2], or by contacting L. Sarrat, co-author of this work.

11.3.2 Audit Dataset

A second set of transactions is available thanks to SiS-id. It consists of the list of transactions that were analyzed using the expert system in the past 2 years (July 2017 - July 2019). The dataset, called the "Audit" dataset, is composed of 218,325 suspicious transactions submitted by 317 unique client companies. The transactions underwent the fraud detection process devised by SiS-id, and were labeled with a legitimacy label : "high" indicating that the transaction has a high chance of being legitimate, "medium" meaning that the rule engine lacks the necessary information to assert if the transaction is legitimate or not, and "low" meaning that the transaction's legitimacy is low, which can be the case if the transaction is either invalid or fraudulent. This dataset possesses the following properties: 1) It contains real life queries for transaction validation made by companies. 2) Each of the transaction of this dataset contains a target variable corresponding to the classification done by the rule engine. 3) The transactions were cross-validated by the clients for consistency.

The legitimacy label found in the dataset might tempt us to use this dataset to perform supervised learning. However, this approach has a major drawback: by using data analysis on a dataset that is the result of the rule-engine system (described in the next section), we will only manage to "rediscover" the rules. However, this dataset might be used as a validation set for other fraud detection systems, in order to compare their performance with SiS-id's expert system and investigate the potential convergence of their results.

11.3.3 SiS-id Expert System

The fraud detection system SiS-id currently runs on its platform[3] is an expert fraud detection system, where a potentially fraudulent transaction is examined in order to assert its legitimacy, using knowledge available on the platform. This kind of systems inherits directly from the tradition of fraud detection teams ([172]), and aims to formalize their knowledge in order to efficiently process a large number of transactions. The fraud detection system designed by SiS-id consists of two separate steps: a feature engineering step where the features from the tested transaction are used to gather additional information, and then the gathered data is matched against a set of expert-defined rules in order to assert the transaction's legitimacy. For confidentiality reasons, it is not possible to discuss the inner workings of the system in detail. However, it has been designed by a team of SIF detection experts and thus provides a valid approximation of the expert knowledge.

[2]https://www.sis-id.com/#contact
[3]https://my.sis-id.com

11.4 GraphSIF Overview

GraphSIF is a SIF detection system based on anomaly detection that uses the relationships created between the companies interacting in a B2B environment in order to determine the legitimacy of a transaction, given the company that issued the payment.

This system is composed of four phases:

1. A pre-processing phase, where historical transactions are sorted by the companies that issued them, and grouped in time windows in order to describe a time-evolving sequence composed of several fixed-length windows of transactions, describing the behavior of a specific client. This phase is described in Section 11.5.

2. A feature engineering phase where each of the windows of transactions is transformed into a graph. This graph sums up the interactions that occurs between the client and its suppliers during the specified windows. Section 11.6 describes this phase.

3. An anomaly detection phase where a specific transaction (the "suspicious" transaction) is added to the most recent graph, creating a "test graph". This graph's similarity with the ones occurring in the historical sequence is computed. Section 11.7 provides more details about this phase.

4. A label attribution phase where the similarity of the test graph given a set of different sizes of the windows of transactions are aggregated. A legitimacy label derived from the aggregation score is computed. Details about this phase are found in Section 11.8.

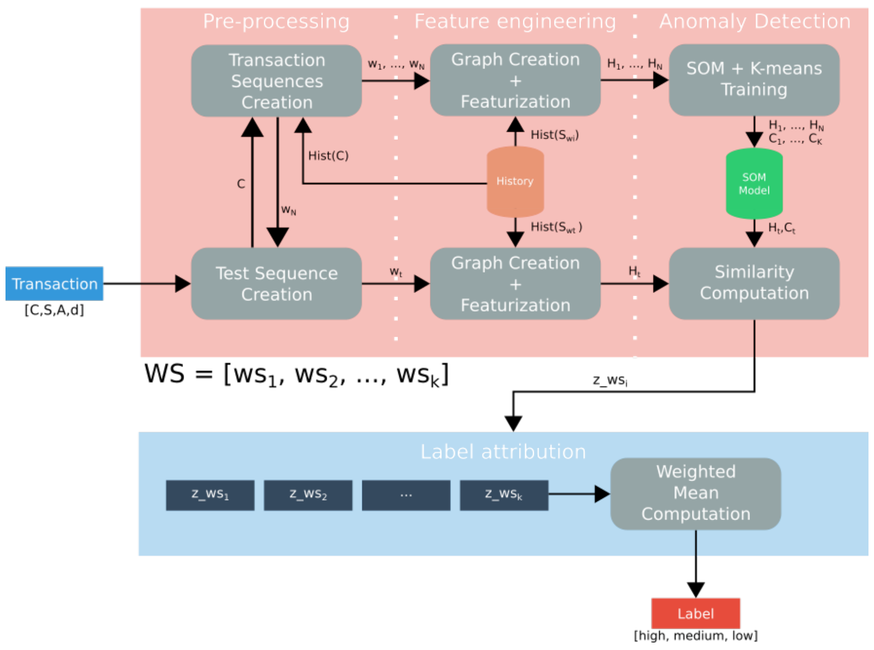


Figure 11.1: GraphSIF overview.

Figure 11.1 shows an overview of the process. A transaction t involving the client C, the supplier S and the account A at a date d is given as input to GraphSIF. For a set of window sizes $ws_1, ws_2, ..., ws_k$, the following process is repeated:

First, the identifier of C is used to gather all transactions involving C (Hist(C)) from the History dataset (that contains the list of all the transactions occurring between all the companies of the B2B ecosystem). Then, this list of transactions is ordered by date of occurrence and split in N fixed-size windows of size ws_i where $i \in 1, ..., k$. The sequence of transactions (dubbed "behavior sequence") is then sent to the next phase. At the same time, the oldest transaction of the most recent window of the sequence (w_N) is removed and t is added as its most recent transaction, creating the "test window" w_t. This step allows us to isolate the transaction relevant to the specific client and suppliers potentially victims of the supplier impersonation fraud.

In the next step, the transactions found in each window $w_1, ..., w_N$ and w_t are used to create a graph that represents the relationships between C and all of its suppliers. Each window is complemented by a set of transactions from the History dataset Hist(S_{wi}), where S_{wi} is the set of supplier involved with C during w_i found in the History Dataset. Similarly, Hist(S_{wt}) is used to create the test graph corresponding to w_t. The graphs are then transformed into a histogram that uses relationships between the accounts paid by C and used by its supplier as characterizing features of the graphs. Each of the graphs is converted into its corresponding histogram H_i, thus creating the sequence $H_1, ..., H_N$ and H_t the histogram corresponding to the test graph. This step allows us to transform a sequence of transaction into a feature vector representing the relationship between a client and its suppliers, taking into account the historical behavior of the client.

These histograms are then used to assert the similarity of H_t with the histograms of the behavior sequence $H_1, ..., H_N$. First, the histograms are clustered using the clustering algorithm K-Means ([155]), and then are used to train a Self-Organizing Map (SOM) ([61]). The k centroids $C_1, ..., C_K$ are also located on the SOM. Then, H_t is assigned a cluster using the previously trained K-Means algorithm, and its similarity with the other members of the cluster is computed using the z-score metric. This anomaly detection step allows the system to distinguish usual relationship and unusual ones, that are more likely to be fraud attempts.

Once all the z-scores corresponding to the set of window sizes $ws_1, ..., ws_k$ is computed, a threshold function is applied and a weighted mean is used to aggregate the results into a label indicating the legitimacy of the transaction. This step is needed to consider the different granularity of each windows, and to produce a label that synthesizes all the knowledge provided by the previous analysis.

In the remainder of the chapter, we first describe the different algorithms used at each phase of the system, and discuss the underlying motives behind their design. We then provide an experimental evaluation of the system using the labeled transactions found in the Audit dataset that contains the results of the expert system designed by SiS-id.

11.5 Transactions Pre-Processing

This section details the first phase of the system, where the transactions from the History dataset are pre-processed in order to create local behavior profiles. The goal of this pre-processing is to partition the transactions emitted by a client C in order to detect repeatability in its payments.

11.5.1 Company Local Profile

In this phase of the fraud detection system, all the transactions involving the client C involved in the tested transaction $t = [C, A, S, d]$ are gathered from the History dataset. The History dataset contains all the N_H transactions made by all clients in the studied B2B environment. By isolating only the transactions issued by C, we create a local profile of C. As shown by [57], the use of local profile allows to detect anomalous behavior that would have been deemed legitimate when using a global profile. This local profile is dubbed Hist(C).

11.5.2 Behavior Sequence

As companies evolve and thus interact differently with suppliers or clients, the transactions they issue or receive change as time passes. In order to quantify this evolution, we first order all the transactions in Hist(C) temporally. This gives us an overview of C's interaction with its supplier through time. Then, in order to characterize this behavior, we partition Hist(C) into sets of transactions of size ws, that we call "windows". Using fixed-length windows in order to describe the behavior of a system is a well-known technique, and has been successfully used in fraud detection systems such as ([241]) and ([217]).

11.5.3 Window Creation

In order to create the windows, two kinds of partitioning are possible: by transaction date (from July 1st to August 1st for example), or by transaction rank (10th transaction to 5th transaction, 5th transaction to 1st transaction, etc.). A major issue with partitioning by date is that there is no guarantee that the transactions will be homogeneously divided into the different windows. In the most extreme case, all transactions might occur in a single window, and all the other windows are rendered useless for the system. Thus, creating windows by transaction order allows us to ensure that an equal number of transactions will be found in each window.

The number of windows N created from Hist(C) is inversely proportional with the size ws of the windows: $N = \frac{N_H}{ws}$. In the case when N_H is not a multiple of ws, the $N_H\%ws$ oldest transactions in $Hist(C)$ are discarded, where % is the modulo operation. Indeed, the oldest transactions are the least likely to inform us of a fraud in the present.

The size ws of the windows has a major impact on the system. A small window size creates more data points for the system to analyze, at the cost of a decreased variability in the possible payment behavior, and thus a less detailed view of C's behavior. Inversely, a larger window allows for more detailed view of the system, but less input will be provided to the system. Additionally, adding more transactions might add too much noise to the window and thus obfuscate meaningful patterns. Therefore, a careful trade-off has to be found for ws. We propose a way to solve this issue in Section 11.8.

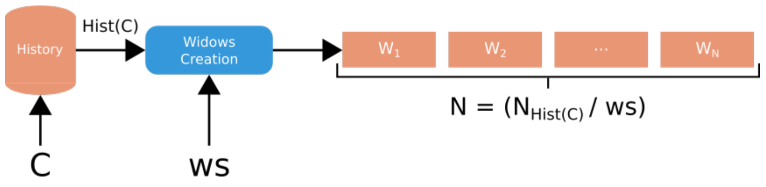


Figure 11.2: Behavior sequence creation The behavior sequence is created by partitioning the list of transactions issued bt C in the History dataset into sequential windows of size ws.

11.5.4 Test Window

In order to assert the legitimacy of a singular transaction, we use the previously partitioned sequence as a basis. The key hypothesis is that a legitimate transaction will not significantly disturb the payment behavior of C, while a fraudulent transaction will be different from the previously recorded behavior. The transaction to be tested is added to the most recent windows of the sequence, whose oldest transaction has been removed, thus simulating the occurrence of the new transaction as the next step in the sequence. Indeed, if only a single transaction was tested against the partitioned sequence, an anomaly would always be found due to the discrepancy in the number of transactions.

In the next section we detail how the relational data found in partitioned sequence (called "behavior sequence") and the test window is extracted and used to assert the legitimacy of the tested transaction.

11.6 Graph-based Feature Engineering

In this phase, each of the windows created in the pre-processing phase is transformed into a graph. This graph, called "transaction graph", allows the representation of the relationships between companies as a mathematical structure. The graph is composed of companies and accounts represented as vertices (also called "nodes"), while the flow of money between companies creates the edges of the graph. However, in order to use the graphs derived from the behavior sequence as a basis for the anomaly detection system, they need to be converted into a structured data form (commonly referred to as "feature vector") in order to be used. This type of transformation is known as "graph embedding" ([123]). We propose a tailored graph embedding approach in order to express a transaction graph as a feature vector called "graph histogram". This approach exploits several properties of the transaction graph in order to construct the embedding. The graph histograms corresponding to the behavior sequence are then used to train our anomaly detection system, while the graph histogram corresponding to the test windows is investigated.

The role of graph theory in GraphSIF is to transform the four attributes found in the History dataset into a set of graph-theoretic features allowing to perform fraud detection by taking into account the underlying relationships between companies and bank accounts found in the dataset. In the original form of the History dataset, the data points cannot be used directly to construct meaningful models, as their feature are categorical variables. The use of graph theory allows GraphSIF to express the links between each client and their suppliers through the bank account that they share, and to transform the list of categorical variables into a set of embedded graphs. Without this step the model could not be computed and thus fraud detection could only be performed on the reduced subset of original features from the History dataset.

11.6.1 Transaction Graph Creation

In this subsection, we describe the process of creating a transaction graph from a set T of transactions (as a reminder, a transaction t has the following structure: $t = [C_t, A_t, S_t, d_t]$ where C_t is the client issuing the transaction, A_t is the account receiving money from C_t, S_t is the supplier receiving the transaction, and d_t the date when the transaction takes place). A graph $G =< \boldsymbol{V}, \boldsymbol{E} >$ is composed of two sets: a set of vertices $\boldsymbol{V}$ that represents the entity of the targeted system, and a set $\boldsymbol{E}$ of edges that represents the relationship of the entities, and $e \in \boldsymbol{E} =< n_1, n_2 >$ with $n_1, n_2 \in \boldsymbol{V}^{\mathbf{2}}$.

Algorithm 11.3 shows the process that creates a transaction graph $G_{C,T}$ from a set of transaction $\boldsymbol{T}$. An example of output of Algorithm 11.3 is given in Figure 11.4. The algorithm takes as input a client C and parses $\boldsymbol{T}$ in order to map all of the accounts and suppliers involved in a transaction with C as vertices. For each transaction, two edges are created: one that links C and the account involved in the transaction, and one that links the account with the supplier receiving payment for the transaction. If a vertex representing an account or a supplier is already in the vertices set, it is not duplicated. Similarly, since edges already in the edge set are not duplicated, an occurrence metric is updated in both cases in order to prevent the loss of information.

If the algorithm stops at this point, only the payment information related to C is used. This means that if an account is used to pay a supplier S by a client different than C, it will not appear on the graph. In order to add the information provided by other clients, the accounts they use to pay S are appended to the graph, along with edges that link them to S. This addition allows us to make use of the collaborative knowledge of the other clients.

