Voting without Self-Voting

Peter B. Rønne[0000−0002−2785−8301]

SnT & University of Luxembourg, Esch-sur-Alzette, Luxembourg
peter.roenne@uni.lu

Abstract. In an election, self-voting, i.e. candidates voting for themselves or their own proposals, might only capture an obvious inclination or a fear of loss of reputation, and hence may not be useful towards choosing the best candidate. In some contexts, e.g. for small scale boardroom elections, it can thus be sensible to prohibit self-voting, especially, this will prevent everybody pointing to themselves as the best choice. In the case of public elections this is easy to enforce, however, in standard secret ballot elections the no-self-voting condition is unchecked and relies on the honesty of the participants. More commonly, the constraint is simply not imposed in the first place due to lack of enforcement. A generalisation is where certain groups are not allowed to vote for their own candidate. In this case, preventing self-voting can also reduce the level of coercion, e.g., if team leaders demand, or more subtly simply expect, all their team members to vote for them in an election for the best team leader.
With the aid of secure e-voting, imposing the no-self-voting constraint becomes possible. We show how this constraint can be implemented efficiently, in the context of both centralised and decentralised voting. Especially, we show how to obtain a robust (i.e. allowing absentees) decentralised voting system preventing self-voting by using just standard linkable ring signatures and anonymous vote-casting channels.

1 Introduction

In elections, it is often the case that candidates are themselves part of the electorate. Sometimes *self-voting* is excluded in order to obtain an expressive result and avoiding everybody pointing to themselves as the best candidate. In general, we can envisage that for some voters, certain voting options are excluded. In the Eurovision Song Contest, as an example, countries are not allowed to vote for themselves. A more illustrative example is the Security Protocols Workshop where the best presentation is elected in a secret ballot but presenters are not allowed to vote for their own talk. Interestingly, prohibiting PhD students to vote for the talk of their supervisor, or vice versa, also helps to prevent implicit coercion threats, and hence to select a better winner.

Ideally, voting systems should fulfill at least two basic properties: 1) Verifiability: Allowing the detection of any vote fraud, including casting more or different votes than allowed, modifying or deleting the votes of other participants, and finally ensuring a correct tally of the votes; 2) Ballot Privacy: Protecting

the privacy of the cast vote, besides the unavoidable privacy leakage from the actual result of the election. The challenge is to balance the levels of verifiability and privacy with good usability and efficiency. Disallowing self-voting adds yet another constraint that we need to be able to verify, and if done poorly can lead to privacy leaks.

In this paper, we show how to efficiently enforce the no-self-voting condition in different settings. For voting systems with central tally authorities, we show how to efficiently prove the correctness of cast ballots using a voter-side plaintext (in)equivalency test/proof. For these proofs, we carefully avoid pitfalls both in terms of soundness (see also discussion in [15]) and privacy (which does not seem to have been discussed earlier).

We also consider protocols in the decentralised setting (aka boardroom voting), i.e., in the case where we do not have or do not want to rely on a central authority trusted for privacy.

Decentralised voting systems come in (at least) two flavours. In systems like [10, 9] cryptographic ballots are cast with a direct (authenticated) relation to the voter, and we can apply plaintext (in)equivalence proofs like in the central system case. However, if just a single voter fails to participate, we need to run recovery rounds until all anticipated voters participate (see [8] for a method to gain robustness but paying in terms of privacy and accuracy). The second type of decentralised scheme assumes anonymous channels to cast plaintext votes.[1] The advantage is that the protocol is robust, i.e. tolerates abstaining voters without having to redo the election. In this setting, we present a scheme which is simple, light-weight and fulfills both ballot-privacy, verifiability, and allows detection and removal of self-votes.

To achieve this we alter the known linkable ring signatures (LRS) decentralised voting protocol. We propose a new primitive called Conditional Linkable Ring Signatures to replace the LRS and detect self-votes. Importantly, we manage to achieve this without having to implement new zero-knowledge proofs, and build our scheme using two instantiations of the basic primitive of LRS.

In this short paper, we give possible solutions to prevent self-voting in paper-based voting (Sec. 2.1) and central-authority e-voting (Sec. 2.2) including an outlook towards coercion-resistant voting. We will also point out a privacy pitfall problem with plaintext-equivalence tests which relates to their general use in e-voting and beyond (Sec. 2.3). Finally, we discuss decentralised e-voting in Sec. 2.4, introduce the conditional linkable ring signatures and our decentralised scheme.

