A transparent referendum protocol with immutable proceedings and verifiable outcome for trustless networks

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

[1] McGill University (DISL), 3480 University, MC322, H3A0E9 Montréal, QC, Canada
maximilian.schiedermeier@mail.mcgill.ca
[2] INSA Lyon (LIRIS), 20 Albert Einstein, Blaise Pascal, 69621 Villeurbanne, France
omar.hasan@insa-lyon.fr, lionel.brunie@insa-lyon.fr
[3] Verimi GmbH, Oranienstraße 91, 10969 Berlin, Germany
tobias.mayer@verimi.com
[4] Universität Passau, Innstraße 43, ITZ/IH, 94032 Passau, Germany
harald.kosch@uni-passau.de

Abstract. High voter turnout in elections and referendums is very 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 paper, 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. We provide a formal description and conduct a thorough security evaluation of our proposal.

Keywords: E-Voting, Trustless Networks, Political Networks, Transparency, Unlinkability, Blockchain

1 Introduction

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. It is an ongoing objective to render the voters' active participation as effortless and convenient as possible in order to promote a high 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 options offered for vote. Even though referendums are a simpler case of elections, implementing them correctly is still very challenging [5] [11]. 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 rely on for their confidentiality. 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 article we address this problem with an electronic-referendum scheme that puts emphasis on transparency that is to say, full client-sided verification of correctness.

2 Related Work

In [7], the authors suggest and evaluate an architecture for a privacy-aware electronic petition system. As petitions express only two opinions (non-participation meaning approval, participation meaning disapproval), it can be considered a referendum system. The core element in this approach is involving anonymous certificates to 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, a dishonest petition server could discard signatures. The outcome would be indistinguishable from a case where the voter has never 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 voting decisions. This allows us to ensure transparency, which ultimately renders dishonest server behavior detectable.
Further related work is discussed in our extended technical report [10], where we compare our approach to a formal SMPC model for secret ballot elections [2], a general Ledger-enabled SMPC platform [13] [12], pure $t - n$-threshold based voting systems [6], metric-based evaluations for compromised voting systems [4] and multiple blockchain based voting systems [9] [8] [1]. We there also discuss how each of these works proposes to electronic voting, but unlike our approach does not provide full client-side verifiability of protocol-proceedings, to locally

exclude the possibility of privacy breaches. Local and autonomous verifications are a prerequisite for a trustless network context, as it eliminates the need for trusted third parties.

3 Our Model

3.1 Participants

We distinguish between physical entities, identifiers and roles. Each physical entity possesses a unique and anonymous identifier.

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.
- **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: $W = \{w_1, ..., w_n\}$, $W \subset V$.

Identities When we talk of *participants* P, we implicitly mean the physical entities behind voters and workers.[5] Participants only know by an anonymous pseudo-identifier $\bar{p}$. Likewise we introduce the set of all pseudo-identifiers as $\bar{P}$. For illustration purposes, we denote a mapping function $id : P \to \bar{P}$ that translates from an entity $p \in P$ to the associated identifier $\bar{p} \in \bar{P}$. 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*.

3.2 Ledger

A key component of our model is an immutable and integrity-protected datastore 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 can be neither modified nor erased. We use the ledger as the *exclusive* communication medium among participants. The exchange messages by adding and polling ledger records whenever they communicate.

[5] With the definition $P = V \cup W$, P is equal to V. However, by using P instead of V, we highlight that we are not only interested in voter behavior.

3.3 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.

3.4 Adversary Model

We consider all voters and workers as potential adversaries. In section 4.2, we outline the *expected* behavior of referendum participants. Our adversary model covers that any Voter or Worker may deviate from this expected behavior at any time. 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.

3.5 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 referendum initialization until the determination of the result. Proceedings must arise implicitly due execution of the described actions. Persisted proceedings must be immutable.

4 The Protocol

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 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. 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.

4.1 Protocol Overview

Our referendum protocol is based on a secure multi-party computation (SMPC) 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 correctness 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 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.

4.2 Protocol Outline

This section provides details regarding the individual protocol phases. The overall sequence is shown in Figure 1, it illustrates how individual roles chronologically submit and retrieve messages to the ledger. The graphic also highlights corresponding protocol 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:

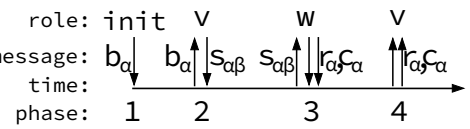

Fig. 1. Illustration of *protocol phases.* Downward arrows indicate the persistence of messages *types* into the ledger, upward arrows indicate the lookup of messages (likewise indicated by *type*). Time advances from left to right.

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: $\bar{V} = \{id(v_i)|v_i \in V\}$.
   - A subset of identities that names the designated workers: $\bar{W} = \{id(w_j)|w_j \in W\}$, 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 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, 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 $b_{id(init)}$.
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 2 illustrate this step.

