View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Dundee Online Publications

University of Dundee

Alethea

Basin, David; Radomirovi, Saša; Schmid, Lara

Published in:
Proceedings - IEEE 31st Computer Security Foundations Symposium, CSF 2018

DOI:
10.1109/CSF.2018.00028

Publication date:
2018

Document Version
Peer reviewed version

Link to publication in Discovery Research Portal

Citation for published version (APA):
Basin, D., Radomirovi, S., & Schmid, L. (2018). Alethea: A Provably Secure Random Sample Voting Protocol. In Proceedings - IEEE 31st Computer Security Foundations Symposium, CSF 2018 (Vol. 2018-July, pp. 283-297). [8429312] IEEE Computer Society. https://doi.org/10.1109/CSF.2018.00028

General rights
Copyright and moral rights for the publications made accessible in Discovery Research Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from Discovery Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain.
- You may freely distribute the URL identifying the publication in the public portal.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. Apr. 2019

Accepted Manuscript Version ©2018 IEEE. Paper presented at 31st IEEE Computer Security Foundations Symposium, Oxford, United Kingdom

Alethea: A Provably Secure Random Sample Voting Protocol

David Basin* Saša Radomirović† Lara Schmid*

*Institute of Information Security, Department of Computer Science, ETH Zürich
{basin, schmidla}@inf.ethz.ch

†School of Science and Engineering, University of Dundee
s.radomirovic@dundee.ac.uk

***Abstract*—In random sample voting, only a randomly chosen subset of all eligible voters are selected to vote. This poses new security challenges for the voting protocol used. In particular, one must ensure that the chosen voters were randomly selected while preserving their anonymity. Moreover, the small number of selected voters leaves little room for error and only a few manipulations of the votes may significantly change the outcome.**

We propose *Alethea*, the first random sample voting protocol that satisfies end-to-end verifiability and receipt-freeness. Our protocol makes explicit the distinction between human voters and their devices. This allows for more fine-grained statements about the required capabilities and trust assumptions of each agent than is possible in previous work. We define new security properties related to the randomness and anonymity of the sample group and the probability of undetected manipulations. We prove correctness of the protocol and its properties both using traditional paper and pen proofs and with tool support.

I. Introduction

The purpose of a democratic election is to reach a decision based on the will of the voters. Direct democracy is not the best way to achieve this goal as it is too expensive and the voters do not have the resources to make well informed decisions on all matters they can vote for. Thus alternative forms of democracy have been considered [8], [16], [18] and studying their benefits is an active research area in political economy.

Random sample voting, first proposed by *Chaum* [8], is an alternative form of democracy that aims to improve the utility of elections by polling a small number of randomly selected citizens, the *sample group*. The size of this group is chosen such that it accurately reflects the will of the entire electorate. Elections with a small sample group are cheaper than elections involving the entire electorate. Moreover, since each voter is polled less frequently and believes that his vote has a larger influence, the voters have more time and motivation to inform themselves of the issues prior to voting.

Random sample voting gives rise to new challenges. The first concerns the selection of the sample group. It is obviously crucial that the voting authority should not be able to influence the selection of the *sample voters*. It must also be verifiable that the voting authority correctly communicates the selected group to the voters. Moreover, the selection must be random and the sample group members must be anonymous, lest they be unduly influenced, harassed, or made responsible for an election outcome by those that were not selected.

Further challenges are posed by the sample group's small size. It is of paramount importance that the few votes cast are not manipulated. We must therefore work with the realistic assumption that the voters' platforms are compromised and prove that the voting protocol is end-to-end verifiable from the human voter (as opposed to his platform) to the remote server. This necessitates explicitly separating the roles of the human voter and his computing devices, in contrast to the approaches taken in previous work. We refer to Section VI-B for a comparison. The small number of votes cast also leaves little room for voter error. For a protocol to be end-to-end verifiable, the voters must perform certain checks. Yet in practice many voters will skip them. We must therefore explicitly quantify the probability that vote manipulations remain undetected, given assumptions on the number of checks and manipulations that are made.