2 Proposals for No-Self-Voting Solutions

2.1 Paper ballots

In standard elections using paper ballots, where voters mark their choice among a number of candidates on the ballot, it seems hard to prevent self-voting. How-

[1] Anonymous channels can be hard to implement in practice especially in a strong adversarial model, e.g. a malicious bulletin board seeing network metadata.

ever, in some election systems, each candidate has their own ballot, and the voter makes their choice by casting their chosen candidate-ballot in the ballot box. Such systems have been used historically, but have privacy issues. Despite these, they are still in use, e.g. in Israel, see [1] which also discusses possible privacy attacks. However, in such a system we could hand out a selected range of ballots to each voter individually, preventing self-voting. The main difficulty here consists of verifying that only one vote is cast, and recollecting the non-cast ballots in a privacy-preserving way.
Advantages: Easy to implement for candidate-specific paper ballots.
Disadvantages: Hard to handle many candidates. Possibilities of privacy- and verifiability attacks.

2.2 E-Voting with Central Tally Authority

In the most common setup of e-voting we have a central authority responsible for the tally holding a key pair $(\mathsf{pk}, \mathsf{sk})$ for a public key encryption system. For better privacy, these keys might be distributed over several players which jointly will apply threshold verifiable decryption of the final election result, either after homomorphic aggregation of votes or using a mixnet for privacy. In this case, a voter will normally cast a ballot to the bulletin board as

$$\mathsf{ID}, \mathsf{Enc}(\mathsf{C}; r), \pi$$

where ID is the identity of the voter, C the choice of candidate, which is encrypted with randomness r. Finally, π is a zero-knowledge proof (ZKP) of correct encryption of a valid candidate which is also a proof of knowledge, and normally it ensures NM-CPA security of the ciphertext with the ZKP. To prevent self-voting, we want to prove that $\mathsf{ID} \neq \mathsf{C}$ (for some encoded version of ID).

There are several ways to ensure the absence of self-voting.

- We can adjust the proof π of correctness of C to ensure $\mathsf{C} \neq \mathsf{ID}$. If this is a simple OR proof over candidates, removing a candidate can even make the proof smaller. However, in some cases this is less efficient, e.g. π could be a range proof for the full candidate list which might be twice or more expensive to split; or in some cases there are no constraints on C over the message space, e.g., a write-in. Further, it can be easier for the implementation to have the same proof type π for all voters, and simply add a smaller ZKP for the exclusion of ID on top.
- The are methods for non-membership proofs, e.g., [4] gives efficient constructions but using the Groth-Sahai proof system in a bilinear group setting. Depending on the setting this can be good solution.
- The election authority could do standard Plaintext Equivalence Tests/Proofs (PETs) [15] on $\mathsf{Enc}(\mathsf{C}; r)$ to prove that $\mathsf{C} \neq \mathsf{ID}$. However, this is not desirable. In particular, it will involve the secret election key which we prefer to keep inactive until the decryption of the final tally. Especially, if we want to verify ballots on the fly, this will be troublesome since PETs requires a threshold set of authorities to be available online in order to decrypt.

We here consider ElGamal encryption in a general DDH group, due to its extensive use in voting systems. We show how an encryptor can make an efficient plaintext (in)equivalence zero-knowledge proof using the knowledge of the encryption randomness, using straightforward known and very efficient techniques.[2] For generality, parallel to PETs, we show how to prove both equality and inequality, which can be useful in other settings.

Let $\mathbb{G}$ be a prime-order group with q being the order. Let g be a generator of $\mathbb{G}$ and $\mathsf{pk} = g^{\mathsf{sk}}$ the public election key. An ElGamal encryption of C is $\mathsf{Enc}(\mathsf{C}; r) = (g^r, \mathsf{pk}^r \mathsf{C})$ where we abuse notation and use C both as the choice and the corresponding encoding as a group element. We have suppressed notation for group multiplications.

In our use case to prevent self-voting, we want a protocol to prove that the cast vote $\mathsf{Enc}(\mathsf{C}; r_1)$ does not have the voter themself as chosen candidate. For generality, we present a more general proof of plaintext (in)equality test from a prover who is the encryptor of two ciphertexts.