If the amount of payment was available, a possible use for the feature would be to assign weight to the different edges instead of using a simple binary weight. While having no impact on the graph structure, this might indicate the accounts privileged by a client company.

```
Algorithm 1: Transaction Graph Creation
Data:
    • $C$: identifier of Client company
    • $\boldsymbol{T}$: Set of transactions
Result:
    • $\boldsymbol{V}$: Set of vertices of $G_{C,\boldsymbol{T}}$
    • $\boldsymbol{E}$: Set of edges of $G_{C,\boldsymbol{T}}$
$\boldsymbol{V_c} = C$, $\boldsymbol{V_s} = []$, $\boldsymbol{V_a} = []$, $\boldsymbol{E} = []$;
foreach $t = [C_t, A_t, S_t, d_t]$ in $\boldsymbol{T}$ do
    if $C_t = C$ then
        $\boldsymbol{V_a}$.insert($A_t$);
        $\boldsymbol{V_s}$.insert($S_t$);
        $\boldsymbol{E}$.insert(($C, A_t$));
        $\boldsymbol{E}$.insert(($A_t, S_t$));
        $\boldsymbol{T}$.remove($t$);
foreach $r = [C_r, A_r, S_r, d_r]$ in $\boldsymbol{T}$ do
    if $S_r$ in $\boldsymbol{V_s}$ then
        $\boldsymbol{V_a}$.insert($A_r$);
        $\boldsymbol{E}$.insert(($A_r, S_r$));
$\boldsymbol{N} = \boldsymbol{V_c} \cup \boldsymbol{V_s} \cup \boldsymbol{V_a}$
```

Figure 11.3: Transaction Graph Creation

11.6.2 Properties of a Transaction Graph

A transaction graph $G_{C,T}$ shows interesting properties. It is a directed graph, as the edges represent the movement of funds from a client to a supplier through an account. A transaction graph is also a *bipartite graph*. A bipartite graph is defined in ([41]) as a graph whose vertices can be divided into two disjoint and independent sets u and v and such that every edge connects a vertex in u to one in v. The transaction graph satisfies this property as the created edges are only from company to account and from account to company (no account-to-account or company-to-company edges exist in the graph). Table 11.2 shows the representation of the transaction graph T1 (shown in 11.4) as a connectivity matrix: each of the row corresponds to a company vertex, while a column represents an account vertex. The value in the row indicate the number of times a transaction has been issued involving the specified company and account.

From the connectivity matrix, it is apparent that the sole vertex representing the client company ($C1$ in the example) plays a central role in the transaction graph. Centrality is an important metric in graph as it informs how a vertex can influence its neighbors. More specifically, the *graph theoretic center* is defined in ([41]) as the vertex with the smallest maximum distance to all other vertices in

Table 11.2: Connectivity Matrix of Transaction Graph T1. Each column represents an account vertex. Each row represents a company vertex. Numbers are the number of time a transaction created an edge between the two vertices.

	A1	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**
C1	1	1	1	1	1	1	0
S1	1	0	0	0	0	1	1
S2	0	1	0	0	0	0	0
S3	0	0	1	0	0	0	0
S4	0	0	1	1	0	0	0
S5	0	0	0	1	0	0	0
S6	0	0	0	0	1	0	0

the network. This vertex is always the company vertex C in the case of a transaction graph $G_{C,T}$. We use this property to create payment patterns in order to characterize transaction graphs.

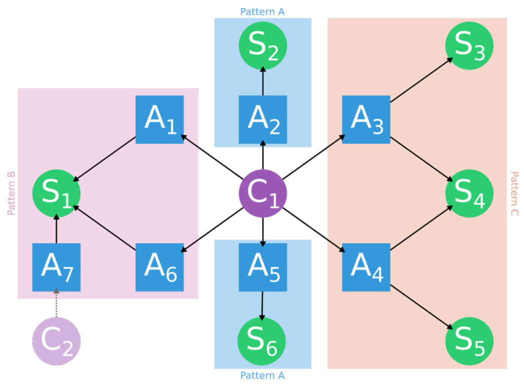


Figure 11.4: Transaction Graph T1. The transaction graph is created by assigning vertices to every account and supplier companie found in the set of transactions. Edges are created when an account and a supplier company are found in the same transaction.

11.6.3 Payment Patterns

In this subsection, we describe how we use the specific properties of a transaction graph in order to create a set of features capturing the transaction relationship between a client and the accounts used to pay its suppliers. Our approach is akin to the one developed by [24] where a similar featuring process is used to characterize ego-network of specific nodes in the graph. In order to create the features, we first remove the client company's vertex from the graph ($C1$ in Figure 11.4), thus creating a set of D of disconnected sub-graphs composed of account vertices linked to supplier vertices. Among these D sub-graphs (that we dubbed "payment patterns"), if we only take into account the type of the node ("Supplier" or "Account") and not its label ("S_1" or "A_4"), then it might occur that some of these sub-graphs are isomorphic, meaning that they share the exact same structure ([41]). It is the case for example in Figure 11.5 where the sub-graph composed by (S_2,A_2) and (S_6,A_5) are isomorphic.

This set of features can also be translated in the connectivity matrix shown in Table 11.2. Removing the central node means ignoring the first row of the matrix. A connected sub-graph can be connected in two ways: when a supplier is connected to a specific number of accounts (which is the case for S_1), or when an account is connected to a specific number of suppliers (such as A_3). The case of Pattern C is a special one where the pattern satisfies both of these conditions.

Functionally speaking, these connected sub-graphs indicate how the client interacts with its suppliers, and thus shows a "map" of the client's activity in the set of T transactions used to create the transaction graph.

We calculate the possible number of unique payment patterns that can be created for a specific number of transactions based on Algorithm 11.3. This number seems to grow at an exponential speed, meaning that for x transactions used to build the transaction graph, x^x possible unique payment patterns can be found. This number corresponds to the number of features of an histogram. This fast growth in the number of features indicates that our data point might be placed in a very sparse high-dimensional space. Thus our system might fall prey to the curse of dimensionality [276].

Table 11.3: Examples of featurized sets of transactions

Transaction set ID	**Pattern A**	**Pattern B**	**Pattern C**	**Pattern D**
T1	2	1	1	0
T2	3	3	0	1
T3	1	0	0	0

If the amount of the transactions were available, it could be used as a way to discriminate identical patterns of transactions by computing the cumulative amount found in a pattern and adding it as a feature for the anomaly detection model.

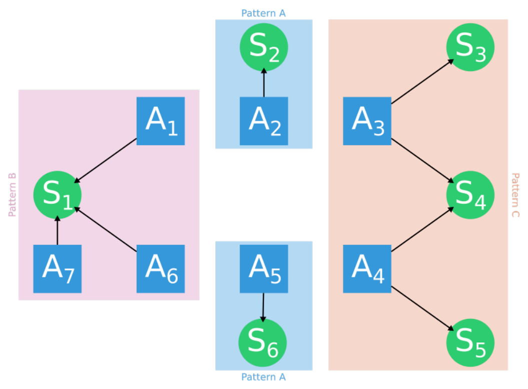


Figure 11.5: Subgraphs ('patterns') characterizing the transaction graph. Patterns are the connected subgraphs found when removing the node C from the graph. They represent suppliers and accounts found together in the set of transactions.

11.6.4 Feature Set Creation

In order to create an overview of a client's behavior through time, we first use the transactions found in each N windows created in the feature engineering process to create the N corresponding graphs centered on the client C. A "test graph" is also created for the test window. We then transform the graph into a histogram with the technique previously described, thus creating N histograms where the features are the unique connected sub-graphs (i.e unique payment patterns) and the values are the number of occurrences of the pattern in the graph. Table 11.3 shows an example of such a process with T1 representing the graph shown in 11.4 and T2 and T3 representing other graphs. Similarly, the histogram corresponding to the test graph is also created using the same process. These histograms are then used as the basic features of our anomaly detection system.

Figure 11.6 shows an overview of the feature engineering process, proposed as a reminder. First, the transactions' windows created in the pre-processing phase are converted into a transaction graph representing the relationships between a client and the accounts it uses to pay its supplier. Then the transaction graphs are in turn transformed into a set of feature vectors composed of the number of sub-graphs found in the transaction graphs.

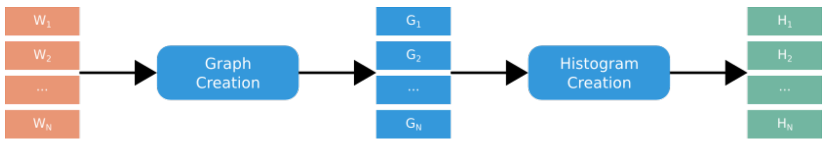


Figure 11.6: Graph histogram sequence creation. First, each of the windows of the transaction sequence is transformed into a graph composed of companies and accounts. Then the graphs are transformed into histograms using their connected subgraphs.

11.7 Anomaly Detection

In this section, we describe the anomaly detection system we use in GraphSIF in order to assert if the test graph created by the tested transaction and the most recent transactions of C is similar to the graphs found in C's behavior sequence created from C's historical transactions.

The anomaly detection system relies on two main building blocks: a parametric clustering algorithm (K-means ([155])) that create clusters of transaction graphs represented as histograms according to their similarity, and a single-layer neural network called a Self-Organizing Map ([62]) that projects multi-dimensional features such as the histograms into a two-dimensional space, in order to facilitate the computation of distance between feature vectors and to alleviate the curse of dimensionality ([276]) that states that the more dimensions, the more difficult it becomes to compute a meaningful distance between two feature vectors.

11.7.1 Overview

Figure 11.7 shows an overview of the anomaly detection process. First, the N histograms created at the end of the feature engineering phase are regrouped into clusters thanks to the K-means algorithm. Each of the histograms are associated with their clusters, and the centroid of each cluster C_i with $i \in K$ is also computed. K is the number of clusters set as the parameter for the K-means algorithm.

Then, the histograms are used to train a Self-Organizing Map (SOM). In order to do so, each of the histogram is fed to the SOM, where a unique neuron (also called "node") is activated. The weights of this node and the eight neighboring ones are then updated in order to match the values of the histogram. The operation is repeated for each of the histograms until every one of them is associated with a node. Several histograms can be associated with the same node. Finally, once the SOM is trained, the centroid of each cluster is fed to the SOM and the node activated by it is retrieved. This creates the "SOM model" that is composed of the nodes trained with the histograms along with the nodes corresponding to the centroids.

Once the SOM model created, when an histogram corresponding to a test graph needs to be evaluated, it first goes through the clustering phase undergone by the other histograms in order to be assigned a cluster c. Then, it is fed to the SOM in order to find the node activated by it. The distance between this node n_t and the node activated by the centroid of the cluster c (n_c) in the SOM (D_t) is retrieved, along with all the distance between the nodes activated by the members of c and n_c. All of these distances are then used to compute the z-score ([22]) of D_t. The z-score is a statistical measure that tells us how many variation away a data point is from the mean. The higher the z-score, the less similar an histogram is from the others.

In the remainder of this section we detail the different algorithms used to obtain the z-score from our input data.

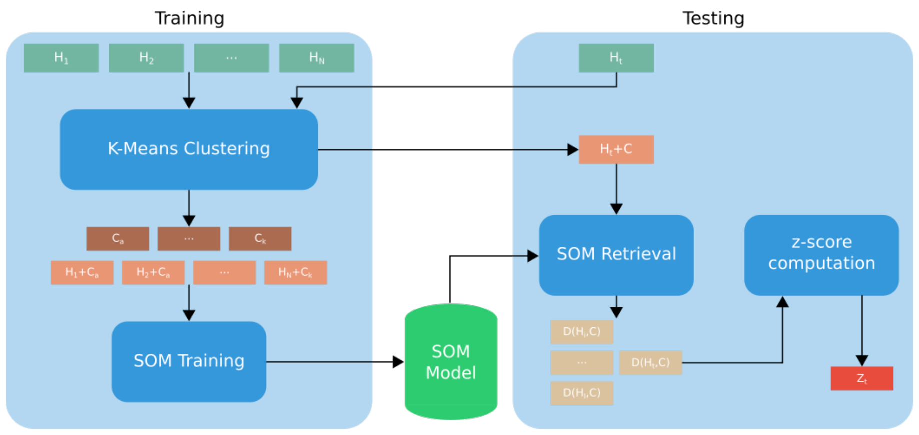


Figure 11.7: Overview of the anomaly detection process. First the histograms from the training dataset are clustered using k-means, and fed to a Self-Organizing Map. The histogram corresponding to the tested transaction is compared with the model, and its similarity with the other histograms from its cluster is computed.

11.7.2 Training

In this subsection, we detail the training of the two models (the K-Means algorithm and Self-Organizing Map)) used in the anomaly detection system. Training the models means that we use the historical histograms created at the end of the graph-based feature engineering phase to fit the models so that they accurately represent the past behavior of the considered client. The training phase is divided in three parts: the histogram clustering where the histograms are assigned a cluster, the SOM training where the weights of the nodes of the SOM are adjusted to match the value of the histograms, and finally the histograms projection where the SOM nodes corresponding to the histograms and centroids are determined in order to be used in the testing phase.

Histograms Clustering

Algorithm 2: Histograms clustering using K-Means

```
Data:
    • $\boldsymbol{H} = (h_1, ..., h_N)$: Set of histograms (observations)
    • $\boldsymbol{C^{(1)}} = (C_1^{(1)}, ..., C_k^{(1)})$: Set of centroïds at time 1
Result:
    • $\boldsymbol{S_i^{(t)}}$: histograms for each clusters.
    • $\boldsymbol{C^{(t)}} = (C_1^{(t)}, ..., C_k^{(t)})$: Set of centroïds at time $t$
1 $Init(\boldsymbol{C^{(1)}})$;
2 while Not convergence do
3     $S_i^{(t)} = \{h_p : \|h_p - C_i^{(t)}\|^2 \leq \|h_p - C_j^{(t)}\|^2 \ \forall j, 1 \leq j \leq k\}$;
      // assign the histograms to the closest centroïd.
4     $C_i^{(t+1)} = \frac{1}{S_i^{(t)}} \sum_{h_j \in S_i^{(t)}} h_j$;
      // calculate the new position of the centroïd as the mean of the
         observations in the new cluster.
5     $t = t + 1$
```

Figure 11.8: Histograms clustering using K-Means

Clustering the histograms is the first step of the training process. It consists of using the K-Means ([155]) algorithm on the set of historical transactions in order to assign them a cluster based on their similarity according to a selected distance metric. K-Means is a well-known clustering algorithm that assigns clusters to feature vectors according to their proximity to a centroid that is the mean of the member of the clusters when the algorithm reaches convergence. This proximity is computed according to a distance metric such as the Manhattan distance or the Euclidian distance. We use the Euclidian distance in our current implementation. Algorithm 11.8 shows an overview of the algorithm.

When using K-means, it is very important to carefully choose the number of clusters K so that it represents accurately the underlying distribution of the data. Due to time constraints a thorough analysis of the optimal number of parameters could not be performed. However, a preliminary experimental study conducted on a selected test company showed that $K = 3$ seems to yield the best results in terms of stability of the cluster.

The motivation behind the use of a clustering algorithm as a first step in our training process is to partition the local profile of a user in order to detect re-occurring transactions graphs. As a graph represents the interaction of a client with its supplier, it is logical to think that more than one type of transaction graph might occur in the lifetime of the client company. For example, assuming that the transactions occur homogeneously every month, two suppliers might be paid every sixth month using a specific account each, while three other suppliers might use only one account to be paid every two months. These two examples will create different transaction graphs throughout the behavior sequence, all sharing similar properties. These transaction graphs will be assigned different clusters by the clustering algorithm, thus identifying two different component of the client's behavior. Thus, clustering the graphs into different clusters allows us to further decompose this behavior.

