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

Maximilian Schiedermeier[1,2], Omar Hasan[2], Tobias Mayer[3], Lionel Brunie[2] and Harald Kosch[4]

[1]Université du Québec à Montréal, Canada.
[2]LIRIS, INSA Lyon, France.
[3]ConnectedCare GmbH, Germany.
[4]University of Passau, Germany.

Contributing authors: schiedermeier.maximilian@uqam.ca; omar.hasan@insa-lyon.fr; tobias.mayer@connectedcare.net; lionel.brunie@insa-lyon.fr; harald.kosch@uni-passau.de;

Abstract

High voter turnout in elections and referendums is desirable to ensure a robust democracy. Secure electronic voting is a vision for the future of elections and referendums. Such a system can counteract factors hindering 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 implementation must support reproducibility, and the choice of individual voters must remain confidential. In this paper, we propose and evaluate a novel 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 technology, e.g., a Blockchain. The persistence and immutability of the protocol communication allow verifiability of the referendum outcome by any participant. Voters therefore do not need to trust third parties. We provide a formal description and conduct a thorough security evaluation of our proposal.

Keywords: E-Voting, Anonymity, Transparency, Distributed Ledger, Blockchain, SMPC, Trustless Networks

1 Introduction

Over the past years, we have witnessed a sharp decline in voter turnout in major elections [1]. For example, in the 2022 French legislative elections, the turnout was only 46.2% in the critical second round of the elections [2]. The voter turnout for the 2018 US midterm election was at 53.4% [3]. 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 as 36.7% and 41.8% respectively [4].

It is a longstanding goal to render the voters' active participation as effortless and convenient as possible to mitigate 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 paper, we focus on referendums, which can be seen as a special instance of elections, with only two voting options. Even though referendums are a simpler case of elections, implementing them correctly is still very challenging [5] [6]. Many parties may have an interest in the 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 procedure ensures individual ballots are unlikable to the voter. However, the logistic effort imposed by 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 for the individual to hold proof of compliance at every step. In this article, we try to address this problem with an electronic referendum scheme that emphasizes transparency, that is to say, full client-sided verification of correctness.

1.1 Contribution

We propose a transparent referendum protocol with immutable proceedings and verifiable outcome. We define this immutability as the impossibility of tampering with the log of participant actions. Although there already exist protocols with similar ambitions, they commonly provide little evidence to the end user that the designated protocol was followed in practice. We suggest a protocol that is based on a creative combination of existing cryptographic tools. To achieve transparency, we also assess the viability of our proposal considering mobile clients and discuss to which extent the protocol can withstand adversary attacks. Our evaluation concentrates on the confidentiality of votes, transparency and immutability of proceedings and a verifiable outcome.

The key idea behind our contribution is to use a distributed ledger, to keep a complete and publicly accessible log of all communication between participants. Blockchain technology fulfills this criterion but so does any distributed ledger technology with integrity protection, e.g. a distributed database with logs and hashsums. 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 of the voting outcome can occur locally. In return, this means participants gain proof of correctness by themselves (instead of having to trust a third party).

We note that this paper is an extended version of our prior conference paper [7] in the proceedings of the *International Conference on Complex Networks and Their Applications*. This paper extends the conference paper by more than 30% and provides a much more in-depth and updated discussion of the topic. An earlier version [8] of this paper has also been made available on the arXiv preprint server. The extensions in the present paper as compared to the conference paper are summarized below.

- Several sections have been extended with new and updated information.
  - The "Introduction" section has been revised and updated. Two new subsections on the "Contribution" and the "Outline" of the article have been added as well.
  - The "Related work" section has been completely rewritten. It now spans two and a half pages instead of previously half a page. It discusses the current state of the art in further detail. The paper now cites 31 relevant references in total instead of 13 previously.
  - The "Adversary model" section has been extended with two new subsections: "Malicious communication" and "Assumptions".
  - The section on "The protocol" now presents a more detailed description of the proposed protocol.
  - The "Conclusion" now includes a new subsection on "Future work".
- A number of new full sections have been added. These include:
  - A new section on the "Building blocks" used for the construction of the protocol, including four subsections describing "Secret sharing scheme", "Distributed ledger technology", "Asymmetric encryption", and "Anonymous credentials".
  - A new section that discusses the "Scalability" aspects of the protocol.
  - A new section on "Enforcing honest behavior in real-world application".
- The existing figures have been renewed for better clarity. Moreover, new figures have been added for more thorough explanation, which include:
  - "Illustration of Shamir's Secret Sharing".
  - "Illustration of $t - n$ threshold system robustness".

1.2 Outline

In this article, we first give a quick overview of related articles that pursue similar objectives. Some of them follow strategies that are very different from our approach. We point out the issues that they pose and how we intend to address them. Next comes a brief presentation of our model, followed by an enumeration of the cryptographic tools we apply within our protocol. Afterwards, we delve into the exact phases and actions that describe our protocol, followed by an evaluation in two parts. The first part discusses how well our initial objectives are met by the proposal. The second 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 investigation.

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 the associated disadvantages.

Benaloh et al. describe in [9] how secret-sharing schemes can be used as SMPC for secret ballot elections. This work unarguably is the cryptographic foundation of 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, i.e., 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.

The work by *Diaz et al.* [10] describes and evaluates an architecture for a privacy-aware electronic petition system. As petitions typically express only two opinions (non-participation meaning approval, participation meaning disapproval of a topic), it can be considered a referendum system. The core element of this approach involves 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 semantics of issued tokens are independent of effectuated voting decisions. This allows us to ensure transparency, which ultimately renders dishonest server behavior detectable.

In [11] and [12], *Zyskind et al.* provide a description of the Ledger-enabled SMPC 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 an exclusive channel for inter-participant communication. Therefore participants do not obtain the same level of communicative transparency as in our solution.