The protocol for the proof of plaintext (in)equality of $\mathsf{Enc}(\mathsf{C}; r_1) = (g^{r_1}, \mathsf{pk}^{r_1}\mathsf{C}_1)$ and $\mathsf{Enc}(\mathsf{C}; r_2) = (g^{r_2}, \mathsf{pk}^{r_2}\mathsf{C}_2)$ between a prover P knowing r_1, r_2 (the voter) and a verfier V is now as follows[3]

1. P sends $\mathsf{Enc}(\mathsf{C}_1; r_1) = (g^{r_1}, \mathsf{pk}^{r_1}\mathsf{C}_1) := (a_1, b_1)$ and $\mathsf{Enc}(\mathsf{C}_2; r_2) = (g^{r_2}, \mathsf{pk}^{r_2}\mathsf{C}_2) := (a_2, b_2)$
2. P, V use the homomorphic property of ElGamal encryption to compute $(a_1/a_2, b_1/b_2) = \mathsf{Enc}(\mathsf{C}_1/\mathsf{C}_2; r_1 - r_2) := (d, e)$. The problem is now equivalent to showing whether $\mathsf{C}_1/\mathsf{C}_2 \stackrel{?}{=} 1$.
3. (a) In case of in-equivalence, P gives a ZKP $\pi_{\text{non-eq}}$ of discrete log inequivalence which proves that $d = g^r = g^{r_1 - r_2}$ (including knowledge of r), but that $e \neq \mathsf{pk}^r$. This is done without proving knowledge of a discrete log of e (which could be unknown). We give references for this proof below.
   (b) In the case of equivalence, P gives a ZKP π_{eq} of discrete log equivalence which proves that $d = g^r = g^{r_1 - r_2}$ and $e = \mathsf{pk}^r = \mathsf{pk}^{r_1 - r_2}$. This can be done using a standard Chaum-Pedersen proof [7].
4. V verifies the correctness of π_{eq} or $\pi_{\text{non-eq}}$ (unless proof was interactive).

Note that if the ZKP is non-interactive, then this protocol is also non-interactive. Whereas the Chaum-Pedersen proof is very well-known, the discrete log inequivalence proof is less known, but follows from [5, Sec. 5]. Both are very efficient, the size of the equality proof is just two group elements and the in-equality has a size of three group elements [16]. Both can be made non-interactive using the strong Fiat-Shamir transformation [2].

Advantages: We only need to add a very small extra zero-knowledge proof to standard systems without changing other parts of the system. It also works easily if a group of voters has to be prevented from voting for a given candidate.

[2] Ref. [3] gives general constructions of PETs and PETs for encryptors, but due to their generality each proof iteration only has soundness 1/2 and will be less efficient.

[3] For our case, we can set $r_2 = 0$ but the general case is relevant in other contexts, e.g. when submitting ballots anonymously in JCJ [11].

Disadvantages: If we need to exclude more than one candidate, the proof size will be linear in the number of excluded candidates.

2.3 Privacy Problems in Standard PETs

An important question is why did not follow the procedure of standard PETs by simply exponentiating $\mathsf{Enc}(\mathsf{C}_1/\mathsf{C}_2; r_1 - r_2)$ with r (for PETs this is done in a distributed way using several parties [15]) to get

$$(g^{r(r_1-r_2)}, \mathsf{pk}^{r(r_1-r_2)}(\mathsf{C}_1/\mathsf{C}_2)^r)$$

Then revealing $r(r_1 - r_2)$ to the verifier allows retrieving $(\mathsf{C}_1/\mathsf{C}_2)^r$ and checking whether this is 1 for equality or random for inequality. Following [15], the verifier should also check that $r(r_1 - r_2) \neq 0$ for soundness of the proof. However, this can actually result in privacy leaks of the encrypted candidate.

To see this, note that both here and for standard PETs [15], we reveal the terms

$$g^{r_1-r_2}, g^{r(r_1-r_2)}, (\mathsf{C}_1/\mathsf{C}_2)^r \tag{1}$$

which we cannot prove to be random to the adversary without further assumptions. In particular, if an active adversary happens to know $g^{r_1-r_2}$ beforehand when choosing the (encoding of) plaintexts, he could set $\mathsf{C}_1/\mathsf{C}_2 = g^{x(r_1-r_2)}$ for some x known to the adversary. Now, the plaintext equivalence proof between ciphertexts of resp. C_1 and C_2 would fail but would still reveal the choice of plaintexts by comparing the two last terms in Eq. 1, which would be $g^{r(r_1-r_2)}, g^{xr(r_1-r_2)}$ which can be checked by the adversary knowing x. In practice, one could imagine that this could happen if mixnets are prepared in advance for a fast online mixing phase, e.g., in JCJ where this is the bottleneck for the tally efficiency. Even if this attack seems theoretical, the privacy leak is important since it would make a proof of privacy of a scheme using such PETs troublesome.