This research could be furthered by studying the different clusters found and determining if they relate to a real behavior for the client company (such as the acquisition of materials, taxes payments, and so on). However, in the context of fraud detection, we solely focus on the fact that the clusters can be used as a point of comparison for new transactions graphs.

Alternative clustering algorithms such as the k-medoids algorithm ([233]) or x-means algorithm ([154]) might be investigated in order to optimize the clustering process.

Self-Organizing Map Training

One of the issue with using the transaction patterns described in the feature engineering phase as the dimension for our feature vectors is that we do not have direct control on the dimension of said feature vectors. Furthermore, the number of possible patterns grows exponentially with the number of transactions that are found in the set used to create the underlying transaction graph. This fact leads to two major issues: firstly, the feature vectors representing the algorithm might be projected in a high-dimensional but sparse feature space, thus resulting in artificially high distance due to the curse of dimensionality. Secondly, it is hard for a human to interpret the notion of proximity in such a high-dimensional space. In order to alleviate these two issues, we use a Self-Organizing Map, and more particularly a Kohonen network. ([174],[61],[62]). This type of network aims to organize all of the feature vectors in a two-dimensional plane according to their similarity.

A SOM is an unsupervised neural network based on competitive learning, in the form of a neural network where only one neuron (also called node) is activated at any one time. The specificity of the Kohonen network is that the single computational layer is arranged in rows and columns, and each node is fully connected to all the source nodes in the input layer. In order to train the SOM, after an initialization phase, three steps are performed until convergence: sampling, matching and updating.

In the initialization phase, each of the neurons of the computational layer of the SOM is assigned random activation weights. The values of the weights need to be reasonably close so that every neuron has a chance to be activated. The number of activation weights of a neuron is the same as the dimension of the feature space.

In the sampling phase, a sample x is drawn from the set of feature vectors X. This sample is randomly chosen so that the ordering of the set does not have any impact on the training process.

In the matching phase, the winning neuron $I(x)$ is found by comparing the weight vectors of each neuron and finding the one closest to the values of the input vector. More specifically, the similarity of the vector to a neuron j's weights is computed using a discriminant function $d_j(x) = \sum_{i=1}^{D}(x_i - w_{ji})^2$. Closely related input vectors might activate the same winning neuron $I(x)$.

In the updating phase, the winning neuron and its neighbors are updated using the equation

$$\Delta w_{j,i} = \eta(t) T_{j,I(x)}(x_i - w_{ji}). \tag{11.1}$$

In this equation, $T_{j,I(x)}$ symbolizes the topological neighborhood of the winning neurons, and is defined as $T_{j,I(x)} = e^{\frac{-S^2_{j_{I(x)}}}{2\sigma^2}}$ where σ is the size of the neighborhood of the winning neuron. $\eta(t)$ is the learning rate that dictates how the weights of the winning neuron and its neighbors gets to be updated to a value closest to the input vector. This learning rate is defined as

$$\eta(t) = \eta_0 \; e^{\frac{-t}{\tau}} \tag{11.2}$$

where η_0 is the initial learning rate and τ a parameter for the exponential decay function that decrease the learning rate over time.

The sampling phase, matching phase, and updating phase are repeated until the SOM reaches convergence, meaning that no significant modification in the weights occurs.

In our setting, the set of feature vectors X corresponds to the set of histograms H created by the feature engineering process.

Histograms Projection

Once the SOM is training and convergence is reached, an additional step is performed: each histogram h found in the set of created histograms H is fed to the SOM, and the neurons $n(h)$ activated by the histogram is associated to it. Similarly, each centroid c of k centroids created by the clustering phase are also fed to the SOM and the neurons $n(c)$ are associated with the centroids.

This phase allows us to project the histograms from their high-dimensional feature space to the 2-dimensional space of the SOM nodes, in a way that preserves their similarity. A valuable effect of this projection is that it enables a human expert to read and interpret the created map, and thus allows us to use human knowledge to understand the transactions patterns uncovered.

11.7.3 Testing

In the testing phase, a transaction is given to the anomaly detection system in order to assert its similarity with C's historical transactions. Before being submitted to the anomaly detection system, the test transaction is integrated to the most recent transactions of C and turned into a transaction histogram following the steps of the feature engineering process previously described. The result of this phase is a legitimacy score summing up how distant the transaction is from the historical ones. This phase is divided in 3 steps: the test histogram clustering, then the SOM distance retrieval, and finally the similarity computation.

Test Histogram Clustering

In this step, the test histogram h_t is assigned a cluster based on the centroids found in the training phase. It is assigned the cluster whose Euclidian distance is the closest, following the equation

$$c_{h_t} = \min(c_i : \{\|h_t - C_i\|^2 \leq \|h_t - C_j\|^2 \ \forall j, 1 \leq j \leq k\} \tag{11.3}$$

Assigning a cluster to h_t allows us to compare it to the historical transactions closest to it. Furthermore, as the centroids determined by K-means algorithm represent the mean of all the historical transactions of the cluster, it represents the representative member of the cluster. Thus, the farthest h_t is from the centroid, the less similar to the other member of the cluster it is. In other words, the further away h_t is is from the centroid, then the closer from the edge of the cluster it is, and thus is dissimilar to the other members of the cluster. However, as mentioned previously, the curse of dimensionality might hinder the computation of a meaningful distance in our case. Thus, we use the trained SOM in order to compute the similarity of h_t with the member of its cluster.

Self-Organizing Map Distance Retrieval

In this phase, we feed h_t to the previously trained SOM and thus retrieve n_{h_t} the neuron activated by h_t, and we compute $D(n_{h_t}, n_{c_{h_t}})$ the Euclidian distance between the neuron activated by h_t and the neuron activated by c_{h_t} that is the centroÃ¯d of the cluster assigned to h_t.

Once this distance is acquired, it might be tempting to use it directly as a way to determine h_t's legitimacy score, by for example assigning a threshold distance from which h_t would be considered anomalous. However, assigning a value to the threshold might prove a challenge as the distance corresponds to a neuron-to-neuron distance and not a vector-to-vector distance. Thus, it is not clear what the meaning of such a threshold would be. We propose a solution to this issue in Section 11.8.

Similarity Computation

In order to provide a more robust way to assert h_t's similarity, the distances from the other histogram members of the cluster C_{h_t} and its centroid $(\{D(n(h_i), n(c_{h_t})) \ \forall h_i \in S_{h_t}\} = \boldsymbol{D})$ are also retrieved. Once retrieved, the z-score of $D(n(h_t), n(c_{h_t}))$ with respect to $\boldsymbol{D}$ is computed. The z-score, as described in [22], is a statistical metric that corresponds to how many standard deviation away a data point is from the mean of an ensemble. In our case, it gives us insight about how far away h_t is from the centroid, with respect to all the other members of the cluster. The higher the z-score is, the more dissimilar h_t is from the other members of the clusters, and thus the more likely it is to be an anomaly.

11.8 Label Attribution

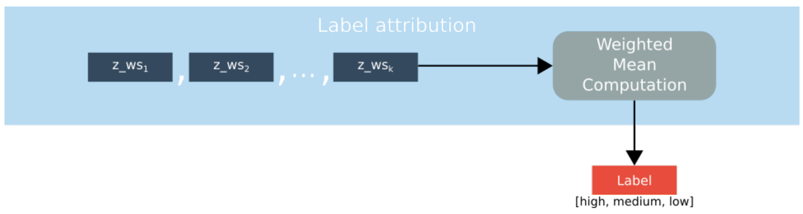


Figure 11.9: Label attribution process. A weighted mean combining the similarity score computed from various window size is used to compute the final label of the tested transaction.

The previously described training and testing algorithm uses a set of historical histograms to assert the similarity of a test histogram, through the computation of a z-score. However, the computation of these histograms is dependent on the size of the window of transaction ws that is used in the pre-processing algorithm. Indeed, the window size directly impacts the graph extracted from the window, and thus the histogram describing the graph.

11.8.1 Impact of Window Size

A small window size will create smaller graphs that encompass the short-time behavior of a client C. Furthermore, a small window size will also provide more behavior histograms to perform the clustering and SOM training, at the expense of a decrease in the complexity of patterns found in a graph, as less transactions means less possible relationships between account and supplier. Lastly, anomaly detection using graphs created from small windows are less stable as adding a single transaction can create a huge topological difference between two graphs.

On the contrary, a large window size will create larger graphs, that sums up a large number of transaction and thus the long-term behavior of the client. However, as the number of transactions needed to create the graphs will be higher, less data points will be created. As a certain amount of data point (depending on the size of the SOM) is needed for the training algorithm, very high windows size are thus not suitable for the detection system. However, larger transaction graphs means that more complex patterns might be formed, which would have been missed if smaller windows were used. Lastly, using a large window size means that the topological modification following the introduction of a single transaction might not be enough for the anomaly detection system to pick up an anomaly.

11.8.2 Optimizing Window Size

A straightforward way to determine which windows size is more suitable for anomaly detection for a specific client C would be to perform an optimization method such as grid search ([48]) or random search ([48]). However, these optimization methods rely on the assumption that target labels are available, which is not the case in our use-case. Thus, an alternate method has to be found.

Instead of trying to optimize the size of the window, a possible solution would be to perform the anomaly detection in a range of different windows size, and aggregate the results in a way similar to bagging [95]. This way, the anomaly detection system would be able to draw its conclusion from both small size transaction graphs and high size transactions graph when assigning the legitimacy score of a transaction.

As a way to perform this aggregation, the following algorithm is proposed. For every size ws in $\boldsymbol{W}$ of length $l(\boldsymbol{W})$, the pre-processing phase, feature engineering phase and anomaly detection phase of the anomaly detection system are performed, effectively creating $l(\boldsymbol{W})$ anomaly detection systems, and a z-score is computed for a transaction t from each of them. Then, the following process is applied:

1. The z-scores undergo a discretization process when the score is turned into one of the three legitimacy labels ("high","medium","low"). In order to do so, two risk thresholds ($0 < r_1 < r2 < 1$) are used as parameters for the threshold function. These risk thresholds represent the fraction of the maximal z-score corresponding to each of the legitimacy labels. As a rule of thumb, a z-score of 3 indicates that a value is an outlier with respect to a give data set. Thus, a risk score $r_1 = 0.2$ indicates that if the z-score of a given value is smaller than $0.2 * 3 = 0.6$ then it is considered legitimate by the anomaly detection system. The risk thresholds are defined by the investigation team, as it relies on the costs implied by a false positive (legitimate transaction mislabeled as fraud) or a false negative (fraud mislabeled as legitimate transaction). These costs depend on factors that reside outside of the scope of the anomaly detection system, and thus need to be asserted by a team of experts.

2. Each of the label is assigned a weight (w_h,w_m,w_l) that represent a bonus in the overall legitimacy score. These weights can be parameterized by the investigator in order to control the sensitivity of the system. Usually, $w_h > w_m > 0 > w_l$ so that "high" legitimacy labels pull the score up and "low" legitimacy label decrease the overall legitimacy score.

3. The sum of the $l(\boldsymbol{W})$ weights is computed and normalized so that an overall legitimacy score $0 < L_{\boldsymbol{W}} < 1$ is calculated.

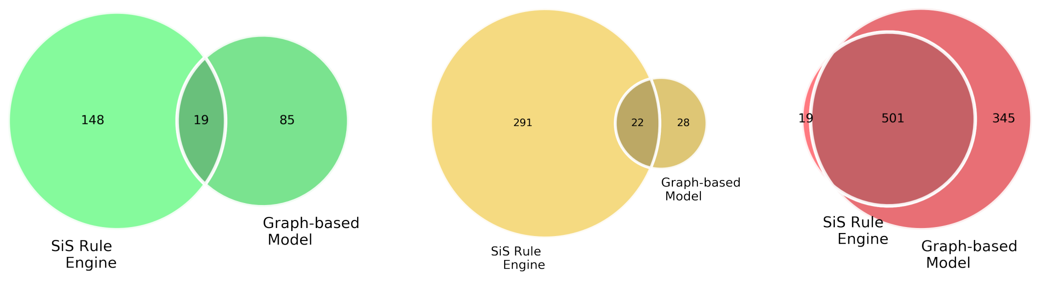
$L_{\boldsymbol{W}}$ can also be discretized using a threshold function if the need to provide labels is found. This threshold function can use as input a list of threshold risks $\delta_1, \delta_2, ..., \delta_n$ corresponding to the operational needs of the fraud detection team. Alternatively, a voting system might be used in order to aggregate the value of each of the anomaly detection systems. This label attribution phase thus alleviates the need to search for an optimal value of ws and allows the anomaly detection to be performed on both short-term and long-term transaction behavior of the investigated client C.

If the amount of transactions were available, it could be used to create a cost function that would impact the thresholding parameters so that more attention would be given to high-value transactions.

11.9 Experimental Results

For a complete description of the experiment setup and results, we refer the reader to the article [71] on which this chapter is based.

We evaluate the performance of GraphSIF on the subset of transaction issued by a single client C. In this evaluation, only the transaction issued by C in the Audit dataset are considered. This subset contains 1000 test transactions to evaluate the performances and 218,325 historical transactions for the History Dataset to train the model.



(a) High (b) Medium (c) Low

Figure 11.10: Distribution of legitimacy labels.

High Legitimacy Label First, we consider the results given by both the expert system and GraphSIF concerning the high legitimacy labels. Figure 11.10 shows a Venn diagram indicating the distribution of the high legitimacy label. The consistency of the two set is not high, only 19 out of the 167 transactions given a "high" legitimacy label by the expert systems are given the same label by GraphSIF. However, this inconsistency might be explained by the fact that the expert system relies on knowledge internal to the platform, in the form of a registration of secured suppliers, that is not available for the graph-based model.

Table 11.4 shows the confusion matrix indicating the complete distribution of each label. Most of the high legitimacy labels (128 out of 167) have been labeled as low legitimacy transactions by GraphSIF. This behavior is consistent with the hypothesis of the expert system using additional knowledge to perform its classification.

Table 11.4: Confusion Matrix - SiS Rule Engine and GraphSIF

GraphSIF → Expert system ↓	High	Medium	Low
High	**19**	20	128
Medium	74	**22**	217
Low	11	8	**501**

Medium Legitimacy Label Then, we consider the results given by both the expert system and GraphSIF concerning the medium legitimacy labels. Figure 11.10 shows the distribution of the medium legitimacy label. For the expert system, a medium label indicates a lack of knowledge. Almost a third (313 out of 1000) of the transactions have been assigned this label by the expert system. On the contrary, a medium label doesn't indicate a lack of knowledge for GraphSIF, but rather informs the user that a specific transaction is slightly unusual. Thus the consistency between the two sets is not really expected. However, the low number of transactions labeled with the medium label by GraphSIF (50 out of 1000) seems to indicate a higher assertiveness of the proposed model. Table 11.4 shows that most (217 out of 291) of the medium labels given by the expert system where assigned a low legitimacy label by GraphSIF.