The work of *Cortier et al.* [13] relies on a threshold system that can defend the secrecy of ballots until 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 remained undisclosed.

Bursuc et al. identified similar objectives [14]: 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 is very focused on that specific aspect and provides no mechanisms for other important goals, such as the prevention of ballot dropping.

Very related to our approach is a proposal by *Li et al.* [15], where an IoT-enabled protocol is discussed. 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 of performing integrity checks on the client's side. Clients thus have to rely on external entities for full integrity checks of the blockchain.

In [16], *Lee et al.* describe a blockchain-based voting protocol. In contrast to our proposal, their solution involves a trusted third party for vote filtering. The work proposed in [17] also suggests 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 [18], *Riemann et al.* introduce a taxonomy of further notions for distributed voting protocols.

Song et al. [19] tackle the problem of scalability in anonymous voting implementations on the Ethereum platform. They identify several bottlenecks that impede the 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 a substantial 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 [20].

Zaghloul et al. [21] 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 [22] 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. [23] 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.

3 Our model

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 3.1.1.

3.1.1 Roles

Our protocol considers several 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.

- **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 of 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 for 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.

3.1.2 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 each other 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 key pair. 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.* [10] demonstrated how anonymous credentials (described in Section 4.4) can be issued among eligible voters, using an external credential server. As a proof-of-concept, Diaz et al. use the Belgian e-ID card as an authentication source. Their use case is about issuing anonymous credentials that are used to anonymously sign petitions. The system guarantees that the users' anonymity is preserved for the step of petition signing due to the use of anonymous credentials. Despite this anonymity, the system ensures that the eligibility can be verified before the issuance of the credentials. This approach unlinks a petition signer's true identity from the pseudo-identity that they would use for signing the petition. In the section on related work (Section 2), we discuss the work by Diaz et al. in further detail as well as the differences from our protocol for the referendum phase.

In the context of our protocol, we can assume that there exists a system similar to Diaz et al. outside the protocol to verify the eligibility of the voters and subsequently issue anonymous credentials. This system could be based on the approach of Diaz et al. or any other one that allows eligibility verification yet ensures anonymity for the voting phase. The eligibility verification entity would learn the identity of the user as well as personal attributes such as age, nationality, etc., which are required for validating eligibility. However, it would not be able to link this information with the user's vote. We also note here that the eligibility verification system may be decentralized or indeed centralized. If the initiator is a centralized entity, for example, the election authority of a geographical region, then the eligibility verification system would naturally be centralized. However, the architecture of the eligibility verification system does not impact the properties of the proposed referendum protocol because the eligibility verification phase is assumed to take place before the referendum and outside of the protocol.

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 the submission of new records. Persisted records however can be neither modified nor erased.

3.2.1 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.

3.2.2 Message notation

For every message m placed on the ledger, we indicate the sender and the recipient information via the indexes α and β, i.e. the notation $m_{\alpha\beta}$ represents a message m, which has been placed on the ledger by sender α, and is destined for recipient β. Some messages are *broadcast messages*, destined for all participants. In this case, the notation shows no recipient, β. The α and β variables are placeholders, throughout the paper we gradually replace them with specific sender and recipient information.

Throughout the protocol, we distinguish between four different message types:

- b_α: Initial broadcast message, contains all referendum parameters. $\alpha = id(init)$
- $s_{\alpha\beta}$: A protected voting message (share), destined for a specific worker. $\alpha = id(v_i), v_i \in V$, $\beta = id(w_j), w_j \in W$

- r_α: A worker's broadcast message, contributing to the referendum outcome. $\alpha = id(w_j), w_j \in W$
- c_α: A worker's broadcast message, contributing to the referendum validation. $\alpha = id(w_j), w_j \in W$

The authenticity of message origins is ensured by the author's signature. As the registration of voters' public keys, described in Section 3.1.2, can be realized over the ledger, it is fair to presume a trusted key exchange among participants, prior to the referendum taking place.

3.3 Adversary model

We consider all voters and workers as potential adversaries. In Section 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.

3.3.1 Malicious communication

In terms of message exchange, we consider:

- submission of syntactically incorrect messages, for instance, messages that lack mandatory meta-information such as a cryptographic signature.
- submission of semantically incorrect messages. We notably consider the submission of undefined values (e.g. out of valid numeric range) and incorrect result values (i.e. values not corresponding to a delegated computation outcome). We consider insane values may appear both, in the message payload and header information, such as the sender field.
- submission of messages that by format or content do not correspond to the current protocol phase.
- inactivity where interaction is requested, that is, deliberate non-communication.

Adversaries may deviate from the expected behavior individually or as a collusive group.

3.3.2 Assumptions

We assume that all openly verifiable information in $b_{id(init)}$, is considered correct by each participant. If a user does not accept the parameters, then the user can decide not to participate in the given instance of the referendum. The referendum is susceptible to having low voter turnout, the proposed parameters of $b_{id(init)}$ are assumed unacceptable to the general voting population. Moreover, since those parameters are public and immutable, their validity can be debated in any public forum at any time after their publication.

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. Although our proposal does not explicitly require blockchain technology, one of the characteristics of blockchains is strong integrity protection for persisted data [24]. It is considered practically infeasible for an adversary to overcome the integrity protection, given the unattainable computation power such an attack would require. 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.

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.
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 secretly modify or erase stored proceedings.

4 Building blocks

4.1 Secret sharing scheme

Any secret sharing scheme supporting additive and multiplicative homomorphic operations will serve our protocol. We decided for the SEPIA [25] specification of Shamir's Secret Sharing, due to its good documentation and ease of integration. Shamir's Secret Sharing is an instance of an SMPC scheme. As such, it allows to perform computations without having to reveal the original inputs to individual parties.