We leave it as an open problem to avoid this privacy problem in general, but our solution above for the case where the encryptor makes the PET side-steps this problem.

Coercion-Resistant Voting: We finally comment on the possibilities in coercion-resistant voting. Here, the voter's submitted ciphertext should not be linkable to the voter. This makes the problem of avoiding self-voting harder. In JCJ [11], one could imagine encoding candidate choices via the publicly encrypted credentials and then weed out self-votes using PETs. However, we leave it as an open problem to prevent self-voting efficiently while preserving coercion-resistance.

2.4 Decentralised Voting

Decentralised schemes like [10, 9] have ballots which can be extended to ElGamal ciphertexts with a public key. This means, we can use the plaintext in-equivalence

proof from Sec. 2.2 to prevent self-voting. However, these schemes are not robust - all eligible voters need to cast their vote to compute the election result. We now show how to get robust decentralised voting schemes without self-voting, and that this can be achieved without implementing new cryptographic primitives using only linkable signatures.

Linkable Ring Signatures In the seminal paper [13] Linkable Ring Signatures (LRS) were introduced and shown to facilitate the construction of a robust decentralised voting protocol. We describe this approach here in an informal and abbreviated form. An LRS scheme contains four algorithms $\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{Link}$. Each user, U_i, holds a signing and public verification key pair $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{Gen}(1^k)$. Let L be a subset of the public keys, which is called the ring and provides an anonymity set. We need to use an enhanced version of LRS which in addition to the ring also has a label lbl, called *event-id* in [12]. U_i can then sign a message m as $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m, (\mathsf{lbl}, L))$. These signatures are verified using Ver. Finally, the signatures can be pairwise checked if they are linked via Link. Signatures are linked iff they have the same signer, same ring and same label, formally $\mathsf{Link}(L, m, m', \mathsf{Sign}(m, \mathsf{sk}_i, (\mathsf{lbl}, L)), \mathsf{Sign}(m', \mathsf{sk}_j, (\mathsf{lbl}', L'))) = \delta_{i,j}\delta_{L,L'}\delta_{\mathsf{lbl},\mathsf{lbl}'}$.[4]

The security properties are defined via games, see e.g. [14], and informally ensure: 1) **Existential Unforgeability:** Given a set of honestly generated public keys, EU ensures that even in the presence of a signing oracle that produces signatures for different, messages, rings and signers, no new verifiable signature can be constructed. 2) **Signer Anonymity:** Allows the adversary to adaptively corrupt and obtain oracle signatures. Then the adversary gets a challenge signature for a chosen ring and label, and has to guess better than random which of the non-corrupted users created it. 3) **Linkability:** Ensures a) that the adversary cannot frame a user by producing a new signature linkable to a signature from the honest user, other than by copying. b) That users cannot create extra unlinkable signatures for the same ring.

Given an LRS setup, the voting protocol in [13] is simple: Votes are cast in plaintext to a bulletin board BB via an *anonymous channel*, but signed with LRS, where the ring is over the public keys of all voters. The signatures prevent ballot stuffing from non-eligible parties. The signer anonymity of LRS means that the ballots are anonymous in the ring of all voters, and ensures ballot privacy. Finally, the linkability of LRS means that if someone submits multiple ballots these are linkable via the Link algorithm, and ensures one-vote-per-voter verifiability. Since the ballots have plaintext votes, the election is *self-tallying*, also with absentees, and anyone can confirm the result.

Conditional Linkable Ring Signatures Our idea to prevent self-voting is to let voters publish public-verifiable signatures of their prohibited voting options, and change the LRS primitive to allow linking signatures across different rings

[4] δ denotes the Kronecker delta function.

and labels but only if the user signs the same message twice.[5] To be more general, this could also be a condition in terms of the messages, labels and rings of the two signatures – hence we name them Conditional Linkable Ring Signatures (CLRS).

We thus define a second algorithm for linking, denoted Link_R, for a given (symmetric) predicate R such that (here lbl, L, m are implicit in the signature) if $\sigma_i = \mathsf{Sign}(m, \mathsf{sk}_i, (\mathsf{lbl}, L))$ and $\sigma'_j = \mathsf{Sign}(m', \mathsf{sk}_j, (\mathsf{lbl}', L'))$

$$\mathsf{Link}_R(\sigma'_i, \sigma'_j) = \delta_{i,j} R\big((\mathsf{lbl}, L, m), (\mathsf{lbl}', L', m')\big)$$

In our case it is sufficient to consider the predicate defined to be true iff $m = m'$. We denote this by $R = id$.