Low Legitimacy Label Finally, we consider the results given by both the expert system and GraphSIF concerning the low legitimacy labels. Figure 11.10 shows the distribution of the low legitimacy label. There is a high consistency between the expert system and GraphSIF concerning this set of suspicious transactions, as 501 out of 520 transactions where given the low legitimacy label by both of the detection systems. The 345 transaction assigned a low legitimacy transaction by GraphSIF and not the expert system come from the two other sets. Furthermore, the last row of Table 11.4 shows that only a handful of transactions labeled as low by the expert system have been given another label by GraphSIF. It is possible that the 11 transactions given a low legitimacy label by the rule engine could use knowledge not available for GraphSIF, such as when a supplier closes an account.

11.9.1 Discussion

While results shows that GraphSIF is able to provide consistent results in terms of consistency with the expert system, the most pressing issue is the lack of ground truths about actual legitimate and fraudulent transaction. While SiS-id's expert system provides an alternative to expert knowledge regarding the labeled transaction, this system is prone to error, most notably because fraud is dynamic and often changes faster than the rules of an expert system. Furthermore, comparing GraphSIF graph-based approach and SiS-id expert system is difficult, as they both rely on different types of knowledge (expert-based vs data-driven). However, in the absence of a dataset containing trusted labels, comparing our systems with SiS-id's expert system is the only way to propose an evaluation of their performance.

11.10 Conclusion

In this work, we introduce GraphSIF, a novel feature-engineering process that creates a feature vector based on the relationships between a client company and the accounts it used to pay its supplier companies, providing a new tool to describe the underlying transaction mechanism involved in their interaction.

In conclusion, we used the temporal information contained in the transactions of the History dataset to create a behavior sequence composed of the transactions emitted by a client aggregated in several bounded time windows. We showed how to use this behavior sequence to create a data model based on Self-Organizing Maps representing the behavior of a client company through time. We then used this data model to infer the legitimacy of new transactions using the K-means clustering algorithm, along with an aggregation algorithm allowing us to combine the results obtained for different window sizes in a comprehensive score.

Part V

Conclusion and Future Research Directions

Chapter 12

Conclusion

In this chapter, we summarize the conclusions of the selected contributions that we have included in this habilitation thesis. We present a general conclusion and some perspectives in Section 12.4.

12.1 Privacy-Preserving Reputation Systems

A Survey of Decentralized Privacy-Preserving Reputation Systems

We conducted a fine-grained analysis and comparison of 44 privacy-preserving reputation systems. We established several categories of systems according to their security mechanisms and classified the privacy-preserving reputation systems according to these categories. Our detailed comparison of privacy-preserving reputation systems in a normalized manner using our analysis frameworks reveals the differences between the systems in the literature as well as their chronological evolution. Our fine-grained analysis, comparison, and discussion (Section 13.1) led to the identification of a number of insights into this area of research. We noted that one of the future directions is to leverage the blockchain technology to its full potential and build truly trustless systems. Moreover, we identified authorizability as one of the important properties that needs to be addressed by systems in the future. Lastly, analyzing the systems in terms of their countermeasures against common attacks, we observed that designing systems that provide comprehensive protection against a broad range of attacks is an evident direction for future research in the area.

A Decentralized Privacy-Preserving Reputation Protocol for the Malicious Adversarial Model

We presented a privacy-preserving reputation protocol for the malicious adversarial model. The protocol counters attacks by malicious agents such as submitting illegal feedback values or making erroneous computations. The characteristics that differentiate the protocol from other protocols in the literature include: (1) full decentralization, (2) no need for trusted third parties and specialized platforms, (3) low communication complexity. Our experiments on three real and large trust graphs demonstrated the validity of the two hypotheses that the Malicious-k-shares protocol is based on: (1) A source agent can preserve its privacy by trusting on only k fellow source agents, where k is much smaller than $n-1$, the size of the set of all fellow source agents. (2) Accurate reputation values can be computed even if the source agents whose privacy can not be preserved abstain and thus do not provide their feedback values.

A Privacy-Preserving and Reputation-Aware Mobile Participatory Sensing System

We proposed a privacy-preserving reputation system, PrivaSense, for participatory sensing applications. The system employs separate registration and authentication phases that ensure participants' anonymity and improve the system resilience against the Sybil and replay attacks. In addition, a privacy-preserving mechanism is used for the contents of the participants' contributions which prevents adversaries from using the data to infer the identity of participants. Moreover, data reliability is ensured due to the incorporation of a reputation system. Finally, PrivaSense adopts a mechanism to prevent the reputation scores of participants from revealing their identity. The experiments conducted on a real dataset demonstrate that the PrivaSense system ensures good anonymity and un-linkability with a much lower mean square error introduced into the aggregated data.

Anonymous Voting using Distributed Ledger-assisted Secure Multi-Party Computation

Bringing together the blockchain technology and secure multiparty computation, we constructed a highly transparent referendum protocol that allows participants to autonomously verify proceedings and outcome. Traditional (t, n) threshold based systems rely on collusive attacks being unlikely due to the selected parameters. However, our system goes further and places all communication on a single public immutable blockchain such that the communication can be inspected at any stage of the execution of the protocol to validate correctness. Other than an anonymous credential issuer, the system eliminates the need for any trusted third parties.

Collusion-Resistant Worker Set Selection for Transparent and Verifiable Voting

We proposed two solutions to provide better collusion-resistance in distributed protocols where a subset of workers needs to be selected from the set of participants. Workers are the nodes who provide intermediate processing in the protocol. Firstly, we used the blockchain's immutability and ordering to design a collusion-resistant decentralized protocol to randomly select workers. Secondly, we considered a social graph representing participants and proposed an algorithm to distance workers from each other in the graph. Based on a social graph and our problem's constraints, we computed the size of the largest clique of workers to evaluate the number of workers' bounds for which our solutions were resistant to collusion with high confidence. The decentralized random worker selection works from low numbers of workers to an upper limit which depends on the size of the graph's largest clique. As expected, the method taking advantage of the graph's structure provides better results: it distances workers better for a wider range of numbers of workers.

12.2 Privacy-Preserving Message Routing

A Study of the Unwillingness of Nodes to Participate in MDTN Routing: Selfishness and Privacy

We studied the reasons for the unwillingness of nodes to participate in MDTN routing. We identified the factors of selfishness and privacy as the two primary causes. We developed a classification of the aspects of selfish behavior. We then classified the existing strategies for preventing selfish behavior into three categories: barter-based, credit-based and reputation-based. We subsequently analyzed the mechanisms of the proposed strategies and pointed out the problems in each category. We then conducted an experiment to investigate the performance of the representative strategies for preventing different types of selfish behavior. For privacy, we classified the existing privacy-preserving protocols for MDTNs according to their specific privacy objectives: identity, location, message content, and relationships. We reviewed the various strategies proposed in the literature for preserving the privacy of nodes under each of these categories. We also presented an analytical comparison of the privacy-preserving protocols.

Privacy-Preserving Routing in Mobile Delay Tolerant Networks

We described the 4PR protocol, which provides privacy-preserving probabilistic prediction-based routing in mobile delay tolerant networks. 4PR is similar to prior prediction-based protocols (e.g., PRoPHET and Bubble), which take advantage of the mobility patterns of nodes to route messages. However, the 4PR protocol forwards messages by comparing aggregated information about communities instead of individual nodes, in order to preserve their privacy. Our experimental evaluation using a well established community-based mobility model demonstrates that 4PR is comparable to the above noted protocols in terms of performance. Yet, 4PR preserves the privacy of nodes by hiding their individual mobility patterns, whereas the prior protocols do not.

12.3 Privacy Preservation in Financial Networks

Privacy Considerations for a Decentralized Lending Platform

We presented an approach to calculate the users' trustworthiness on peer-to-peer lending platforms. This approach is neither collateral nor credit score based. Instead, we calculate a user's social score that relies on the social capital theory and consequently the information retrieved from the user's social network accounts. The social scoring formula is implemented in a smart contract. We quantified the amount of gas that is consumed by the smart contract. Moreover, we also evaluated the execution of the functions on a real social network dataset. An experiment demonstrated how we can use analysis of social network data to determine optimal thresholds for a platform in production.

Analyzing Flow of Payments in a Business-To-Business Network to Detect Supplier Impersonation

We introduced GraphSIF, which comprises of a novel feature-engineering process that creates a feature vector based on the relationships between a client company and the accounts it used to pay its supplier companies. This provides a new tool to describe the underlying transaction mechanism involved in their interaction. GraphSIF analyzes data that has been heavily anonymized in order to preserve the privacy of the participating companies. We used the temporal information contained in the transactions of the real "History" dataset to create a behavior sequence composed of the transactions emitted by a client aggregated in several bounded time windows. We showed how to use this behavior sequence to create a data model based on Self-Organizing Maps representing the behavior of a client company through time. We then used this data model to infer the legitimacy of new transactions using the K-means clustering algorithm, along with an aggregation algorithm allowing us to combine the results obtained for different window sizes in a comprehensive score. Experiments showed high consistency of GraphSIF's results with the ones from the SiS-id expert system.

12.4 General Conclusion and Perspectives

In this habilitation thesis, we have described the research area of privacy preservation in trust-deficient decentralized systems. We discussed that decentralized systems inherently favor several desirable properties such as fault tolerance, attack resistance, censorship resistance, openness, autonomy, etc. Decentralized systems have grown in popularity since the advent of distributed ledger and blockchain technologies. These technologies enable decentralized systems to securely function even in environments where nodes are unable to trust each other. A notable application area of decentralized systems that we discussed is Web3, which envisions a version of the web where users own the content that they create or generate and determine how they share it. This would be an alternative to the many centralized web-based platforms these days that own and monetize user data. We further discussed that user privacy is a major concern when users need to share personal and sensitive data in order to be able to use services in centralized as well as decentralized systems. Privacy preservation, that is protecting users' private data while allowing them to still use services, is rendered particularly difficult when there is a deficiency of trust between nodes. The trust deficiency may range from full trust in an entity to no trust in any entity in the network, or in other words, complete trustlessness.

We have included several works in this habilitation thesis that address the problem of privacy preservation in trust-deficient decentralized systems. These works are categorized into three categories: privacy-preserving reputation systems, privacy-preserving message routing, and privacy preservation in financial networks. In the area of privacy-preserving reputation systems, we proposed several protocols that have advantageous properties, such as security under the difficult malicious adversarial model, resistance against linkability and re-identification in a participatory sensing application, and transparency and verifiability in a voting protocol. Moreover, we discussed approaches for selecting nodes for intermediate tasks in the protocols such that the risk of collusion of nodes is minimized. In the area of privacy-preserving message routing, we observed that prediction-based routing protocols in the literature perform well in mobile delay tolerant networks. Unfortunately, they compromise user privacy due to the use of mobility and social interaction patterns. We proposed two protocols (4PR, and its predecessor 3PR) that route and deliver messages with comparable efficiency while preserving privacy of the users. They do so by securely aggregating and comparing information about communities instead of individual users. In the area of privacy preservation in financial networks, we described a decentralized peer-to-peer lending platform that computes a social trustworthiness score as an alternative to collateral or credit history. We suggested privacy-preserving mechanisms that enable a borrower to let the lender verify their social data without disclosing it. Another application that we addressed was supplier impersonation fraud detection in the business-to-business context. We proposed an approach based on the analysis of heavily anonymized data in order to preserve the privacy of the participating companies.

The need for privacy-preserving solutions for decentralized systems is becoming increasingly crucial due to the rising popularity and use of these systems. Awareness is also growing among users about privacy issues and thus they are increasingly demanding systems that respect their privacy. Additionally, systems that are trustless or at least do not make strong trustfulness assumptions have shown robustness and are becoming more desirable. We thus foresee many opportunities for continued research work in the area of privacy preservation in trust-deficient decentralized systems. In the next chapter, we have identified a number of directions for future research. In the specific area of privacy-preserving reputation systems, these include building fully trustless systems, achieving essential security properties (such as authorizability), and defending against attacks on reputation while preserving privacy. In the more broad area of privacy preservation in blockchains, we identify research directions that include user anonymity, transaction confidentiality and unlinkability, accountability, and smart contract privacy. These objectives need to be achieved while ensuring the appealing properties of blockchains such as decentralization, immutability, transparency, verifiability, trustlessness, etc. Another important avenue is secure and privacy-preserving blockchain interoperability, which aims at enabling transfer of user assets across multiple independent blockchains while ensuring security and privacy.

Chapter 13

Future Research Directions

In this chapter, we first discuss future research directions in the specific area of privacy-preserving reputation systems. In the second part of this chapter, we then identify future research directions in the area of privacy preservation in blockchains, which is a much broader area with many open problems and opportunities for future research work.

13.1 Privacy-Preserving Reputation Systems

In this section, we discuss in detail the insights that we can draw from our survey in Chapter 3 on privacy preserving reputation systems. This discussion leads to the identification of several future research directions in this area. We previously summarized these research directions in Table 3.7. Each of the research direction is labeled with an ID (Di).

Our first observation relates to the utilization of blockchain by privacy-preserving reputation systems. We note that the advent of the blockchain technology has provided a fresh impetus to research on privacy-preserving reputation systems. A majority of the systems published since 2016 utilize blockchain as one of the building blocks. We found 15 privacy-preserving reputation systems that are blockchain-based. In contrast, we discovered only 6 systems developed since 2016 that do not utilize blockchain. The reasons for the adoption of blockchain are evident. For example, in the case of Schaub et al.'s [255] system, using blockchain enables the system to provide the property of trustlessness, which was not offered by any prior systems. Another example is the system by Schiedermeier et al. [256], which is able to guarantee transparency and immutability by employing a blockchain. These properties are mostly absent in pre-blockchain systems.

Despite the successful application of blockchain, we do note that the development of non-blockchain-based privacy-preserving reputation systems still holds importance (D1). We can cite a couple of reasons. Firstly, blockchain can be an expensive building block to rely on in terms of the resources consumed. The computing cycles and the network bandwidth spent, and more worryingly the carbon footprint of popular blockchain-based systems such as Bitcoin, remain a significant concern [275]. Secondly, certain applications do not benefit as much as others from the decentralization and the trustlessness that blockchain offers. One such application is mobile participatory or crowdsensing. We note that two (Ma et al. [200] and Mousa et al. [218]) of the six non-blockchain-based privacy-preserving reputation systems since 2016 that have been analyzed are for this application area. They both employ a centralized architecture due to the nature of the application, which collects reports from mobile users and centralizes the data for subsequent analysis. We acknowledge that at least three (Zhang et al. [293], Jo and Choi [159], and Zhao et al. [296]) of the blockchain-based systems included in the survey also target the participatory sensing application area. These systems benefit from the smart contract functionality of blockchain technology to transparently manage the reputation of participants. However, we can observe that all three systems employ centralized TTPs in their architecture and thus do not take full advantage of the decentralization and trustlessness properties of blockchain.

Our above observations lead us to another notable and perhaps undesirable trend. Fully decentralized systems have existed since before blockchain. A key advantage that blockchain is able to offer in addition to decentralization is trustlessness. However, we observe that among all the blockchain-based systems analyzed, only one system (Schaub et al. [255]) benefits from this novel trust model to propose a fully trustless privacy-preserving reputation system. The system by Dimitriou [96] is another one that is primarily trustless, but it relies on a TTP for one of its operations. Other blockchain-based systems do benefit in part from the trustlessness of blockchain, but end up proposing hybrid trust models that include arbitrary k trusted users, chosen k trusted users, or TTPs. We believe that one of the future directions in this area of research is to leverage the blockchain technology to its full potential and build truly trustless systems (D2).