4.1.1 $t-n$ 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 thus create an effective mechanism

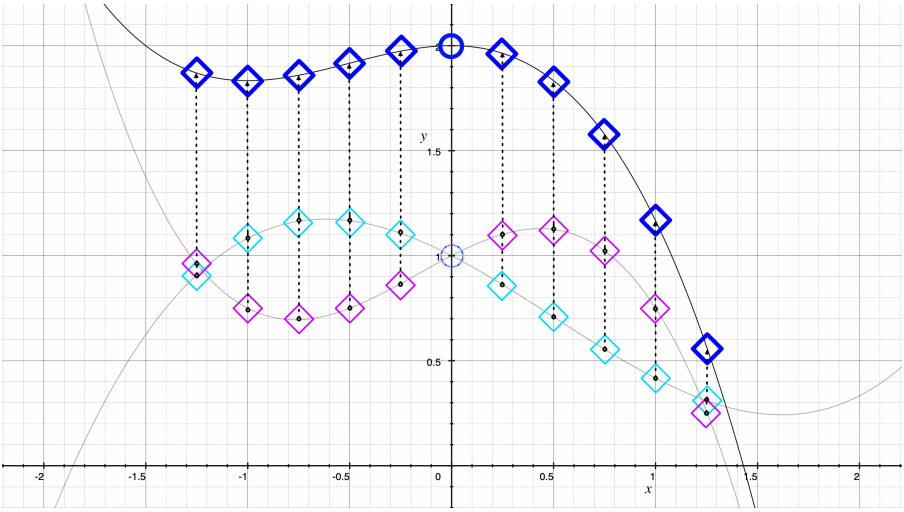

Figure 1 Illustration of $t-n$ threshold system characteristic and support for homomorphic operations in Shamir's Secret Sharing. The sum of two secrets (dashed circles) is computed by first concealing each secret behind samples (turquoise and violet diamonds) of a random polynomial (grey). Adding sample points of the same x-value (dashed upward arrows) produces a new set of result sample points (blue diamonds). The sum of the original secrets (blue circle) is reconstructed by inferring the corresponding polynomial from the result sample points.

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 t, the protection against collusive reconstruction rises. However, in the case of a desired reconstruction, increasing the offset between t and n leads to enhanced robustness, as it makes the reconstruction robust against the unavailability of selected parties. The ratio of t to n can thus be adjusted, to meet a distributed protocol's security and robustness requirements.

4.1.2 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 was first mentioned in [9]. The practical consequences of our protocol are discussed in Section 7.

Figure 1 illustrates how Shamir's Secret Sharing is an SMPC that showcases both traits, the $t-n$ threshold system, and support for homomorphic

operations. Diamonds represent the $t-n$ threshold system. For a given polynomial, as many samples as desired can be constructed (n). However, only t samples are required to reconstruct the polynomial, with t equal the polynomial's degree +1. Computing the sum (blue circle) of two secrets (dashed circles) by passing into polynomial representation, creating samples, summing up samples, and reconstructing the result polynomial is a homomorphism. The summing up of samples can be handled by separate parties, rendering the process an SMPC system.

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.

We note that the proposed referendum protocol is independent of the specific implementation of the underlying distributed ledger. However, it is required that certain essential properties are guaranteed by the distributed ledger. These properties notably include immutability and integrity. For example, the distributed ledger may be implemented by a blockchain, such as Bitcoin that relies on the Proof-of-Work mechanism for consensus, or the ledger may be implemented by Ethereum, that uses the Proof-of-Stake mechanism. Either of these two implementations can ensure immutability and integrity. In addition to immutability and integrity, other practical attributes such as scalability and cost would need to be considered as well for the selection of the underlying distributed ledger. A dedicated distributed ledger may be deployed for the proposed protocol as well as long as the desired properties are guaranteed.

As shown in Figure 2, we consider mobile clients as potential nodes of the distributed ledger as well. These nodes would replicate the distributed ledger, however, in the case of the ledger being a blockchain, they may not have the computing capabilities to participate in the "mining" or "validating" processes of the blockchain. For this reason, a sufficient proportion of "mining" or "validating" nodes must be present as well as required by the underlying distributed ledger to guarantee its overall security. It is the responsibility of the initiator node (described in Section 3.1.1) to verify the general security of the underlying distributed ledger before initiating a referendum. For well-established blockchains, such as Bitcoin and Ethereum, this would generally not be an issue. The fact whether these blockchains are operating securely or not is closely monitored and publicly known. Moreover, voters may also individually verify the security of the underlying distributed ledger before participating and potentially decide to abstain if the security appears to be compromised.

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 key pairs which allow encryption and decryption of directed messages. Furthermore, these keys are used for message signing and authorship validation.

4.4 Anonymous credentials

Anonymous credentials allow a restriction 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 [26]. Those users can later use their credentials to gain admission to an access-controlled service. Though modern implementations [27] respond to advanced requirements such as the detection of double spending or a privacy-aware verification of user-specific attributes, we only make use of the key feature, as it allows the anonymous registration of eligible voters.

Technically, the external entity could issue the cryptographic tokens as hardware devices and to eligible users only. The issuing can include a validation of identity or permissions. Such validation is optional and depends on contextual criticality and the goals of the deploying organization. The issued device could serve as multi-factor authorization (MFA) where also double-usage can be prevented. The actual service use remains anonymous. The entity has to operate honestly due to its critical role. In real-world systems, this is ensured by the legal frame (e.g. data processing contracts, GDPR) and security certifications with regular audits, where violations can cause serious consequences. See also Section 8 for typical measures of real-world application. Note that the external entity is a common and proven practice. An example is certificate authorities (CA) issuing certificates for use in web servers.

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 effectively prevents communications via side channels. This is discussed in more detail in Section 7.