The security properties of CLRS are straightforward generalisations of LRS, with a new linkability game for Link_R and anonymity needs to be updated since signatures fulfilling the relation are now linkable. When we only link same message signatures, users can ensure that anonymity is preserved by only signing messages including extra unique identifiers.

The idea is now as follows: In an initial round, each voter signs their disallowed voting options where the ring is their own public key. This is sent to BB and is publicly verifiable due to the ring being only the single voter. For the actual election round, all voters cast their plaintext votes with their ring signature, where the ring is now over all voters. If someone casts a self-vote it will be detectable since the plaintext vote is the same in the first and second round, making it linkable. We now make this precise.

The No-Self-Voting Voting Protocol Given a CLRS, the protocol preventing self-voting is a simple two-round protocol. Each of the n voters U_i generates a key pair $\mathsf{sk}_i, \mathsf{pk}_i$ for the chosen CLRS. Let C_k denote the voting options, and assume U_i is not allowed to vote for $\widehat{\mathsf{C}}_{U_i}$. In an initial round each voter U_i signs and publishes the prohibited option $\widehat{\mathsf{C}}_{U_i}$ signed under the ring consisting only of the user's own public key and label lbl_0 (this round is skipped for voters without candidate constraints):

$$U_i,\ \mathsf{Sign}\big(\widehat{\mathsf{C}}_{U_i}, \mathsf{sk}_i, (\mathsf{lbl}_0, \{\mathsf{pk}_i\})\big) \longrightarrow \mathsf{BB}$$

In the actual voting round, each voter U_i anonymously submits their candidate choice C_{U_i} signed under the ring of all voter public keys $L = \{\mathsf{pk}_i\}_{i=1,\ldots,n}$ and a second unique label lbl_1.

$$\mathsf{Sign}\big(\mathsf{C}_{U_i}, \mathsf{sk}_i, (\mathsf{lbl}_1, L)\big) \overset{anom.}{\longrightarrow} \mathsf{BB}$$

Finally, everyone can verify all signatures, check that none of the second round signatures are linked with Link and check that none are linked to the initial

[5] The idea of tracing same message signatures goes back to the untraceable e-cash system [6] where it was used to avoid double spending.

round signatures using Link_{id}. Note that it is clear who signed each message in the initial round since the ring there is just the single public key of the signer.

If we re-use keys for several elections or election rounds then each round needs a unique label lbl_j, and the candidate names in each round are appended with lbl_j to prevent linking between rounds. In the initial round, the voters sign all prohibited choices with the corresponding round labels. Alternatively, a CLRS with a different predicate can be used to ensure that there is no linkability between the voting rounds, but still between the voting round and the initial round for the same messages.

Verifiability: Each voter can check that their ballot appears on BB. Anyone can verify signatures (ensures no ballot stuffing from externals by existential unforgeability) and that they are not linked within the voting round using Link (ensures one-vote-per-voter) and not linkable to the initial round using Link_{id} (ensures no self-voting). Linkable ballots are simply removed.

Ballot-Privacy: Ballot privacy follows from the anonymity of the vote casting channels and the anonymity of the CLRS primitive. For multiple elections the unique labels ensure no linkability between rounds. It would be interesting to develop ballot privacy notions that also capture side-channels, e.g., if a voter always casts a vote fast and can be correlated between rounds.

Instantiating the Primitive Until now we have only considered the CLRS as a primitive. We now demonstrate that when we have a given overall set of public keys L_{all}, we can instantiate CLRS for linking the same messages (or more generally if $f(\mathsf{lbl}, L, m) = f(\mathsf{lbl}', L', m')$ for a function f) by using two LRS signatures. This goes against the spontaneous nature of LRS which allows adding users ad hoc, but in the case of a voting protocol with a known set of public keys this is sufficient. The idea is that we sign as expected using the first signature and let the label of the second signature be the message itself. The message of the second signature is the first signature to bind them together[6] and the ring is L_{all}, i.e. the CLRS signature $\mathsf{Sign}^{CLRS}(\mathsf{sk}_i, m, (\mathsf{lbl}, L))$ consists of the two LRS signatures

$$\sigma_1 = \mathsf{Sign}^{LRS}\big(\mathsf{sk}_i,\, 0||m,\, (0||\mathsf{lbl}, L)\big), \sigma_2 = \mathsf{Sign}^{LRS}\big(\mathsf{sk}_i,\, 1||\sigma_1,\, (1||m, L_{all})\big)$$