Contributions: We propose and formally verify *Alethea*[1], the first random sample voting protocol that satisfies end-to-end verifiability and receipt-freeness. Compared to other voting protocols, we make explicit the distinction between voters and their devices. This enables us to state refined trust assumptions with respect to voters and their platforms and to analyze our protocol with respect to a more fine-grained adversary model than was possible in previous work.

We formalize the properties that are necessary for and specific to random sample voting as well as conventional properties for e-voting. In particular, we introduce the concept of *global properties* that express the probability that a cheating authority is not detected as a function of how many individual checks are made.

This research project was funded in part by the ETH Risk Center together with ETH Foundation and by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/P013694/1.

[1]*Alethea* is derived from the ancient Greek word *aletheia* meaning "the state of being evident." This name reflects one of our main goals: a verifiable voting protocol where all necessary information is disclosed to auditors.

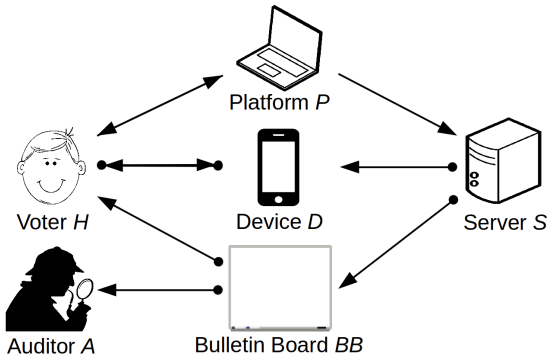

Fig. 1. System Model

We model the protocol and its properties symbolically and prove many of the properties using the Tamarin tool [20], [23]. We give hand-written proofs for those properties that cannot be efficiently verified for an unbounded number of voters by any state-of-the-art tool. This combination of automated and manual proofs allows us to prove all relevant security properties while benefiting from automation in many cases.

Organization: We introduce our setup in Section II and *Alethea* in Section III. We analyze the protocol in Section IV. We describe extensions and present related work in Sections V and VI, respectively, and conclude in Section VII.

II. Setup

We first introduce the protocol roles and communication channel assumptions, then our adversary model and trust assumptions, and finally our human agent model.

A. Roles and communication channels

Figure 1 depicts the roles and the communication channels between them. We consider the following roles in our protocol: the human voters H,[2] their platforms P, their personal devices D, the voting authority or voting server S, a public bulletin board *BB*, and the auditors A. In the actual protocol there can be several instances of each of these roles, except for the server S and bulletin board *BB*.

Following *Maurer and Schmid* [19], we write $A \rightarrow B$, $A \bullet\!\!\rightarrow B$, $A \rightarrow\!\!\bullet\ B$, and $A \bullet\!\!\rightarrow\!\!\bullet\ B$ to denote insecure, authentic, confidential, and secure channels from (instances of) role A to role B, respectively, and refer to [4] for a formal semantics for this notation. We write $A \leftrightarrow B$ and $A \bullet\!\!\leftrightarrow\!\!\bullet\ B$ for an insecure and secure channel, respectively, between A and B.

The voting server S is responsible for setting up elections and collecting and tallying the ballots. The platform P is used by the voter to send his vote to S. We assume that the platform can be compromised. Therefore, the channels between H and P as well as between P and S are insecure. As some trust is necessary for a voter to cast votes confidentially, we employ a simple personal device D for this purpose. It is more realistic to trust D than a general purpose platform as D can have limited capabilities and need not connect to the Internet. Each voter has exactly one personal device D. We assume that D is only accessible by the voter H to whom it is assigned and that D stays in H's possession, even if an adversary tries to coerce him. This is modeled by the secure channel between H and D. Also, there is an authentic channel from the server S to the device D. This channel can, for example, be established by S sending to H a QR code by post, which the voter H can scan with D.

S can authentically publish information on the bulletin board *BB*, which can then be read by voters and auditors over an authentic channel. The auditor role A specifies checks that must be performed by at least one honest party. By modeling A as a separate role, we allow it to be instantiated by anyone, including the voters. The direct link from *BB* to H indicates that H can access *BB* by means other than his insecure platform. The bulletin board's contents can, for example, be published in newspapers.