Since our protocol is based on a secret sharing system, it seems at first counter-intuitive that we also make use of a public ledger. Secret sharing systems protect secret information by dividing information into separate shares. The protection lies in the fact that the secret information can only be reconstructed when shares are reunited, i.e. one party gets hold of enough shares. Yet, we suggest to store all shares in a single place, namely, in a public ledger. This seems pointless, for anyone can access the public ledger, and hence combine all shares, immediately breaking all protection. Therefore, we do not store the shares in plain but apply additional asymmetric encryption. The interest of this encryption is to maintain a clear trace of all shares while ensuring

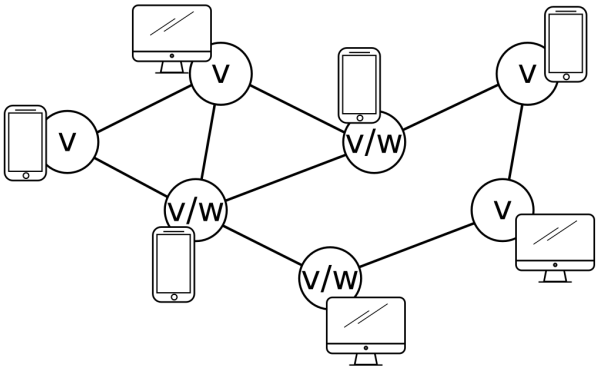

Figure 2 Illustration of referendum participants, if a blockchain is used as ledger. Every referendum participant replicates the ledger. Although the nodes constantly synchronize, referendum-related messages are exchanged exclusively via ledger records. Therefore all clients hold a transparent copy of the proceedings. As pictured above, the protocol does not bind specific roles to particular hardware.

share confidentiality to a specific party. In more general terms: Only the intended target entity (party) has access to a specific set of sensitive information (shares). At the same time, the ledger as an exclusive communication channel allows us to monitor the message meta-data of all participants. Participants only know one another by their anonymous identities, i.e. the ledger is the only means of communication. In reverse, 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 not an issue, as participants only know each other by their anonymous identities.

In case a blockchain is used for the ledger, and mining sets on proof-of-work, we can infer the following restrictions for participant hardware: The verification of the ledger's integrity by itself does not require clients to actively mine, we consider it reasonable to enable mobile clients as blockchain replicating nodes. This is a valid assumption, given two conditions:

1. The used blockchain serves exclusively for the current referendum. By restricting the ledger content to this specific payload, the blockchain's data volume is significantly reduced. This is an essential decision, since popular public chains can easily exceed the storage capacity of a mobile client and therefore render a replication infeasible.
2. As shown in Figure 2, a portion of the participants rely on non-mobile hardware, bearing the resources for active mining. This is likewise an important condition, as the blockchain only provides integrity protection when an honest majority of miners can not be computationally outperformed by adversaries [24].

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 the 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.

5.2 Protocol outline

Figure 3 illustrates how individual roles chronologically submit and retrieve messages to the ledger. For each action, it also indicates the corresponding protocol phase.

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 outcome and run verifications.

The next section provides more details regarding the individual phases.

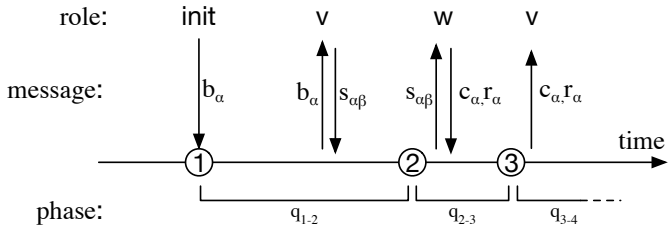

Figure 3 Illustration of *protocol phases*. Downward arrows indicate the persistence of messages *types* into the ledger, upward arrows indicate the lookup of messages (indicated by *type*). Time advances from left to right.

5.2.1 Initiation

The goal of the first phase is to ensure all participants operate on identical referendum parameters. The referendum initiator *init* communicates these parameters 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. The two options are always encoded by numeric values $\in \{-1, +1\}$. The initial broadcast message's payload $\tilde{b}_{id(init)}$ only defines the semantics for each voting option value, for instance:
     Are cats cooler than dogs? Yes $= +1$, *No* $= -1$.
   - A set of time-stamps that define the transitions between subsequent phases $Q = \{q_{1-2}, q_{2-3}, q_{3-4}\}$. The fixed time stamps 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, we ensure a unanimous participant understanding of the upcoming referendum proceedings, i.e. the initial message contains all the information required to outline subsequent participant communications.

5.2.2 Vote submission

In the second phase, voters cast their votes. Start and end of the *Vote Submission* phase are defined through the referendum init message $b_{id(init)}$, as q_{1-2},

respectively q_{2-3}. Throughout the *Vote Submission* phase, 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 $\psi_i \in \{-1, +1\}$. The mapping is specified in $b_{id(init)}$, e.g. if v_i were to believe that cats are indeed cooler than dogs, he would proceed using the numeric value corresponding to Yes: "+1".
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 $b_{id(init)}$.
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 4 illustrate this step.

Voters can submit votes until a timestamp q_{2-3}. During the valid time window, voters can submit multiple votes. Only the last vote is taken into account and messages s_{ij} submitted after timestamp q_{2-3} are considered non-compliant with the protocol and are ignored.

Note that vote changes during the valid time windows are explicitly wanted as a practical feature to allow voters to change their minds and demonstrate protocol flexibility. The exact mechanism for "*Double Voting*" is described in detail in Section 7. It can be easily changed to traditional voting by considering only the first vote and ignoring subsequent ones even within the valid time window. Note also that the timestamp is computed using a block number or storage id on the ledger as timestamps in distributed systems cannot guaranteed to be reliable due to consensus protocols.