When the same message is signed twice, the second part becomes linkable since it has the same label, but otherwise when messages are different it is not linkable and anonymity is preserved since it is over the ring of all public keys. The security of the construction should be proven formally. However, intuitively existential unforgeability follows from the LRS primitive. We have appended 0 and 1 to the messages in σ_1 and σ_2 to prevent any malleability in reverting the order of signatures. For linkability, we define Link as the LRS link on the first signature and Link_{id} as the LRS link on the second signature. The anonymity of the CLRS follows intuitively like the anonymity of the underlying LRS. Even in the special

[6] Using the first signature as the message in the second signature might not be strictly necessary but can provide stronger security e.g. strong unforgeability.

case where a message should happen to collide with a label, there can be no cross-links between the first and the second part of the signatures because the labels have been appended with 0 resp. 1.

3 Conclusion

We have discussed how to prevent self-voting in many different voting contexts, from paper-based voting over centralised e-voting to decentralised voting. Especially, we designed a decentralised, robust scheme that allows public detection of self-votes using linkable ring signatures. Future work consists in proper definitions of security for the CLRS primitive and a detailed security analyses of the proposed decentralised scheme and its instantiation.

Acknowledgments A special thanks goes to Peter Y A Ryan for comments and suggestions. This research was funded in part by the Luxembourg National Research Fund (FNR), grant reference C21/IS/16221219/ImPAKT. For the purpose of open access, and in fulfilment of the obligations arising from the grant agreement, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

References

1. Ashur, T., Dunkelman, O., Talmon, N.: Breaching the privacy of israel's paper ballot voting system. In: Electronic Voting: First International Joint Conference, E-Vote-ID 2016, Bregenz, Austria, October 18-21, 2016, Proceedings 1. pp. 108–124. Springer (2017)
2. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In: Advances in Cryptology–ASIACRYPT 2012: Proceedings 18. pp. 626–643. Springer (2012)
3. Blazy, O., Bultel, X., Lafourcade, P., Kempner, O.P.: Generic plaintext equality and inequality proofs. In: International Conference on Financial Cryptography and Data Security. pp. 415–435. Springer (2021)
4. Blazy, O., Chevalier, C., Vergnaud, D.: Non-interactive zero-knowledge proofs of non-membership. In: Cryptographers' Track at the RSA Conference. pp. 145–164. Springer (2015)
5. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Annual International Cryptology Conference. pp. 126–144. Springer (2003)
6. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Conference on the Theory and Application of Cryptography. pp. 319–327. Springer (1988)
7. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Annual international cryptology conference. pp. 89–105. Springer (1992)
8. El Orche, F.E., Géraud-Stewart, R., Rønne, P.B., Bana, G., Naccache, D., Ryan, P.Y., Biroli, M., Dervishi, M., Waltsburger, H.: Time, privacy, robustness, accuracy: Trade-offs for the open vote network protocol. In: International Joint Conference on Electronic Voting. pp. 19–35. Springer International Publishing (2022)

9. Giustolisi, R., Iovino, V., Rønne, P.B.: On the possibility of non-interactive e-voting in the public-key setting. In: International Conference on Financial Cryptography and Data Security. pp. 193–208. Springer (2016)
10. Hao, F., Ryan, P.Y., Zieliński, P.: Anonymous voting by two-round public discussion. IET Information Security **4**(2), 62–67 (2010)
11. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society. pp. 61–70 (2005)
12. Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Linkable ring signature with unconditional anonymity. IEEE Transactions on Knowledge and Data Engineering **26**(1), 157–165 (2013)
13. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Australasian Conference on Information Security and Privacy. pp. 325–335. Springer (2004)
14. Liu, J.K., Wong, D.S.: Linkable ring signatures: Security models and new schemes. In: International Conference on Computational Science and Its Applications. pp. 614–623. Springer (2005)
15. McMurtry, E., Pereira, O., Teague, V.: When is a test not a proof? In: Computer Security–ESORICS 2020: Proceedings, Part II 25. pp. 23–41. Springer (2020)
16. Villar, J.: Zero-knowledge proofs notes. `https://web.mat.upc.edu/jorge.villar/doc/notes/DataProt/zk.pdf`, accessed: 29-01-2025