Next, we look at the success of the surveyed systems in guaranteeing the security of users. As discussed earlier in Section 3.3, the objectives of security include privacy and integrity. We first address user anonymity-oriented systems. In terms of privacy properties, we observe that all the surveyed systems guarantee user-pseudo unlinkability (26 systems). This is to be expected since this is a vital goal of user anonymity-oriented systems. Moreover, a high majority of the systems enable multiple pseudonyms (23 systems), pseudo-pseudo unlinkability (21 systems), and rater anonymity (22 systems). This is another positive sign indicating success of the systems toward providing strong privacy to the users. On the other hand, we note that much fewer systems aim for guaranteeing ratee anonymity (14 systems) and inquirer anonymity (7 systems). These properties have been ignored by a large number of the systems even though these are important properties for the privacy of roles other than the raters. We can identify inclusion of these objectives in future privacy-preserving reputation systems as another direction of research (D3). Reputation transfer and aggregation is another property that is offered by some systems but not provided by most others. We believe that this is an important property for long term sustainable privacy in the system and should thus be given priority as well (D4).

Moving to the properties of integrity, we are pleased to observe that almost all systems (25) enforce unforgeability, an essential property for the correct functioning of the user anonymity-oriented systems. Unfortunately, the assessment is not as bright for the rest of the integrity properties. There are 9 or less systems implementing the properties of either distinctness, accountability, or verifiability. The property of authorizability is offered by only 15 of the systems that we have analyzed. This is a worrisome figure since we believe that authorizability must be a critical feature for all privacy-preserving reputation systems. Absence of this property can allow an adversary to take unfair advantage of anonymity and mount attacks such as ballot stuffing and slandering. The somewhat encouraging news is that if we consider only the subset of systems since 2016, we can observe that 8 out of the 12 systems offer authorizability. Thus, the trend is moving favorably toward including authorizability and should continue to do so (D5).

We now discuss the feedback confidentiality-oriented systems and their success in enforcing the listed security objectives. Considering the privacy objectives, we observe that all surveyed systems (18 systems) ensure that feedback confidentiality is maintained even if the adversary has access to intermediate information revealed during the execution of the protocols. This is the primary privacy objective of feedback confidentiality systems. Therefore, this property is the minimum expectation from any system. In contrast, we observe that less than half of the systems can guarantee to some degree that an adversary will be unable to infer the feedback values from publicly available information, which includes the computed reputation scores. However, this issue is generally of concern when the number of participants is low. Therefore, even if future systems do not ensure this property, they should take measures to either warn users when their privacy is at risk or prevent execution of protocol instances with few participants (D6). The property of privacy of relationships concerns a subset of the systems that rely on relationships between users for privacy preservation. We observe that only 3 systems are able to satisfy this property to some extent. Future systems should protect the privacy of relationships in addition to the confidentiality of feedback (D7).

Looking at the integrity objectives, we appreciate that almost all systems fully enforce correct computation as well as guarantee that submitted feedback will respect the correct range. This is a reassuring trend since these two properties imply that systems are able to produce correct reputation scores despite the confidentiality of the feedback values. Regrettably, similar to user anonymity-oriented systems, the feedback confidentiality-oriented systems also largely ignore the properties of authorizability (6 systems) and verifiability (5 systems). Even if we consider recent feedback confidentiality-oriented systems since 2016, we observe that only 4 out of the 9 systems fully satisfy the property of authorizability. As we argued earlier, this is an important property. Therefore, future work on feedback confidentiality-oriented privacy-preserving reputation systems should focus on its inclusion (D8).

Lastly, we discuss the systems in terms of their countermeasures against common attacks as analyzed in Table 3.6. We observe that the number of systems implementing countermeasures against these attacks is fairly low all across the board. This is particularly true for systems that propose strong countermeasures. A majority of the systems shows some level of resistance to the Sybil attack (32 systems, out of 44), ballot stuffing (31 systems), slandering (29 systems), and whitewashing (25 systems). Defenses against other attacks are mostly overlooked: oscillation (12 systems), random ratings (15 systems), and free riding (7 systems). The figures are starkly lower when we consider only systems that offer strong countermeasures. For example, no more than 9 systems implement strong countermeasures against any of the following attacks: ballot stuffing, slandering, oscillation, random ratings, and free riding.

Moreover, Table 3.6 reveals that only two systems (Mousa et al. [218] and Benthencourt et al. [49]), out of the 44 systems analyzed, provide somewhat comprehensive resistance to the attacks. However, both these systems employ TTPs in their architecture. None of the systems with a fully decentralized architecture or with less intrusive trust models offers resistance to the full range of attacks. Table 3.6 further shows that there is no noticeable improvement in recent systems toward offering better resistance to these attacks.

There is clearly more work that needs to be done in the area of privacy-preserving reputation systems in terms of defenses against attacks other than breach of privacy. Privacy-preserving reputation systems are fundamentally reputation systems and their overall success thus relies on countering their basic challenges as well. One possible reason for the non-inclusion of robust protection against common attacks is that anonymity and privacy add further obstacles to preventing attacks such as ballot stuffing, slandering, random ratings, free riding, and others. An adversary may exploit the anonymity and privacy offered by a system to mount these attacks while simultaneously foregoing accountability. From these observations, an evident direction for future research in the area is designing systems that provide comprehensive protection against the broad range of attacks faced by reputations systems (D9). This is particularly true for decentralized systems, none of which were found to offer comprehensive countermeasures.

13.2 Privacy Preservation in Blockchains

Permisionless blockchains such as Bitcoin and Ethereum enable users to transact in an open and transparent manner. This is done in a fully decentralized environment without reliance on third parties as a source of trust. However, the openness and transparency of permissionless blockchains implies that the users' privacy and confidentiality may be negatively impacted. Transactions published on the blockchain may contain information that is considered sensitive by the users, for example, transaction amount, date and time, recipient, smart contract functions called, etc. Moreover, attackers can analyze the transaction graph of a blockchain and discover information such as the association between a user's multiple transaction addresses, the transaction patterns, etc. Even though the users use pseudonymous identities, this analysis can lead to the inference of their true identities.

In order to address this privacy challenge, privacy preservation in blockchains aims to achieve the following objectives.

- Transaction confidentiality: The data of the transaction between two users should not be accessible to unauthorized users.
- Anonymity: A user is not uniquely identifiable in an anonymity set, which is a group of fellow users on the blockchain in a given context.
- Transaction unlinkability: A transaction cannot be linked to even a pseudonym of a user.
- Smart contract privacy: Maintain the confidentiality of the information associated with a smart contract such as function calls, variable data, user addresses, etc.
- A parallel goal is to achieve the above properties while preserving the fundamental blockchain attributes: decentralization, autonomy, openness, immutability, transparency, verifiability, etc.

Some well known privacy-preserving blockchain systems include Monero, Zcash, and Secret Network. Privacy-preserving blockchains rely on cryptographic building blocks for security. These include one time public and private keys, ring signatures, zero-knowledge proofs, zk-SNARKs, zk-STARKs, homomorphic encryption, and Trusted Execution Environments (TEEs). Since cryptographic building blocks are often computationally very expensive, this results in privacy preserving blockchains being relatively less efficient and less scalable than their non-privacy preserving counterparts. Moreover, privacy-preserving blockchain systems sometimes end up relying on strong trust assumptions. This is the case in Zcash, which initially required a trusted setup due to its reliance on zk-SNARKs. With Network Upgrade 5 (NU5), Zcash has attempted to move away from the requirement of a trusted setup. However, there is often a tradeoff between trust and efficiency.

An important avenue of research is to develop blockchain systems that are privacy-preserving as well as fully trustless, efficient, and scalable.

Another theme in this area is blockchain interoperability. A large number of alternative cryptocurrencies have been proposed since the inception of Bitcoin. At the time of this writing, coinmarketcap.com lists over nine thousand established cryptocurrencies. Bitcoin and other alternative cryptocurrencies are focused solely on monetary transactions. Whereas, blockchains such as Ethereum and Neo offer Turing-complete scripting languages and smart contracts that enable the development of more general purpose decentralized applications.

The existence of a wide variety of blockchains creates diversity and allows users to select the most suitable solution. However, the multitude of blockchains also creates the problem of fragmentation and suffers from the inability of an application on one blockchain to interact with those on other blockchains. Blockchain interoperability is a topic of research that seeks to find solutions for enabling blockchain-to-blockchain interactions.

There are several scenarios where interactions between blockchains are desirable. Three of these scenarios are discussed below:

- Exchanging tokens between users who may be on different blockchains. For example, Alice, who has a balance in Bitcoin, being able to send her Bitcoins to Bob, who has an account on Ethereum. The expected outcome is that Bob would receive the equivalent amount in Ether, Ethereum's native cryptocurrency, and Alice's spent Bitcoins would be simultaneously removed from the Bitcoin blockchain.
- Invoking smart contract functionality and obtaining returned values across blockchains. A smart contract on the Ethereum blockchain is currently able to call the functions of other smart contracts that are deployed only on the Ethereum blockchain. This is an undesirable constraint since smart contracts on other platforms offer valuable functionality and services that Etherum applications could benefit from and vice versa.
- Validating the state of a variable or complex data that exists on another blockchain. The guarantees provided by a blockchain such as integrity and immutability apply only to data that are stored on the blockchain. Any data that is outside the blockchain, even if it is on another blockchain, cannot be generally validated for these properties.

The challenge is to address these scenarios in a trustless, secure, and privacy-preserving manner. Several solutions have been proposed for blockchain interoperability, however, many of them do not fulfill one or more of these requirements.

The three main current strategies for blockchain inter-operation are [142]: notary schemes, relays and sidechains, and Hashed Time-Lock Contracts (HTLC). A notary is a trusted third party that facilitates communication across blockchains. This solution makes strong trust assumptions and clearly does not qualify as trustless. The next solution, that is, relays and sidechains, enable a mainchain to maintain a ledger of assets in a sidechain. This allows the mainchain to observe, verify, and accept inter-chain transfers with the sidechain. However, this approach is limited to interoperability with sidechains. Moreover, although relays and sidechains do away with the requirement of trusting a third party, there are a number of attacks that can compromise the security of this approach. These attacks include collusion, double spending, and timing attacks. The third solution, that is, Hashed Time-Lock Contracts or HTLCs, are smart contracts that enable atomic transactions between blockchains by making use of timelocks and hash locks. A user is required to provide a cryptographic proof of a transfer before a timeout. HTLCs are also vulnerable to many security attacks, which include the wormhole attack, collusion, denial of service, double spending, no transaction finality, etc. Additionally, it has been demonstrated [127, 201] that HTLCs may compromise the privacy of users by leaking their identifiers.

Polkadot and Cosmos are two platforms that are built with blockchain interoperability as a primary goal. These platforms have attempted to address some of the above challenges. However, blockchain interoperability has yet to achieve maturity and thus remains an active area of research.

Another important theme of research that we would like to touch on is accountability in privacy-preserving blockchains. As discussed earlier in this section, the objectives of privacy-preserving blockchains include user anonymity, transaction confidentiality, and transaction unlinkability. Well intentioned users can benefit from these guarantees in order to protect their private information while conducting legitimate activities. However, the anonymity and privacy accorded by such blockchains could also end up being exploited by dishonest users for unlawful activities. The goal of accountability in privacy-preserving blockchains is to fully protect the privacy of honest users while allowing the possibility of revealing the identity of an anonymous user who is found to engage in unsanctioned dealings.

One such system is proposed by Damgård et al. [88], which manages user identities on a blockchain with the aim of balancing privacy and accountability. Their approach is adopted by the recently launched Concordium blockchain. The system introduces two new types of parties called *identity providers* and *anonymity revokers*. The identity providers are trusted authorities with whom users need to register by disclosing their true identities. User accounts are granted based on this identity verification. However, information about their real identity is encrypted using a threshold encryption scheme. This encryption can be lifted by the anonymity revokers if a sufficient number of them agree to do so after suspecting the user of wrongdoing. Otherwise, the system employs cryptographic building blocks such as blind signature schemes and zero-knowledge proofs to ensure that a user remains anonymous. The anonymity of the user and the privacy of their transactions is respected as long as the threshold of the anonymity revokers is not met.

Although the system by Damgård et al. [88] offers accountability in addition to privacy, this property is gained at the expense of placing some centralization and trust in entities such as the identity providers and the anonymity revokers. Moreover, these entities are considered to be semi-honest by the adversarial model of the system. Actively malicious or corrupted identity providers or anonymity revokers could disrupt the system.

A direction for future research is to design blockchains that can provide accountability in addition to privacy under strong adversarial models (such as the malicious adversarial model). This should be done while adhering to the goals of achieving decentralization and trustlessness.

Bibliography

[1] Global overview of covid-19 impact on elections. https://www.idea.int/news-media/multimedia-reports/global-overview-covid-19-impact-elections. Accessed: 2021-02-16.

[2] How the much-litigated ballot deadlines affected the us elections. https://www.theguardian.com/us-news/2020/dec/21/us-election-ballot-deadlines-impact. Accessed: 2021-02-16.

[3] Project jupyter. https://jupyter.org/. Accessed: 2021-02-16.

[4] Python igraph library. https://igraph.org/python/. Accessed: 2021-02-16.

[5] Stanford's facebook-combined dataset. http://snap.stanford.edu/data/ego-Facebook.html. Accessed: 2021-02-16.

[6] Twitch social networks. http://snap.stanford.edu/data/twitch-social-networks.html. Accessed: 2021-02-16.

[7] graph-tool python library, February 2021. https://graph-tool.skewed.de/.

[8] Postal mail delivery still facing delays as election nears, senate report finds, February 2021. https://www.forbes.com/sites/alisondurkee/2020/10/09/postal-service-mail-delivery-still-facing-delays-as-election-nears-senate-report-finds.

[9] Businesswire: Global peer to peer (p2p) lending market trends, growth, opportunity report 2020-2025, March 2022. https://www.businesswire.com/news/home/20201215005523/en/Global-Peer-to-Peer-P2P-Lending-Market-Trends-Growth-Opportunity-Report-2020-2025.

[10] Challenge response authentication mechanism (cram), September 2022. https://www.geeksforgeeks.org/challenge-response-authentication-mechanism-cram/.

[11] Coinloan: Peer-to-peer lending platform, March 2022. https://coinloan.io/.

[12] Cryptographic hash functions, March 2022. https://www.ics.uci.edu/~keldefra/teaching/fall2016/uci_compsci134/slides/LEC5-KED.pdf.

[13] Cryptography hash functions, March 2022. https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.htm.

[14] Fico score, March 2022. https://www.fico.com/en/products/fico-score.

[15] Global social media stats, March 2022. https://datareportal.com/social-media-users.

[16] Inlock: Peer-to-peer lending platform, March 2022. https://inlock.io/.

[17] Lendingclub, March 2022. https://www.lendingclub.com/.

[18] Prosper: Peer-to-peer lending platform, March 2022. https://www.prosper.com/.