5.2.3 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 $\tilde{s}_{1j}, ..., \tilde{s}_{kj}$, each encrypted with his public key $\tilde{w}_j$.
3. w_j decrypts every single share using his private key $\hat{w}_j$ 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:

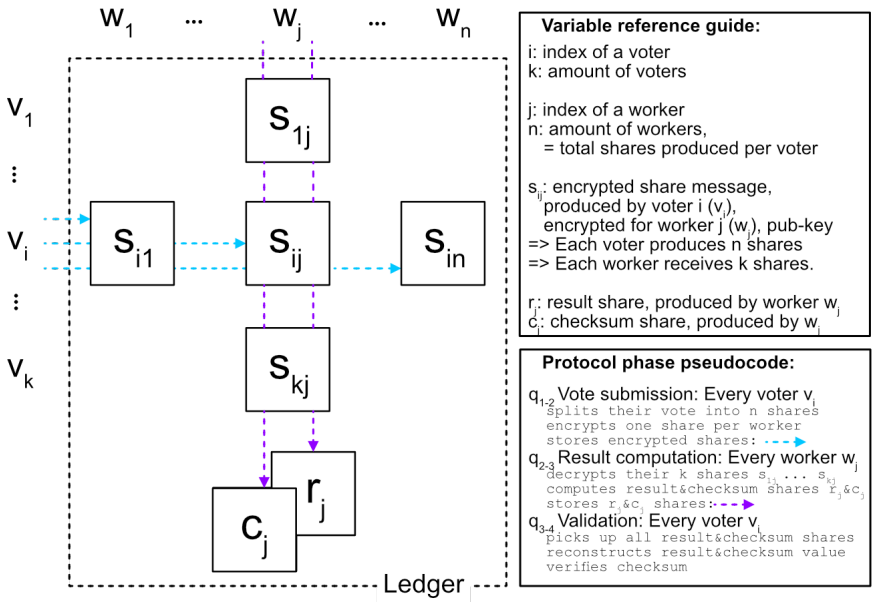

Figure 4 Illustration of *vote submission* by a voter v_i and *intermediate result computation* by a worker w_j. Note that all messages arising throughout these steps are persisted in the ledger.

- 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 persists those in the ledger.

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

5.2.4 Determination and validation of the outcome

In the final phase, voters individually reconstruct the referendum outcome and evaluate the proceedings for conformity with the protocol. To do so, every voter v_i performs the following actions:

1. v_i picks up all result- and checksum messages, deposited in the ledger by the workers: $\{r_j \mid j \in \bar{W}\}$ and $\{c_j \mid j \in \bar{W}\}$.

2. v_i extracts the message payloads. This produces two separate sets. One containing the election result shares and one for the election checksum shares: $\{\tilde{r}_j \mid j \in \bar{W}\}$ and $\{\tilde{c}_j \mid j \in \bar{W}\}$
3. v_i uses each set to reconstruct the corresponding secret, i.e. v_i combines the shares of each set, to remove the protection of the threshold system. That is, v_i 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 7.

6 Analysis of objective fulfillment

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

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 a complete transcript of referendum proceedings. We ensure the ledger's exclusive status as a targeted communication medium by concealing the physical identity of participants behind pseudonyms. Since the ledger ensures the immutability of persisted records, we obtain an immutable log of the referendum proceedings.

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 every vote according to a secret sharing scheme and then encrypting each resulting share asymmetrically. As a reminder: the shares of all votes reside together in the ledger. While the ledger content is public, the shares cannot be readily combined, because each share is encrypted with a specific worker key. Unless an adversary manages to overcome the asymmetric encryption for at least t shares of the same vote (allowing subsequent vote reconstruction), confidentiality of the vote remains ensured. Though asymmetric encryption mechanisms are in theory breakable, such attack is assumed computationally infeasible in

practice. That is to say with current or near-future hardware it is extremely unlikely for an adversary to reconstruct even a single secret share, without the required key material. Furthermore, by analyzing the ledger, voters can reconstruct the message flow among participants and exclude even the possibility that workers colluded to reunite shares. As workers only know one another by their pseudo-identifiers, they can not secretly establish a communication side channel for collusion.

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 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. Figure 5 illustrates the principle on an example. Let's assume 10 sampling points are provided to reconstruct a 3rd-degree polynomial, e.g. $f(x) = \frac{1}{3}x^3 - \frac{1}{2}x^2 + 2$. Ideally, any four sampling points suffice to reconstruct the polynomial. However, additional sampling points provide robustness against incorrect, or in the worst case, intentionally manipulated sampling points. The greater the offset between t and n, the more robust the protection against incoherent samples. In turn, the identification of outliers allows the exclusion of dishonest parties. Figure 5 illustrates this principle, where we assume all but one sampling point is correct. If all samples are correct, any subset of t sampling points will be reconstructed to the same polynomial. In return, if some subsets of size t produce a deviating result, these subsets contain an incorrect sampling point, suggesting one or several dishonest workers.

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.