Voters can perform the above steps until timestamp q_{2-3} is reached. Repeated submissions before the deadline are considered an update. 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 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 2 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 $\{\tilde{r}_j | j \in \bar{W}\}$ and $\{\tilde{c}_j | j \in \bar{W}\}$:

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

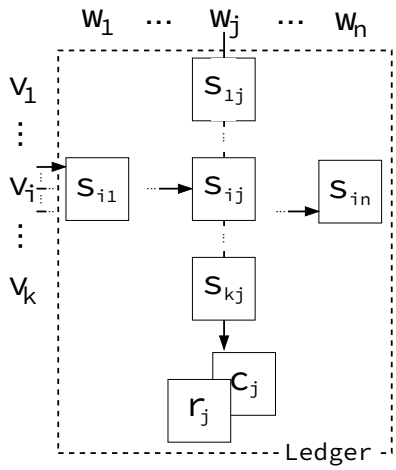

Fig. 2. Illustration of *vote submission* by a voter v_i and *intermediate result computation* by a worker w_j. Note that all messages arising throughout these steps are persisted in the ledger.

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.[6]

5 Analysis of objective fulfillment

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

5.1 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. We ensure the ledgers exclusive status as targeted communication medium by concealing the physical identity of participants behind pseudonyms. Since the blockchain ensures the immutability of persisted records, we obtain an immutable log of the referendum proceedings.

[6] Still, it is possible to generate a valid checksum with cleverly arranged illegal input values. We discuss this threat in section 6.

5.2 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. 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 another by their pseudo-identifiers, they can not secretly establish a communication side channel for collusion.

5.3 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, 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. Assuming that intermediate results are verifiable, the worker-provided checksum allows a privacy aware input validation.

5.4 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 Security analysis

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

- **Intentional inactivity**: A participant can perform intentional inactivity where interaction is expected. Voters can choose not to distribute shares or only send them to a subset of workers. A worker can decide not to submit intermediate result shares. Although vote-messages are encrypted, all participants can inspect the ledger content and detect inactive voters or voter who do not communicate with all designated workers. The default strategy is to ignore all vote-shares of voters that do not comply to the expected behavior. This way the disadvantage of inactivity lies with the adversary. 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 at most of $n-t^2$ inactive workers. (See [2])
- **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.
- **Impersonation**: Participants may send messages in the name of another participant. Impersonated messages are detectable, their signature does not match the expected author. Messages with invalid signature are 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.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*: $\sum_{k=1}^{n} |x_k y_k| \leq (\sum_{k=1}^{n} |x_k|^p)^{1/p} (\sum_{k=1}^{n} |y_k|^q)^{1/q}$, with $n \in \mathbb{N}, \{x_1, ..., x_n\}, \{y_1, ..., y_n\} \in \mathbb{R}, p, q \in [1, \infty)$.[7]
- **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 5). However, massively colluding adversaries could also push an honest minority into an outlier position. Another inconvenience for worker

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

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. We furthermore hinder collusive attacks by concealing the participants' natural identities.

- **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. Though secret inter-participant communication is a severe threat to the protocol's transparency and opens a gate for further attacks, it is hard to establish. A resilient system can counter this by setting the threshold-value reasonably high. 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. Adversaries publicly declaring their will to collude can be easily detected.
- **Voter exclusion**: In our proposal anonymous credentials are only used for registration, not for voting. In [7], 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, for they can be logged in the public init message, $b_{id(init)}$. A legitimate voter could easily prove his exclusion by a malicious server.

7 Conclusion

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 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 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 at client side, because with exception to the anonymous credential issuer a need for trusted third parties is eliminated. We provided a realistic adversary model and analyzed how our protocol withstands corresponding attacks.

References

1. Ayed, A.B.: A conceptual secure blockchain-based electronic voting system 9(3), 1–9 (2017)
2. 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)
3. Bureau, U.C.: US elections voter turnout statistics. https://www.census.gov/library/stories/2019/04/behind-2018-united-states-midterm-election-turnout.html (2019)
4. 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)
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. 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)
7. Diaz, C., Kosta, E., Dekeyser, H., Kohlweiss, M., Nigusse, G.: Privacy preserving electronic petitions (2008), 203–219 (2009)
8. Lee, K., James, J., Kim, H.: Electronic Voting Service Using Block-Chain 11(2) (2016)
9. Li, Y., Susilo, W., Yang, G., Yu, Y., Liu, D., Guizani, M.: A Blockchain-based Self-tallying Voting Scheme in Decentralized IoT (2019)
10. Schiedermeier, M., Hasan, O., Mayer, T., Brunie, L., Kosch, H.: Technical report: A transparent referendum protocol with immutable proceedings and verifiable outcome for trustless networks pp. 1–14, 3 figures (2019), https://arxiv.org/pdf/1909.06462.pdf
11. 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)
12. Zyskind, G.: Enigma : Decentralized Computation Platform with Guaranteed Privacy pp. 1–14 (2015)
13. Zyskind, G., Nathan, O., Pentland, A.S.: Decentralizing privacy: Using blockchain to protect personal data. Proceedings - 2015 IEEE Security and Privacy Workshops, SPW 2015 pp. 180–184 (2015)