[19] Snap: Network datasets: Social circles: Facebook, March 2022. https://snap.stanford.edu/data/ego-Facebook.html.

[20] Zopa: First peer-to-peer lending platform in the uk, March 2022. https://www.zopa.com/.

[21] Coinmarketcap, March 2023. https://coinmarketcap.com/.

[22] H. Abdi. Z-scores. *Encyclopedia of measurement and statistics*, 3:1055–1058, 2007.

[23] AIG. Impersonation Fraud Claims Scenarios, 2019. https://www.aig.com/content/dam/aig/america-canada/us/documents/business/management-liability/impersonation-fraud-claims-scenarios-brochure.pdf.

[24] L. Akoglu, M. McGlohon, and C. Faloutsos. OddBall: Spotting anomalies in weighted graphs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6119 LNAI(PART 2):410–421, 2010.

[25] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.

[26] H. Amintoosi and S. S. Kanhere. A trust framework for social participatory sensing systems. In *MobiQuitous*, pages 237–249, 2012.

[27] H. Amintoosi and S. S. Kanhere. Providing trustworthy contributions via a reputation framework in social participatory sensing systems. *CoRR*, abs/1311.2349, 2013.

[28] H. Amintoosi and S. S. Kanhere. A trust-based recruitment framework for multi-hop social participatory sensing. In *Proceedings of the 2013 IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS '13, pages 266–273, Washington, DC, USA, 2013. IEEE Computer Society.

[29] H. Amintoosi and S. S. Kanhere. A reputation framework for social participatory sensing systems. *MONET*, 19(1):88–100, 2014.

[30] E. Anceaume, G. Guette, P. Lajoie-Mazenc, N. Prigent, and V. V. T. Tong. A privacy preserving distributed reputation mechanism. In *2013 IEEE International Conference on Communications (ICC)*, pages 1951–1956. IEEE, 2013.

[31] E. Anceaume, G. Guette, P. Lajoie-Mazenc, T. Sirvent, and V. Viet Triem Tong. Extending signatures of reputation. *Privacy and Identity Management for Emerging Services and Technologies, IFIP Advances in Information and Communication*, 421:165–176, 2014.

[32] E. Androulaki, S. G. Choi, S. M. Bellovin, and T. Malkin. Reputation systems for anonymous networks. In *Proceedings of the 8th Privacy Enhancing Technologies Symposium (PETS 2008)*, 2008.

[33] M. Anwar and J. Greer. Reputation management in privacy-enhanced e-learning. In *Proceedings of the 3rd Annual Scientific Conference of the LORNET Research Network (I2LOR-06)*, Montreal, Canada, November 2006.

[34] M. Anwar and J. Greer. Enabling reputation-based trust in privacy-enhanced learning systems. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, Montreal, Canada, 2008.

[35] A. Aral, R. B. Uriarte, A. Simonet-Boulogne, and I. Brandic. Reliability management for blockchain-based decentralized multi-cloud. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 21–30, 2020.

[36] M. Asher and C. Brennan. Zero-knowledge proofs: Starks vs snarks, May 2021. https://consensys.net/blog/blockchain-explained/zero-knowledge-proofs-starks-vs-snarks/.

[37] A. B. Ayed. A conceptual secure blockchain-based electronic voting system. 9(3):1–9, 2017.

[38] M. A. Azad, S. Bag, and F. Hao. M2m-rep: Reputation of machines in the internet of things. In *Proceedings of the 12th international conference on availability, reliability and security*, pages 1–7, 2017.

[39] M. A. Azad, S. Bag, and F. Hao. Privbox: Verifiable decentralized reputation system for online marketplaces. *Future Generation Computer Systems*, 89:44–57, 2018.

[40] M. A. Azad, S. Bag, F. Hao, and A. Shalaginov. Decentralized self-enforcing trust management system for social internet of things. *IEEE Internet of Things Journal*, 7(4):2690–2703, 2020.

[41] B. Baesens, V. Van Vlasselaer, and W. Verbeke. *Fraud analytics using descriptive, predictive, and social network techniques a guide to data science for fraud detection.* John Wiley & Sons, 2015.

[42] S. Bag, M. A. Azad, and F. Hao. A privacy-aware decentralized and personalized reputation system. *Computers & Security*, 77:514–530, 2018.

[43] A. Bakas, A. Michalas, and A. Ullah. (f) unctional sifting: A privacy-preserving reputation system through multi-input functional encryption. In *Secure IT Systems: 25th Nordic Conference, NordSec 2020, Virtual Event, November 23–24, 2020, Proceedings 25*, pages 111–126. Springer, 2021.

[44] Ballotpedia. Voter turnout in united states elections. https://ballotpedia.org/Voter_turnout_in_United_States_elections, 2022.

[45] R. Bazin, A. Schaub, O. Hasan, and L. Brunie. Self-reported verifiable reputation with rater privacy. In *IFIP International Conference on Trust Management*, pages 180–195. Springer, 2017.

[46] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *Cryptology ePrint Archive*, 2018.

[47] J. C. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret (Extended Abstract). *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 263 LNCS:251–260, 1987.

[48] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

[49] J. Bethencourt, E. Shi, and D. Song. Signatures of reputation: Towards trust without identity. In *Proceedings of the Fourteenth International Conference on Financial Cryptography and Data Security (FC '10)*, pages 400 – 407, 2010.

[50] M. Bettinger. Workflow manager github repository. https://github.com/mbettinger/workflow-manager. Accessed: 2021-02-16.

[51] M. Bettinger and L. Barbero. Source code repository: Collusion-resistant worker set selection. https://github.com/mbettinger/collusion-resistant-worker-set-selection. Accessed: 2021-02-16.

[52] G. Bigwood and T. Henderson. Ironman: Using social networks to add incentives and reputation to opportunistic networks. 2011.

[53] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 326–349, 2012.

[54] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, Oct 2008.

[55] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[56] Y. Bo, Z. Min, and L. Guohuan. A reputation system with privacy and incentive. In *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'07)*, 2007.

[57] R. J. Bolton, D. J. Hand, and Others. Unsupervised profiling methods for fraud detection. *Credit scoring and credit control VII*, pages 235–255, 2001.

[58] R. J. Bolton, D. J. Hand, F. Provost, L. Breiman, R. J. Bolton, and D. J. Hand. Statistical Fraud Detection: A Review. *Statistical Science*, 17(3):235–255, 2002.

[59] A. Boutet, S. Ben Mokhtar, and V. Primault. Uniqueness Assessment of Human Mobility on Multi-Sensor Datasets. Research report, LIRIS UMR CNRS 5205, Oct. 2016.

[60] D. D. S. Braga, M. Niemann, B. Hellingrath, and F. B. D. L. Neto. Survey on computational trust and reputation models. *ACM Computing Surveys (CSUR)*, 51(5):1–40, 2018.

[61] P. L. Brockett, X. Xia, and R. A. Derrig. Using Kohonen's Self-Organizing Feature Map to Uncover Automobile Bodily Injury Claims Fraud. *The Journal of Risk and Insurance*, 65(2):245, 2006.

[62] J. A. Bullinaria. Self Organizing Maps: Fundamentals, Introduction to Neural Networks : Lecture 16. pages 1–15, 2004.

[63] U. C. Bureau. US elections voter turnout statistics. https://www.census.gov/library/stories/2019/04/behind-2018-united-states-midterm-election-turnout.html, 2019.

[64] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *Workshop on World-Sensor-Web (WSW 06): Mobile Device Centric Sensor Networks and Applications*, pages 117–134, 2006.

[65] S. Bursuc, C.-c. Dragan, S. Kremer, S. Bursuc, C.-c. Dragan, S. Kremer, S. Bursuc, I. N.-g. Est, S. Kremer, and I. N.-g. Est. HAL Id : hal-02099434 Private votes on untrusted platforms : models , attacks and provable scheme. 2019.

[66] N. Busom, R. Petrlic, F. Sebé, C. Sorge, and M. Valls. A privacy-preserving reputation system with user rewards. *Journal of Network and Computer Applications*, 80:58–66, 2017.

[67] L. Buttyan, L. Dora, M. Felegyhazi, and I. Vajda. Barter-based cooperation in delay-tolerant personal wireless networks. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pages 1–6, 2007.

[68] L. Buttyan, L. Dora, M. Felegyhazi, and I. Vajda. Barter trade improves message delivery in opportunistic networks. *Ad Hoc Networks*, 8(1):1–14, 2010.

[69] L. Buttyán and J.-P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '00, pages 87–96, Piscataway, NJ, USA, 2000. IEEE Press.

[70] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[71] R. Canillas, O. Hasan, L. Sarrat, and L. Brunie. Graphsif: analyzing flow of payments in a business-to-business network to detect supplier impersonation. *Applied Network Science*, 5(1):1–31, 2020.

[72] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, pages 606–620, 2007.

[73] S.-H. Chang, Y.-S. Chen, and S.-M. Cheng. Detection of sybil attacks in participatory sensing using cloud based trust management system. In *Wireless and Pervasive Computing (ISWPC), 2013 International Symposium on*, pages 1–6, Nov 2013.

[74] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. 28(70), 1985.

[75] B. B. Chen and M. C. Chan. Mobicent: a credit-based incentive system for disruption tolerant network. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

[76] L. Chen, Q. Li, K. M. Martin, and S.-L. Ng. Private reputation retrieval in public–a privacy-aware announcement scheme for vanets. *IET Information Security*, 11(4):204–210, 2016.

[77] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017.

[78] D. Christin. Privacy in mobile participatory sensing: Current trends and future challenges. *Journal of Systems and Software*, 2015.

[79] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, 84(11):1928–1946, 2011.

[80] D. Christin, C. Rosskopf, M. Hollick, L. Martucci, and S. Kanhere. Incognisense: An anonymity-preserving reputation framework for participatory sensing applications. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 135–143, March 2012.

[81] D. Christin, C. Roßkopf, M. Hollick, L. A. Martucci, and S. S. Kanhere. Incognisense: An anonymity-preserving reputation framework for participatory sensing applications. *Pervasive and mobile Computing*, 9(3):353–371, 2013.

[82] M. Chuah and P. Yang. Impact of selective dropping attacks on network coding performance in dtns and a potential mitigation scheme. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pages 1–6, 2009.

[83] M. R. Clark, K. Stewart, and K. M. Hopkinson. Dynamic, privacy-preserving decentralized reputation systems. *IEEE Transactions on Mobile Computing*, 16(9):2506–2517, 2016.

[84] M. R. Clarkson, A. C. Myers, M. R. Clarkson, and A. C. Myers. Civitas : Toward a Secure Voting System Civitas : Toward a Secure Voting System. 7875, 2008.

[85] J. Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Proceedings on Advances in Cryptology—CRYPTO '86*, pages 251–260, Berlin, Heidelberg, 1987. Springer-Verlag.

[86] M. Concha-Barrientos, D. Campbell-Lendrum, K. Steenland, A. Prüss-Üstün, C. Corvalán, and A. Woodward. Occupational noise. *Assessing the*, 7, 2004.

[87] V. Cortier, P. Gaudry, S. Glondu, V. Cortier, P. Gaudry, and S. Glondu. Belenios : a simple private and verifiable electronic voting system To cite this version : HAL Id : hal-02066930. 2019.

[88] I. Damgård, C. Ganesh, H. Khoshakhlagh, C. Orlandi, and L. Siniscalchi. Balancing privacy and accountability in blockchain identity management. In *Topics in Cryptology–CT-RSA 2021: Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17–20, 2021, Proceedings*, pages 552–576, 2021.

[89] H. Dang and H. Wu. Clustering and cluster-based routing protocol for delay-tolerant mobile networks. *Wireless Communications, IEEE Transactions on*, 9(6):1874–1881, 2010.

[90] M. Darame and P. Roger. French legislative elections: Abstention remains the largest 'party' in france. https://www.lemonde.fr/en/politics/article/2022/06/21/french-legislative-elections-abstention-remains-the-first-party-in-france_5987501_5.html, 2022.

[91] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3:1376, 2013.

[92] M. Deutsch. Cooperation and trust: Some theoretical notes. In *Jones, M. R. (ed), Nebraska Symposium on Motivation*. Nebraska University Press, 1962.

[93] E.-H. DFCG. Barometre Euler Hermes-DFCG 2019. https: //www.eulerhermes.fr/ actualites/ etude-fraude-2019.html, 2019.

[94] C. Diaz, E. Kosta, H. Dekeyser, M. Kohlweiss, and G. Nigusse. Privacy preserving electronic petitions. (2008):203–219, 2009.

[95] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[96] T. Dimitriou. Decentralized reputation. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pages 119–130, 2021.

[97] T. Dimitriou and A. Michalas. Multi-party trust computation in decentralized environments. In *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2012.

[98] T. Dimitriou and A. Michalas. Multi-party trust computation in decentralized environments in the presence of malicious adversaries. *Ad Hoc Networks*, 15:53–66, 2014.

[99] R. Dingledine, N. Mathewson, and P. Syverson. Reputation in p2p anonymity systems. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, 2003.

[100] G. Dini and A. Lo Duca. A reputation-based approach to tolerate misbehaving carriers in delay tolerant networks. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 772–777, 2010.

[101] S. Dolev, N. Gilboa, and M. Kopeetsky. Computing multi-party trust privately: in o (n) time units sending one (possibly large) message at a time. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1460–1465, 2010.

[102] S. Dolev, N. Gilboa, and M. Kopeetsky. Efficient private multi-party computations of trust in the presence of curious and malicious users. *Journal of Trust Management*, 1(1):8, 2014.

[103] Y. Dou, H. C. Chan, and M. H. Au. A distributed trust evaluation protocol with privacy protection for intercloud. *IEEE Transactions on Parallel and Distributed Systems*, 30(6):1208–1221, 2018.

[104] A. Dua, N. Bulusu, W.-C. Feng, and W. Hu. Towards trustworthy participatory sensing. In *Proceedings of the 4th USENIX Conference on Hot Topics in Security*, HotSec'09, pages 8–8, Berkeley, CA, USA, 2009. USENIX Association.

[105] A. Dua, W. Hu, and N. Bulusu. Demo abstract: A trusted platform based framework for participatory sensing. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IPSN '09, pages 419–420, Washington, DC, USA, 2009. IEEE Computer Society.

[106] R. Dubos. *Social capital: Theory and research.* 2017. Routledge.

[107] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[108] R. Falcone and C. Castelfranchi. *Social Trust: A Cognitive Approach. In Trust and Deception in Virtual Societies.* Kluwer Academic Publishers, 2001.

[109] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. of the conf. on applications, technologies, architectures, and protocols for computer communications*, pages 27–34, 2003.

[110] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, 1986.

[111] J. Field. Social capital in policy and practice. In *Social Capital*, pages 84–99. 2016. Routledge.

[112] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[113] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone. Blockchain-based Database to Ensure Data Integrity in Cloud Computing Environments. 2015.

[114] N. Gal-Oz, N. Gilboa, and E. Gudes. Schemes for privately computing trust and reputation. In *IFIP International Conference on Trust Management*, pages 1–16. Springer, 2010.

[115] N. Gal-Oz, E. Gudes, and D. Hendler. A robust and knot-aware trust-based reputation model. In *Proceedings of the Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM 2008)*, 2008.