6.5 Scalability

In addition to the four initial objectives discussed above, we also discuss the scalability aspect of the proposed referendum protocol in this section. Elections

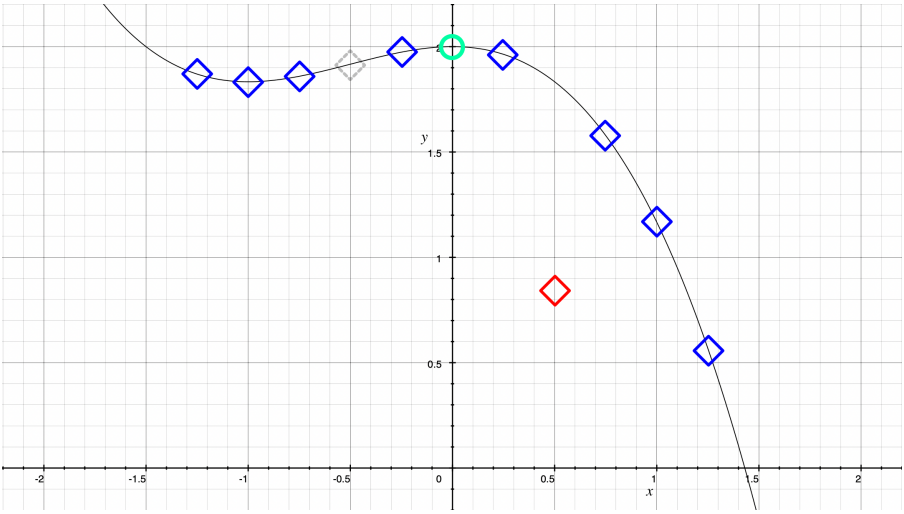

Figure 5 Illustration of $t - n$ *threshold system* robustness. If the amount t of samples needed to reconstruct ($t = 4$ for a third-degree polynomial) is lower than the total number n of total shares ($n = 10$, all diamonds), the system gains in redundancy, i.e. the secret (green circle) can be reconstructed even if shares are missing (grey diamond) or a manipulated value has been provided (red diamond).

and referendums need to take place for democratic decision-making in a variety of institutions and constituencies. These entities could comprise of hundreds, thousands, or even millions of eligible voters.

Scaling the proposed protocol to a few hundred or even a few thousand voters appears feasible. Let's take the *init* phase of the protocol as an example. It requires placing k identities or public keys of the eligible voters on the ledger. The German Federal Office for Information Security (Deutschland Digital Sicher BSI) recommends using a key size of 256 Bits for encryption with an Elliptic Curve based cryptosystem (ECC) [28]. The size of the set of all keys for 10 thousand eligible voters would be 10000×256 Bits = 2560000 Bits = 320 Kilobytes. The feasibility of storing this amount of information on a distributed ledger would depend on its specific implementation. For example, recording 320 KB of information on a blockchain such as Bitcoin or Ethereum could be expensive, however, it would be technically feasible.

Now let's take the example of the country of Germany, which may be considered a medium to large-sized country in terms of population given its ranking of 19 in the list of over 200 countries in the year 2023 [29]. The population of Germany that is eligible to vote is comprised of 60.4 million people according to the German election demographics for the year 2021 [30]. Given the above statistics and considering a single ECC public key of 256 Bits for each voter, we can determine the size of the set of keys of the entire eligible voting population. The size of the set would be 60400000×256 Bits = 15462400000 Bits = 1.80006 Gigabytes. Considering the storage capacities of

modern computing devices, a couple of Gigabytes is a fairly manageable storage requirement. However, recording such a large amount of information on a blockchain such as Bitcoin or Ethereum can be expensive or even prohibitive.

Some alternative solutions could be used to resolve this limitation of scalability. One example would be storing the complete set of keys on a decentralized storage platform such as IPFS instead and only the hash of the set on the blockchain. Other solutions could involve employing better-scaling blockchain technology, such as Layer 2 scaling solutions. Moreover, we note that elections rarely take place on the scale of an entire country. Rather, elections generally take place in smaller constituencies and their individual results are eventually aggregated into a country-wide result.

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. That is, eligible voters can choose not to distribute shares or only send them to a subset of workers. Furthermore, 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 is to systematically ignore all vote shares of voters that do not comply with the expected behavior. This way the disadvantage of inactivity lies entirely with the adversary. Likewise, intentionally inactive workers cannot be prevented, however, the nature of the $t-n$ threshold system provides robustness against that scenario. Precisely, for reconstructing the voting *outcome* only t samples are required, that is the protocol is robust until up to $n-t$ inactive workers. Concerning the referendum outcome's *checksum*, the redundancy is weaker. The reason is that the checksum computation relies on an internal polynomial multiplication (see Section 5.2.4), the effect of which is a doubled degree of the checksum result polynomial. Since reconstruction of a twice-as-high-degree polynomial requires approximately twice as many sampling points, the protocol is less robust against inactive workers, concerning the checksum. The exact tolerance for inactive workers is at $n - t^2$ missing checksum samples. [9]
- **Syntactically incorrect messages**: Participants can violate the protocol by sending syntactically incorrect messages. Syntactic errors can be easily detected with syntax schemes, e.g. if messages are exchanged in JSON format, their syntax can be validated with a corresponding JSON schema instance, defining valid the message meta-structure, and property constraints [31]. 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 signatures 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] Furthermore, it is not a severe threat to the protocol as it requires collusion (which, as previously argued, is difficult to establish, given the concealed identities of protocol participants).
- **Incorrect intermediate results**: Workers might submit incorrect intermediate results on purpose. In the 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). However, massively colluding adversaries could also push an honest minority into an outlier position. Another inconvenience for worker adversaries is that they can not predict the effects of their manipulation. Given the SMPC, an altered value can influence the result in either direction. As discussed previously, we furthermore hinder collusive attacks by concealing the participants' natural identities.
- **Double voting**: Double voting is an explicit protocol feature, i.e. voters are allowed to cast multiple votes. However, this does not increase voting power for individuals. If multiple votes are cast by the same voter, the workers only consider the shares, corresponding to the most recent vote. That is, voters can repeat the generation, encryption and distribution of vote-corresponding shares. The double voting is thereupon detectable (each share message $s_{\alpha,\beta}$ contains an unencrypted, signed header). If workers are confronted with multiple shares from the same voter, they interpret the situation as "*The voter changed their mind*" and therefore only consume the share with the

[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.

most recent timestamp, i.e. the outcome of the SMPC only reflects the latest vote. Throughout the *Vote Submission* phase q_{1-2} (see 5.2.2), a voter can change their mind as many times as they want.