B. Adversary model and trust assumptions

We consider a Dolev-Yao adversary [13] who controls the network by learning all messages sent over the network, constructing new messages, and delivering messages. Additionally, the adversary can *compromise* some of the participating agents to learn all their secrets and control their behavior. In particular, the adversary has access to all channels to and from the compromised agents and can make them send and receive arbitrary messages. Uncompromised agents are called *honest* and they strictly perform their roles as specified.

Trusting an agent means that we assume that the agent is honest. To restrict the adversary, we make the following trust assumptions.

Trust Assumption 1. The agents instantiating the voters' devices D and the bulletin board *BB* are honest.

Thus we assume that the devices and the bulletin board follow their specifications and their keys remain secret. Furthermore, the adversary cannot send messages on the authentic channels from *BB* or receive or send messages on the secure channel to and from D. Our assumption that the bulletin board is not compromised implies that everyone agrees on its contents. Hence, the bulletin board's contents are *binding values* of the election's outcome and intermediate results.

Remark. Note that the voting server S can publish false or inconsistent information on *BB*. However, all voters and auditors will see the same information on *BB*. However, an honest *BB* does not imply that each agent arrives at the same conclusions when inspecting the *BB*. For example, a voter can only decrypt his own vote and check whether it is included in the tally. Thus one voter can conclude that his vote is missing, while another voter cannot reach this conclusion because he cannot decrypt the other voter's vote. △

For a voting protocol, it is crucial that the outcome's integrity is guaranteed even if we do not trust the voting server S. We must therefore examine verifiability properties under the assumption that S is under the adversary's full control. In our

[2]We reserve the term v for the *votes* and refer to the *voters* by H.

protocol, S is responsible for collecting the ballots and tallying them. Hence, S naturally learns how each voter voted and if S is not trusted, there are no privacy guarantees. Many protocols distribute the trust in S over a number of *tellers* (or *trustees*) using standard threshold cryptography techniques [10], [15]. As a result, only a threshold of all tellers must be trusted. As with other authors [1], [8], we consider an abstraction of this setup and model S as a single server that is trusted with respect to privacy properties. This simplifies the protocol and allows us to focus on its novel aspects. Thus, we model an honest S when examining privacy properties, but allow the adversary to compromise S when examining other security properties.

Trust Assumption 2. For privacy properties, we assume that the agent instantiating the voting server S is honest.

Next, we state the trust assumptions on the voters. If a voter is compromised, the adversary can dictate how he votes. The same holds if an adversary is physically present at all times. We therefore examine the security properties that hold for honest voters. Nevertheless, for receipt-freeness it is important to consider voters who are willing to cooperate with the adversary, either because they get something in return or because they are threatened. When analyzing receipt-freeness, we therefore include voters that reveal all their secrets to the adversary to prove how they vote. Note that except for the potential disclosure of secrets, voters follow their role specification and the adversary cannot access the secure channel to their device.

Trust Assumption 3. The agents instantiating the voters H are honest. For receipt-freeness, we assume that voters reveal all their secrets to the adversary.

Finally, we assume that the auditors are honest. The adversary can perform the same computations as the auditors because she also learns all information on *BB*.

Trust Assumption 4. The agents instantiating the auditors A are honest.

C. Human capabilities

In voting protocols, we require end-to-end guarantees from the human voters to the server. To realistically model humans, we assume that they cannot perform cryptography and that they need a device to assist them with computations. They can, however, read and learn terms, concatenate, split, and compare terms, and output terms they have learned. Such a human model was explored in detail in previous work on *Human-Interaction Security Protocols* [4].

III. Protocol

Alethea operates in two phases. In the *selection phase*, the voting authority determines the random group of sample voters by choosing a subset of the electorate based on a publicly verifiable random event. In the *voting phase*, the sample voters are authorized to vote. The main idea of our protocol is that each voter's personal device computes two codes: a *voter code* that acts as the voter's pseudonym and is used for the random selection, and a *(ballot) code* that encrypts the voter's ballot.

We first introduce the protocol model and specify each phase. We then analyze *Alethea*'s complexity.