[116] D. Gambetta. *Trust: Making and Breaking Cooperative Relatioins*, chapter Can We Trust Trust?, pages 213 – 237. Department of Sociology, University of Oxford, 2000.

[117] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92. Springer, 2013.

[118] P. Gilbert, L. P. Cox, J. Jung, and D. Wetherall. Toward trustworthy mobile sensing. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, HotMobile '10, pages 31–36, New York, NY, USA, 2010. ACM.

[119] P. Gilbert, J. Jung, K. Lee, H. Qin, D. Sharkey, A. Sheth, and L. P. Cox. Youprove: authenticity and fidelity in mobile sensing. In *SenSys*, pages 176–189, 2011.

[120] S. Gisdakis, T. Giannetsos, and P. Papadimitratos. Sppear: security & privacy-preserving architecture for participatory-sensing applications. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*, pages 39–50. ACM, 2014.

[121] O. Goldreich. *The Foundations of Cryptography - Volume 2.* Cambridge University Press, 2004.

[122] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[123] P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[124] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. of ACM CCS*, pages 89–98, 2006.

[125] J. Graber. Web3 is self-certifying, December 2021. https://jaygraber.medium.com/web3-is-self-certifying-9dad77fd8d81.

[126] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 4, 2001.

[127] M. Green and I. Miers. Bolt: Anonymous payment channels for decentralized currencies. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 473–489, 2017.

[128] E. Gudes, N. Gal-Oz, and A. Grubshtein. Methods for computing trust and reputation while preserving privacy. In *Proceedings of the 23rd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2009.

[129] L. Guo, Y. Fang, and L. Wei. Fine-grained privacy-preserving reputation system for online social networks. In *2013 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 230–235. IEEE, 2013.

[130] S. Gurajala, J. S. White, B. Hudson, B. R. Voter, and J. N. Matthews. Profile characteristics of fake twitter accounts. *Big Data & Society*, 3(2):1–13, 2016.

[131] J. Hartmann and O. Hasan. A social-capital based approach to blockchain-enabled peer-to-peer lending. In *The Third IEEE International Conference on Blockchain Computing and Applications (BCCA 2021)*, pages 105–110, 2021.

[132] M. A. Hasan and M. J. Zaki. *A Survey of Link Prediction in Social Networks.* Springer US, Boston, MA, 2011.

[133] O. Hasan. *Privacy preserving reputation systems for decentralized environments.* PhD thesis, Lyon, INSA, 2010.

[134] O. Hasan, L. Brunie, and E. Bertino. Preserving privacy of feedback providers in decentralized reputation systems. *Computers & Security*, 31(7):816 – 826, October 2012.

[135] O. Hasan, L. Brunie, and E. Bertino. Privacy preserving reputation systems based on blockchain and other cryptographic building blocks: A survey. Technical report, University of Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, 2020. (HAL Id: hal-03034994).

[136] O. Hasan, L. Brunie, and E. Bertino. Privacy-preserving reputation systems based on blockchain and other cryptographic building blocks: A survey. *ACM Computing Surveys*, 55(2):1–37, 2023.

[137] O. Hasan, L. Brunie, E. Bertino, and N. Shang. A decentralized privacy preserving reputation protocol for the malicious adversarial model. Technical Report RR-LIRIS-2012-008, LIRIS, France, http://liris.cnrs.fr/Documents/Liris-5609.pdf, 2012.

[138] O. Hasan, L. Brunie, E. Bertino, and N. Shang. A decentralized privacy preserving reputation protocol for the malicious adversarial model. Technical Report RR-LIRIS-2012-008, University of Lyon, INSA-Lyon, LIRIS, France, June 2012.

[139] O. Hasan, L. Brunie, E. Bertino, and N. Shang. A decentralized privacy preserving reputation protocol for the malicious adversarial model. *IEEE Transactions on Information Forensics and Security*, 8(6):949–962, 2013.

[140] O. Hasan, J. Miao, S. Ben Mokhtar, and L. Brunie. A Privacy Preserving Prediction-based Routing Protocol for Mobile Delay Tolerant Networks. Technical Report RR-LIRIS-2012-011, LIRIS UMR 5205 CNRS, INSA de Lyon, 2012.

[141] O. Hasan, J. Miao, S. Ben Mokhtar, and L. Brunie. A privacy preserving prediction-based routing protocol for mobile delay tolerant networks. In *Proc. of IEEE AINA*, pages 546–553, 2013.

[142] T. Haugum, B. Hoff, M. Alsadi, and J. Li. Security and privacy challenges in blockchain interoperability-a multivocal literature review. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, pages 347–356, 2022.

[143] G. Hayes. The beginners guide to using an ethereum test network.

[144] F. Hendrikx, K. Bubendorfer, and R. Chard. Reputation systems: A survey and taxonomy. *Journal of Parallel and Distributed Computing*, 75:184–197, 2015.

[145] E. Ho. Why you should think twice before trusting airbnb reviews, May 2015.

[146] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys*, 41(4), December 2009.

[147] K. L. Huang, S. S. Kanhere, and W. Hu. Are you contributing trustworthy data?: The case for a reputation system in participatory sensing. In *Proceedings of the 13th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, MSWIM '10, pages 14–22, New York, NY, USA, 2010. ACM.

[148] K. L. Huang, S. S. Kanhere, and W. Hu. A privacy-preserving reputation system for participatory sensing. In *Proceedings of the 2012 IEEE 37th Conference on Local Computer Networks (LCN 2012)*, LCN '12, pages 10–18, Washington, DC, USA, 2012. IEEE Computer Society.

[149] K. L. Huang, S. S. Kanhere, and W. Hu. On the need for a reputation system in mobile phone based sensing. *Ad Hoc Networks*, 12:130–149, 2014.

[150] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *Proc. of ACM SIGCOMM workshop on Delay-tolerant networking*, WDTN '05, pages 244–251, New York, NY, USA, 2005. ACM.

[151] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. 10(11):1576–1589, 2011.

[152] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *Proc. of ACM/IEEE MobiArch*, pages 7:1–7:8, 2007.

[153] IBM. What are smart contracts on blockchain?

[154] T. Ishioka. Extended K-means with an efficient estimation of the number of clusters. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1983:17–22, 2000.

[155] A. K. Jain. Data clustering: 50 years beyond K-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[156] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. *SIGCOMM Comput. Commun. Rev.*, 34:145–158, 2004.

[157] R. Jansen and R. Beverly. Toward anonymity in dtns: Threshold pivot scheme. In *Proc. of MILCOM*, pages 587–592, 2010.

[158] W. Jiang, F. Li, D. Lin, and E. Bertino. No one can track you: Randomized authentication in vehicular ad-hoc networks. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 197–206. IEEE, 2017.

[159] H. J. Jo and W. Choi. Bprf: Blockchain-based privacy-preserving reputation framework for participatory sensing systems. *Plos one*, 14(12):e0225688, 2019.

[160] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618 – 644, March 2007.

[161] R. R. Kalidindi, K. V. S. V. N. Raju, V. V. Kumari, and C. S. Reddy. Trust based participant driven privacy control in participatory sensing. *CoRR*, abs/1103.4727, 2011.

[162] S. D. Kamvar, M. T. Schlosser, and H. GarciaMolina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web (WWW 2003)*, Budapest, Hungary, May 2003.

[163] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet of Things Journal*, 6(3):4660–4670, 2018.

[164] A. Kapadia, D. Kotz, and N. Triandopoulos. Opportunistic sensing: Security challenges for the new paradigm. In *Proceedings of the First International Conference on COMmunication Systems And NETworks*, COMSNETS'09, pages 127–136, Piscataway, NJ, USA, 2009. IEEE Press.

[165] M. Karaliopoulos. Assessing the vulnerability of dtn data relaying schemes to node selfishness. *IEEE Communications Letters*, 13(12):923–925, 2009.

[166] A. Kate, G. Zaverucha, and U. Hengartner. Anonymity and security in dtns. In *Proc. of SecureComm*, pages 504–513, 2007.

[167] A. Keränen, T. Kärkkäinen, and J. Ott. Simulating mobility and dtns with the one. *Journal of Communications*, 5(2):92–105, 2010.

[168] A. Keranen, M. Pitkanen, M. Vuori, and J. Ott. Effect of non-cooperative nodes in mobile dtns. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–7, 2011.

[169] F. Kerschbaum. A verifiable, centralized, coercion-free reputation system. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society (WPES'09)*. ACM, New York, NY, USA, 2009.

[170] W. Khan, Y. Xiang, M. Aalsalem, and Q. Arshad. Mobile phone sensing systems: A survey. *Communications Surveys Tutorials, IEEE*, 15(1):402–427, First 2013.

[171] M. Kim, D. Kotz, and S. Kim. Extracting a mobility model from real user traces. In *Proc. of INFOCOM 2006*, pages 1–13, 2006.

[172] Y. Kim and S. Y. Sohn. Stock fraud detection using peer group analysis. *Expert Systems with Applications*, 39(10):8986–8992, 2012.

[173] M. Kinateder and S. Pearson. A privacy-enhanced peer-to-peer reputation system. In *Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies*, 2003.

[174] T. Kohonen. Self-learning musical grammar, or 'associative memory of the second kind'. pages 1–5, 1989.

[175] G. Korpal and D. Scott. Decentralization and web3 technologies. 2022.

[176] T. Kosch, C. Adler, S. Eichler, C. Schroth, and M. Strassberger. The scalability problem of vehicular ad hoc networks and how to solve it. *IEEE Wireless Communications*, 13(5):22–28, 2006.

[177] G. Kreitz, M. Dam, and D. Wikstrom. Practical private information aggregation in large networks. In *Proceedings of Nordsec*, 2010.

[178] P. Lajoie-Mazenc, E. Anceaume, G. Guette, T. Sirvent, and V. V. T. Tong. Efficient distributed privacy-preserving reputation mechanism handling non-monotonic ratings. *hal.archives-ouvertes.fr*, 2015.

[179] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[180] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, Sept. 2010.

[181] Z. Le, G. Vakde, and M. Wright. Peon: privacy-enhanced opportunistic networks with applications in assistive environments. In *Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '09, pages 76:1–76:8, New York, NY, USA, 2009. ACM.

[182] K. Lee, J. James, and H. Kim. Electronic Voting Service Using Block-Chain. 11(2), 2016.

[183] N. Li and S. K. Das. Radon: reputation-assisted data forwarding in opportunistic networks. In *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, MobiOpp '10, pages 8–14, New York, NY, USA, 2010. ACM.

[184] Q. Li, W. Gao, S. Zhu, and G. Cao. A routing protocol for socially selfish delay tolerant networks. *Ad Hoc Networks*, 10(8):1619–1632, 2012.

[185] Y. Li, P. Hui, D. Jin, L. Su, and L. Zeng. Evaluating the impact of social selfishness on the epidemic routing in delay tolerant networks. *IEEE Communications Letters*, 14(11):1026–1028, 2010.

[186] Y. Li, P. Hui, D. Jin, L. Su, and L. Zeng. Performance evaluation of routing schemes for energy-constrained delay tolerant networks. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5, 2011.

[187] Y. Li, G. Su, and Z. Wang. Evaluating the effects of node cooperation on dtn routing. *AEU - International Journal of Electronics and Communications*, 66(1):62–67, 2012.

[188] Y. Li, G. Su, D. Wu, D. Jin, L. Su, and L. Zeng. The impact of node selfishness on multicasting in delay tolerant networks. *IEEE Transactions on Vehicular Technology*, 60(5):2224–2238, 2011.

[189] Y. Li, W. Susilo, G. Yang, Y. Yu, D. Liu, and M. Guizani. A Blockchain-based Self-tallying Voting Scheme in Decentralized IoT. 2019.

[190] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *The Journal of Privacy and Confidentiality*, 1(1):59–98, 2009.

[191] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):19–20, 2003.

[192] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen. Anonymous reputation system for iiot-enabled retail marketing atop pos blockchain. *IEEE Transactions on Industrial Informatics*, 15(6):3527–3537, 2019.

[193] J. Liu and V. Issarny. Enhanced reputation mechanism for mobile ad hoc networks. In C. Jensen, S. Poslad, and T. Dimitrakos, editors, *Trust Management*, volume 2995 of *Lecture Notes in Computer Science*, pages 48–62. Springer Berlin / Heidelberg, 2004.

[194] M. Liu, Y. Yang, and Z. Qin. A survey of routing protocols and simulations in dtns. In *Proc. of WASA*, pages 243–253, 2011.

[195] R. Lu, X. Lin, and X. Shen. Spring: A social-based privacy-preserving packet forwarding protocol for vehicular delay tolerant networks. In *Proc. INFOCOM 2010*, pages 1–9, 2010.

[196] R. Lu, X. Lin, H. Zhu, X. Shen, and B. Preiss. Pi: A practical incentive protocol for delay tolerant networks. 9(4):1483–1493, 2010.

[197] X. Lu, P. Hui, D. Towsley, J. Pu, and Z. Xiong. Anti-localization anonymous routing for delay tolerant network. *Computer Networks*, 54(11):1899 – 1910, 2010.

[198] Z. Lu, W. Liu, Q. Wang, G. Qu, and Z. Liu. A privacy-preserving trust model based on blockchain for vanets. *IEEE Access*, 6:45655–45664, 2018.

[199] A. Lysyanskaya and J. Camenish. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. *EUROCRYPT 2001*, 2001.

[200] L. Ma, X. Liu, Q. Pei, and Y. Xiang. Privacy-preserving reputation management for edge computing enhanced mobile crowdsensing. *IEEE Transactions on Services Computing*, 12(5):786–799, 2018.

[201] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi. Concurrency and privacy with payment-channel networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 455–471, 2017.

[202] D. Many, M. Burkhart, and X. Dimitropoulos. Fast Private Set Operations with SEPIA. *Technical report, Department of Computer Science, ETH Zurich*, 2012.

[203] A. Manzoor, M. Asplund, M. Bouroche, S. Clarke, and V. Cahill. Trust evaluation for participatory sensing. In *MobiQuitous*, pages 176–187, 2012.

[204] F. G. Mármol and G. M. Pérez. Security threats scenarios in trust and reputation models for distributed systems. *computers & security*, 28(7):545–556, 2009.

[205] S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Mathematics and Computer Science, University of Stirling, Scotland, UK, 1994.

[206] D. H. McKnight and N. L. Chervany. The meanings of trust. Technical Report MISRC Working Paper Series 96-04, University of Minnesota, Management Information Systems Research Center, 1996.

[207] D. H. McKnight, V. Choudhury, and C. Kacmar. Developing and validating trust measures for e-commerce: An integrative typology. *Information Systems Research*, 13(3):334 – 359, September 2002.

[208] A. Mestre. French legislative elections: Sharp decline in voter turnout highlights a worrying trend. https://www.lemonde.fr/en/politics/article/2022/06/19/french-legislative-elections-sharp-decline-in-voter-turnout-highlights-a-worrying-trend_5987291_5.html, 2022.

[209] J. Miao, O. Hasan, S. Ben Mokhtar, and L. Brunie. A self-regulating protocol for efficient routing in mobile delay tolerant networks. In *Digital Ecosystems Technologies (DEST), 2012 6th IEEE International Conference on*, pages 1–6. IEEE, 2012.