- **De-anonymization**: Participants might be interested in identifying the physical entity that operates behind a participant pseudonym. This would enable outside-ledger undetected communication. As all network traffic runs over TOR connections, 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 and opens a gate for further attacks, it is not trivial to establish. A resilient system can counter this by setting the threshold value reasonably high. Specifically, this means that the probability of random workers falling 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. Adversaries publicly declaring their will to collude can be easily detected.
- **Voter exclusion**: In Section 2, we criticized the usage of an anonymous credential server, because such a server can, if dishonest, omit correctly submitted ballots and thereby exclude votes at will. However, in our case, anonymous credentials are only used for voter registration, not for the voting procedure itself. In [10], a voter cannot provably expose a dishonest petition server. If the server never hands out a confirmation of receipt, a voter cannot prove their previous interaction with the server. In the worst-case scenario blaming a dishonest server may even jeopardize the confidentiality of the vote: If a malicious server reports a unanimous vote outcome, blaming the server is equivalent to disclosing the own vote not being that option. The described risk is eliminated in our case. The voter registration itself can be logged in the ledger, i.e. the public init message $b_{id(init)}$ can be inspected preemptively. If a legit voter were intentionally or accidentally excluded, they can point out the mistake, *before* the voting phase, that is without any risk. Consequently, in our protocol, a legitimate voter can prove their exclusion by a malicious server, without risks.

8 Enforcing honest behavior in real-world application

In classical analogue voting, a person receives a ballot paper after physical authentication of the identity card by specially authorized voting staff. This records that a specific person has received a ballot paper and submitted a vote but keeps this information confidential. The ballot paper and the vote itself are anonymous. The correct behavior of both the voting staff and voters is regulated by a legal framework and supervised by the voting staff.

The presented approach of this paper follows the same principle. An external entity issues the tokens to voters for vote submission and records which

voter has received a token (but not which one) and keeps this information confidential. The token and vote submission remains fully anonymous. Compared to classical analogue voting, even the token issuing of our proposal can be fully anonymous without any form of identity validation. It is easy to imagine that the external entity could be an independent trusted organization that uses a secured channel (e.g. encrypted emails) to provide unique IDs for authorizing token issuing.

Enforcing honest behavior of the external entity (but also the technical infrastructure provider) of our approach in real-world application would still rely on laws and legal agreements as cornerstones. The honest behavior of voters in classical analogue voting is enforced by physical presence and supervision, whereas the technical system ensures correct behavior of the voting procedure. To this end, real-world deployments are hardened by various technical, organizational and operational measures to ensure honest behavior. The following list outlines the most important measures.

- **Legal framework:** Legal contracts specify data processing activities and potential fines for the commercial sector. Also, the data processing has to follow the legal framework of the countries of the residing servers. In Europe, the GDPR defines serious data protection requirements, where violations cause significant consequences. Examples are Meta, which was fined 1.2 billion Euros in May 2023, and Whatsapp which was fined 225 million Euros in 2021 for data processing-related violations.
- **Certified data security:** Organizations typically prove their level of secure data processing with security certifications that imply also regular audits by independent third parties. For example, Amazon Web Services (AWS) owns a multitude of certifications such as ISO 27001, ISO 9001, ISO 22301, ISO 27017, ISO 27701, ISO 27019 as well as SOC 1 / 2 / 3. or PCI DSS and has their infrastructure certified according the Cloud Computing Compliance Controls Catalogue (C5). A full list of certifications and engagements can be found online[3].
- **Architectural design patterns:** Software development should follow development paradigms such as security-by-design, privacy-by-design, privacy-by-default or shift-left-security. Also, data processing should be minimized and server locations should be used according to the legal framework.
- **Cloud security services:** Modern cloud infrastructure providers offer powerful security services. For example, the aforementioned AWS offers DDoS prevention, intrusion detection, security scanners (e.g. operation systems, containers, infrastructure configurations), monitoring and logging, root cause analysis support and compliance validation scanners (e.g. regarding controls of CIS or NIST 800-53 rev 5).
- **Monitoring and logging:** All relevant actions are monitored and administrators are informed about anonymous behavior. Extensive logging is taking place and also stored on a secured dedicated system for possible audits by external parties and thus only accessible by a few trusted persons.

[3] https://aws.amazon.com/de/compliance/programs/

- **Specific measures:** Many specific measures are applied to harden real-world deployments. This includes notably data encryption in transit and at rest, following official recommendations (e.g. BSI TR-2102 for cryptographic algorithms and key lengths), network separation, access control/permissions or firewalls. Dedicated security teams should enforce such measures and be in contact with relevant actors (engineering teams, management, data protection officer etc.)
- **Processes:** Several processes such as vulnerability management, incident management and risk management are crucial for data security. Thus, they are at the core of information security management systems (ISMS) and related certifications (e.g. ISO 27001). Also, some processes become legal requirements in Europe through regulations such as NIS-2 or the Cyber Resilience Act.

9 Conclusion

9.1 Objectives fulfillment

Our work demonstrates how the challenges of electronic referendums can be answered with a creative combination of existing approaches. By bringing together the potential of ledger technology and secure multiparty computation, we constructed a highly transparent referendum protocol that allows participants to autonomously verify proceedings and referendum outcome. Traditional $t - n$ threshold-based systems gain security exclusively by selection of parameters that render successful collusive attacks unlikely. To the best of our knowledge, there is no other system that further enforces security, by considering proofs that inspect communication meta-data, protected by ledger technology. This concept generates trust on the client side, because except for 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.

9.2 Future work

In future research, we would like to further investigate a meaningful selection of referendum parameters. We could also imagine experimenting with machine learning approaches, to reach an optimal trade-off between security and scaling. Machine learning approaches could also be used to identify outliers and filter outliers in provided sampling points. Also, we would 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.

Declarations

Funding

Open Access funding enabled and organized by Projekt DEAL. We acknowledge support by the Open Access Publication Fund of University Library Passau.

Informed consent

Not applicable.

Ethical statement

The authors declare no conflict of interest.

Author contribution

Maximilian Schiedermeier is the primary contributor. Omar Hasan and Tobias Mayer contributed to all aspects. Lionel Brunie and Harald Kosch contributed to the conceptualization and analysis.

Data availability

Not applicable.

References

[1] Mestre, A.: 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-tren 5987291_5.html (2022)

[2] Darame, M., Roger, P.: 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)

[3] Bureau, U.S.C.: US elections voter turnout statistics. https://www.census.gov/library/stories/2019/04/behind-2018-united-states-midterm-election-turnout.html (2019)

[4] Ballotpedia: Voter turnout in United States elections. https://ballotpedia.org/Voter_turnout_in_United_States_elections (2022)

[5] Clarkson, M.R., Myers, A.C., Clarkson, M.R., Myers, A.C.: Civitas : Toward a Secure Voting System Civitas : Toward a Secure Voting System **7875** (2008)

[6] Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., Macalpine, M., Halderman, J.A.: Security Analysis of the Estonian Internet Voting System (May), 703–715 (2014)

[7] Schiedermeier, M., Hasan, O., Brunie, L., Mayer, T., Kosch, H.: A transparent referendum protocol with immutable proceedings and verifiable outcome for trustless networks. In: International Conference on Complex Networks and Their Applications, pp. 647–658 (2019). Springer

[8] Schiedermeier, M., Hasan, O., Mayer, T., Brunie, L., Kosch, H.: A transparent referendum protocol with immutable proceedings and verifiable outcome for trustless networks. arXiv preprint arXiv:1909.06462 (2019)

[9] Benaloh, J.C.: 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)

[10] Diaz, C., Kosta, E., Dekeyser, H., Kohlweiss, M., Nigusse, G.: Privacy preserving electronic petitions (2008), 203–219 (2009)

[11] Zyskind, G.: Enigma : Decentralized Computation Platform with Guaranteed Privacy, 1–14 (2015) arXiv:arXiv:1506.03471v1

[12] Zyskind, G., Nathan, O., Pentland, A.S.: Decentralizing privacy: Using blockchain to protect personal data. Proceedings - 2015 IEEE Security and Privacy Workshops, SPW 2015, 180–184 (2015)

[13] Cortier, V., Gaudry, P., Glondu, S., Cortier, V., Gaudry, P., Glondu, S.: Belenios : a simple private and verifiable electronic voting system To cite this version : HAL Id : hal-02066930 (2019)

[14] Bursuc, S., Dragan, C.-c., Kremer, S., Bursuc, S., Dragan, C.-c., Kremer, S., Bursuc, S., Est, I.N.-g., Kremer, S., Est, I.N.-g.: HAL Id : hal-02099434 Private votes on untrusted platforms : models , attacks and provable scheme (2019)

[15] Li, Y., Susilo, W., Yang, G., Yu, Y., Liu, D., Guizani, M.: A Blockchain-based Self-tallying Voting Scheme in Decentralized IoT (2019) arXiv:arXiv:1902.03710v1

[16] Lee, K., James, J.I., Kim, H.J., James, J.I.: Electronic Voting Service Using Block-Chain **11**(2) (2016)

[17] Ayed, A.B.: A conceptual secure blockchain-based electronic voting system **9**(3), 1–9 (2017)

[18] Riemann, R., Grumbach, S.: Distributed Protocols at the Rescue for Trustworthy Online Voting (2015) arXiv:arXiv:1705.04480v1

[19] Song, J.-G., Moon, S.-J., Jang, J.-W.: A scalable implementation of anonymous voting over ethereum blockchain. Sensors **21**(12), 3958 (2021)

[20] Hao, F., Ryan, P.Y., Zieliński, P.: Anonymous voting by two-round public discussion. IET Information Security **4**(2), 62–67 (2010)

[21] Zaghloul, E., Li, T., Ren, J.: d-bame: distributed blockchain-based anonymous mobile electronic voting. IEEE Internet of Things Journal **8**(22), 16585–16597 (2021)

[22] Onur, C., Yurdakul, A.: Electanon: A blockchain-based, anonymous, robust and scalable ranked-choice voting protocol. arXiv preprint arXiv:2204.00057 (2022)

[23] Uribe, F., Hernandez, I., Jung, L., Amenewolde, P.: Anonymous voting in dao's (2022)

[24] Gaetani, E., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., Sassone, V.: Blockchain-based Database to Ensure Data Integrity in Cloud Computing Environments (2015)

[25] Many, D., Burkhart, M., Dimitropoulos, X.: Fast Private Set Operations with SEPIA. Technical report, Department of Computer Science, ETH Zurich (2012)

[26] Chaum, D.: Security without identification: Transaction systems to make big brother obsolete **28**(70) (1985)

[27] Lysyanskaya, A., Camenish, J.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. EUROCRYPT 2001 (2001)

[28] BSI: BSI TR-02102 Cryptographic Mechanisms. https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr02102/tr02102_node.html (2024)

[29] United-Nations: United Nations Data Portal Population Division. https://population.un.org (2024)

[30] Bateson, I., Thurau, J.: German election demographics. https://www.dw.com/en/german-election-demographics-facts-and-figures/a-59143207

(2021)

[31] Pezoa, F., Reutter, J.L., Suarez, F., Ugarte, M., Vrgoč, D.: Foundations of json schema. In: Proceedings of the 25th International Conference on World Wide Web, pp. 263–273 (2016)