[210] J. Miao, O. Hasan, S. Ben Mokhtar, L. Brunie, and K. Yim. An investigation on the unwillingness of nodes to participate in mobile delay tolerant network routing. *International Journal of Information Management*, 33(2):252–262, 2013.

[211] J. Miao, O. Hasan, S. B. Mokhtar, L. Brunie, and G. Gianini. A delay and cost balancing protocol for message routing in mobile delay tolerant networks. *Ad Hoc Networks*, 25(3):430–443, 2015.

[212] A. Michalas and N. Komninos. The lord of the sense: A privacy preserving reputation system for participatory sensing applications. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–6. IEEE, 2014.

[213] N. Miller, P. Resnick, and R. Zeckhauser. Eliciting informative feedback: The peer-prediction method. *Management Science*, 51(9):1359–1373, 2005.

[214] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[215] T. Minkus and K. W. Ross. I know what you're buying: Privacy breaches on ebay. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 164–183. Springer, 2014.

[216] H. Miranda and L. Rodrigues. A framework to provide anonymity in reputation systems. In *Proceedings of the Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, 2006.

[217] M. Mongiovì, P. Bogdanov, R. Ranca, E. E. Papalexakisy, C. Faloutsos, and A. K. Singh. NetSpot: Spotting significant anomalous regions on dynamic networks. *Proceedings of the 2013 SIAM International Conference on Data Mining, SDM 2013*, 2:28–36, 2013.

[218] H. Mousa, S. B. Mokhtar, O. Hasan, L. Brunie, O. Younes, and M. Hadhoud. Privasense: Privacy-preserving and reputation-aware mobile participatory sensing. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 38–47, 2017.

[219] H. Mousa, S. B. Mokhtar, O. Hasan, O. Younes, M. Hadhoud, and L. Brunie. Trust management and reputation systems in mobile participatory sensing applications. *Computer Networks*, 90(C):49–73, Oct. 2015.

[220] H. Mousa, S. B. Mokhtar, O. Hasan, O. Younes, M. Hadhoud, and L. Brunie. A reputation system resilient against malicious and colluding adversaries in particiatory sensing applications. In *Proceeding of CCNC 2017*. IEEE CCNC, 2017.

[221] M. Mulshine. After a disappointing airbnb stay, i realized there's a major flaw in the review system, June 2015.

[222] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, Princeton, NJ, United States, 2016.

[223] J. Nash. Two-person cooperative games. *Econometrica*, 21(1):128–140, 1953.

[224] T. Nguyen Van, T.-D. Le, T. Nguyen-Anh, M.-P. Nguyen, T. Nguyen-Van, M.-Q. Le-Tran, Q. Le, H. Pham, and K. Nguyen-An. A system for scalable decentralized random number generation. 10 2019.

[225] R. Nithyanand and K. Raman. Fuzzy privacy preserving peer-to-peer reputation management. Cryptology ePrint Archive, Report 2009/442, 2009.

[226] M. R. Ogiela and U. Ogiela. Dna-like linguistic secret sharing for strategic information systems. *International Journal of Information Management*, 32(2):175 – 181, 2012.

[227] M. Onen, A. Shikfa, and R. Molva. Optimistic fair exchange for secure forwarding. In *Mobile and Ubiquitous Systems: Networking Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*, pages 1–5, 2007.

[228] C. Onur and A. Yurdakul. Electanon: A blockchain-based, anonymous, robust and scalable ranked-choice voting protocol. *arXiv preprint arXiv:2204.00057*, 2022.

[229] E. Owiyo, Y. Wang, E. Asamoah, D. Kamenyi, and I. Obiri. Decentralized privacy preserving reputation system. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 665–672. IEEE, 2018.

[230] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, 1999.

[231] A. Panagakis, A. Vaios, and I. Stavrakakis. On the effects of cooperation in dtns. In *Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on*, pages 1–6, 2007.

[232] E. Papapetrou, V. F. Bourgos, and A. G. Voyiatzis. Privacy-preserving routing in delay tolerant networks based on bloom filters. In *Proc. of IEEE WoWMoM*, pages 1–9, 2015.

[233] H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

[234] I. Parris and T. Henderson. Privacy-enhanced social-network routing. *Computer Communications*, 35(1):62–74, 2012.

[235] E. Pavlov, J. S. Rosenschein, and Z. Topol. Supporting privacy in decentralized additive reputation systems. In *Proc. of the 2nd Intl. Conf. on Trust Management (iTrust 2004)*, 2004.

[236] E. Pavlov, J. S. Rosenschein, and Z. Topol. Supporting privacy in decentralized additive reputation systems. In C. Jensen, S. Poslad, and T. Dimitrakos, editors, *Trust Management*, pages 108–119, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[237] R. Petrlic, S. Lutters, and C. Sorge. Privacy-preserving reputation management. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 1712–1718, 2014.

[238] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.org/epfl/mobility/20090224, Feb. 2009.

[239] C. E. Priebe, J. M. Conroy, D. J. Marchette, and Y. Park. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, 11(3):229–247, 2005.

[240] J. Pujol, A. Toledo, and P. Rodriguez. Fair routing in delay tolerant networks. In *INFOCOM 2009, IEEE*, pages 837–845. IEEE, 2009.

[241] Q. Qian and M. Xin. *Hidden Markov Model for System Call Anomaly Detection.* 2007.

[242] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *Proceedings of the 9th International Conference on Information Processing in Sensor Networks, IPSN 2010, April 12-16, 2010, Stockholm, Sweden*, pages 105–116, 2010.

[243] D. K. Rappe. Homomorphic cryptosystems and their applications. Cryptology ePrint Archive, Paper 2006/001, 2006. https://eprint.iacr.org/2006/001.

[244] S. Reddy, D. Estrin, and M. Srivastava. Recruitment framework for participatory sensing data collections. In *Proceedings of the 8th International Conference on Pervasive Computing*, Pervasive'10, pages 138–155, Berlin, Heidelberg, 2010. Springer-Verlag.

[245] S. Reddy, K. Shilton, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Evaluating participation and performance in participatory sensing. *UrbanSense08*, page 1, 2008.

[246] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. *IEEE Selected Areas in Communications*, 16(4):482–494, 1998.

[247] P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. *The Economics of the Internet and E-Commerce. Michael R. Baye, editor. Volume 11 of Advances in Applied Microeconomics*, pages 127–157, 2002.

[248] F. Restuccia and S. K. Das. Fides: A trust-based framework for secure user incentivization in participatory sensing. In *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*, pages 1–10. IEEE, 2014.

[249] R. Riemann and S. Grumbach. Distributed Protocols at the Rescue for Trustworthy Online Voting. 2015.

[250] S. Ries, M. Fischlin, L. A. Martucci, and M. Muuhlhauser. Learning whom to trust in a privacy-friendly way. In *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 214–225. IEEE, 2011.

[251] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[252] E. Royer and C. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55, 1999.

[253] G. Sadowksi and P. Rathle. Fraud Detection: Discovering Connections with Graph Databases The #1 Database for Connected Data Fraud Detection: Discovering Connections Using Graph Databases. (January), 2017.

[254] S. Saroiu and A. Wolman. I am a sensor, and i approve this message. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, HotMobile '10, pages 37–42, New York, NY, USA, 2010. ACM.

[255] A. Schaub, R. Bazin, O. Hasan, and L. Brunie. A trustless privacy-preserving reputation system. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 398–411. Springer, 2016.

[256] M. Schiedermeier, O. Hasan, L. Brunie, T. Mayer, and H. Kosch. A transparent referendum protocol with immutable proceedings and verifiable outcome for trustless networks. In *International Conference on Complex Networks and Their Applications*, pages 647–658. Springer, 2019.

[257] S. Schiffner, S. Clauß, and S. Steinbrecher. Privacy and liveliness for reputation systems. In *Proceedings of the Sixth European Workshop on Public Key Infrastructures, Services and Applications (EuroPKI'09)*, pages 209 – 224, 2009.

[258] S. Schiffner, S. Clauß, and S. Steinbrecher. Privacy, liveliness and fairness for reputation. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 506–519. Springer, 2011.

[259] K. A. Ścigała, C. Schild, and I. Zettler. Dishonesty as a signal of trustworthiness: Honesty-humility and trustworthy dishonesty. *Royal Society Open Science*, 7(10), 2020.

[260] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[261] A. Shamir. Identity-based cryptosystems and signature schemes. In G. Blakley and D. Chaum, editors, *Proc. of Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin / Heidelberg, 1985.

[262] R. Sheikh and D. K. Mishra. Protocols for getting maximum value for multi-party computations. In *Proc. of the 4th Asia Intl. Conf. on Mathematical/Analytical Modelling and Computer Simulation*, 2010.

[263] U. Shevade, H. H. Song, L. Qiu, and Y. Zhang. Incentive-aware routing in dtns. In *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, pages 238–247, 2008.

[264] C. Shi, X. Luo, P. Traynor, M. H. Ammar, and E. W. Zegura. Arden: Anonymous networking in delay tolerant networks. *Ad Hoc Networks*, 10(6):918–930, 2012.

[265] J.-G. Song, S.-J. Moon, and J.-W. Jang. A scalable implementation of anonymous voting over ethereum blockchain. *Sensors*, 21(12):3958, 2021.

[266] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. Macalpine, and J. A. Halderman. Security Analysis of the Estonian Internet Voting System. (May):703–715, 2014.

[267] T. Spyropoulos, K. Psounis, and C. Raghavendra. Performance analysis of mobility-assisted routing. In *Proc. of the 7th ACM Intl. Symp. on Mobile ad hoc networking and computing*, pages 49–60, 2006.

[268] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proc. of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, 2005.

[269] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Efficient routing in intermittently connected mobile networks: The multiple-copy case. 16(1):77–90, 2008.

[270] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Efficient routing in intermittently connected mobile networks: The single-copy case. 16(1):63–76, 2008.

[271] T. Spyropoulos, T. Turletti, and K. Obraczka. Routing in delay-tolerant networks comprising heterogeneous node populations. *IEEE Transaction on Mobile Computing*, 8(8):1132–1147, 2009.

[272] M. v. Steen and A. S. Tanenbaum. *Distributed Systems, Third edition, Version 3.03.* Previously published by Pearson Education, Inc., 2020.

[273] S. Steinbrecher. Design options for privacy-respecting reputation systems within centralised internet communities. In *Security and Privacy in Dynamic Environments*, 2006.

[274] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160, 2001.

[275] C. Stoll, L. Klaaßen, and U. Gallersdörfer. The carbon footprint of bitcoin. *Joule*, 3(7):1647–1661, 2019.

[276] G. V. Trunk. A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(3):306–307, July 1979.

[277] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the facebook social graph. 2011.

[278] F. Uribe, I. Hernandez, L. Jung, and P. Amenewolde. Anonymous voting in daos. 2022.

[279] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical report, Citeseer, 2000.

[280] V. Van Vlasselaer, T. Eliassi-Rad, L. Akoglu, M. Snoeck, and B. Baesens. GOTCHA! Network-Based Fraud Detection for Social Security Fraud. *Management Science*, 63(9):3090–3110, 2016.

[281] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004.*, pages 350–360, 2004.

[282] M. Voss, A. Heinemann, and M. Muhlhauser. A privacy preserving reputation system for mobile information dissemination networks. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 171–181, 2005.

[283] X. O. Wang, W. Cheng, P. Mohapatra, and T. Abdelzaher. Enabling reputation and trust in privacy-preserving mobile sensing. *IEEE Transactions on Mobile Computing*, 99(PrePrints):1, 2014.

[284] X. O. Wang, W. Cheng, P. Mohapatra, and T. F. Abdelzaher. Artsense: Anonymous reputation and trust in participatory sensing. In *INFOCOM*, pages 2517–2525. IEEE, 2013.

[285] L. Wei, H. Zhu, Z. Cao, and X. Shen. Mobiid: A user-centric and social-aware reputation based incentive scheme for delay/disruption tolerant networks. In H. Frey, X. Li, and S. Ruehrup, editors, *Ad-hoc, Mobile, and Wireless Networks*, volume 6811 of *Lecture Notes in Computer Science*, pages 177–190. Springer Berlin / Heidelberg, 2011.

[286] O. Williams. Ibm's watson ai saves woman's life by diagnosing rare form of leukaemia, March 2017. https://www.huffingtonpost.co.uk/entry/ibms-watson-ai-saves-womans-life-after-diagnosing-rare-form-of-leukaemia_uk_57a849ade4b04ca9b5d381ef.

[287] G. Wood. Ethereum : A secure, decentralised generalised transaction ledger, 2014. http://diyhpl.us/~bryan/papers2/bitcoin/Ethereum:%20a%20secure%20decentralised%20generalised%20transaction%20ledger.pdf.

[288] X. Xie, H. Chen, and H. Wu. Bargain-based stimulation mechanism for selfish mobile nodes in participatory sensing network. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, pages 1–9, 2009.

[289] Q. Yuan, I. Cardei, and J. Wu. An efficient prediction-based routing in disruption-tolerant networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(1):19–31, 2012.

[290] E. Zaghloul, T. Li, and J. Ren. d-bame: distributed blockchain-based anonymous mobile electronic voting. *IEEE Internet of Things Journal*, 8(22):16585–16597, 2021.

[291] S. Zakhary and M. Radenkovic. Utilizing social links for location privacy in opportunistic delay-tolerant networks. In *Proc. of IEEE ICC*, pages 1059–1063, 2012.

[292] K. Zhang, Z. Li, and Y. Yang. A reputation system preserving the privacy of feedback providers and resisting sybil attacks. *International Journal of Multimedia and Ubiquitous Engineering*, 9(2):141–152, 2014.

[293] W. Zhang, Y. Luo, S. Fu, and T. Xie. Privacy-preserving reputation management for blockchain-based mobile crowdsensing. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2020.

[294] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *IEEE Communications Surveys and Tutorials*, 8(1):24–37, 2006.

[295] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan. Secure multi-party computation: theory, practice and applications. *Information Sciences*, 476:357–372, 2019.

[296] K. Zhao, S. Tang, B. Zhao, and Y. Wu. Dynamic and privacy-preserving reputation management for blockchain-based mobile crowdsensing. *IEEE Access*, 7:74694–74710, 2019.

[297] H. Z. Zhiwei Yan and I. You. N-nemo: A comprehensive network mobility solution in proxy mobile ipv6 network. *Journal of Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications*, Vol.1(2/3):52–70, 2010.

[298] S. Zhong, J. Chen, and Y. Yang. Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1987–1997, 2003.

[299] H. Zhu, X. Lin, R. Lu, Y. Fan, and X. Shen. Smart: A secure multilayer credit-based incentive scheme for delay-tolerant networks. 58(8):4628–4639, 2009.

[300] G. Zyskind. Enigma : Decentralized Computation Platform with Guaranteed Privacy. pages 1–14, 2015.

[301] G. Zyskind, O. Nathan, and A. S. Pentland. Decentralizing privacy: Using blockchain to protect personal data. *Proceedings - 2015 IEEE Security and Privacy Workshops, SPW 2015*, pages 180–184, 2015.