A. Protocol model

We model the protocol as a transition system, which gives rise to a trace semantics. We present the protocol using *message sequence charts*. Each role is depicted by a vertical *life line* and named by the box on top. A role's life line depicts the role's events, sequentially ordered. For example, the first line in Figure 2 denotes the voter H's role. A role's sent and received messages are depicted on top of arrows that start at the sender and end at the receiver. We denote a role's internal computations by dashed squares and *signals* by solid squares. Signals do not have an effect on a protocol's execution, but serve to label events in executions to facilitate reasoning about the protocol's security properties.

In executions, the roles are instantiated by agents and we consider all possible interleavings of agents' runs in parallel with the adversary. A *trace* tr is a finite sequence of signals that occur in an execution. It records the messages that are sent and received by agents and the messages that are sent, received, and computed by the adversary. Furthermore, a trace contains the signals (containing auxiliary information) that we explicitly add to the protocol specification. We denote by $TR(\wp)$ the set of all traces of a protocol $\wp$.

1) Notation: If a receiver B parses a message differently than the sender A, we write x/y, where x denotes the message sent by A and y denotes the message pattern that is parsed by B. We write $x := y$ for the assignment of y to x and we write $[x_i]_{i\in\{1,\ldots,n\}}$ to denote a list of n messages of the same kind. Similarly, we write $[f(x_i, y_i)]_{i\in\{1,\ldots,n\}}$ for a list of messages, where each message has the same form, but its value differs for each entry of the list. When it is clear from context over which values we quantify, we omit the indices. For example, we write $[x]$ and $[f(x, y)]$ for the above lists, respectively.

2) Term algebra: Our trace model is based on a term algebra $\mathcal{T}$ that is generated from the application of functions in the signature Σ to the set of names $\mathcal{N}$ and variables $\mathcal{V}$. We use standard notation to denote the functions for (left-associative) pairing ($\langle\cdot,\cdot\rangle$), projection to the first (*fst*) and second (*snd*) term of a pair, a cryptographic hash function (h), asymmetric encryption ($aenc$), asymmetric decryption ($adec$), and the public key (pk) corresponding to a private key. We frequently write $\langle a, b, c\rangle$ for $\langle\langle a, b\rangle, c\rangle$ and $\{m\}_k$ for $aenc(m, k)$ and the functions $aenc$, $adec$, pk obey the equation $adec(aenc(m, pk(k)), k) = m$.

We write $\pi[x]$ to denote the permutation of a list $[x]$. The function $rand(e)$ denotes a random term that depends on an input value e. The function $select(r, [x])$ returns a sub-list of the list $[x]$ depending on r.

Alethea uses non-interactive zero knowledge proofs to decrypt ballots in a publicly verifiable manner. To this end, we define the probabilistic asymmetric encryption scheme (cp, dcp) by $dcp(cp(m, r, pk(k)), k) = m$, where r denotes randomness, k is the private key, and $pk(k)$ is the corresponding public key. We frequently write $\{m\}^r_{pk(k)}$ for

$cp(m, r, pk(k))$. We denote by $PeqP([x], [y], k)$ the non-interactive zero knowledge proof of plain text equivalence of two lists $[x]$ and $[y]$, where k is the key used to generate the proof. A proof of plain text equivalence can be verified by any party with the function $VeqP(proof, [x], [y])$, which takes as input the proof and the two lists $[x]$ and $[y]$ claimed to contain the same plain texts up to permutation. A verification is successful if the following three conditions hold. (1) The proof was constructed with respect to permutations of the two lists that were input to the verification function. (2) The elements of the first list correspond to encryptions of the elements of the second list, but can be permuted. (3) The proof was constructed with the private key k corresponding to the public key $pk(k)$ used for the encryptions. The last condition means that only someone who can decrypt the messages can construct such a proof. The following equation models these conditions, where π_1, π_2, and π_3 denote arbitrary permutations.

$$VeqP(PeqP(\pi_1[cp(m, r, pk(k))], \pi_2[m], k), \\ \pi_3[cp(m, r, pk(k))], [m]) = true$$

Such a function could, for example, be implemented by a scheme based on *Chaum-Pedersen* due to *Neff* [21].

3) Execution model: As is standard, whenever a role receives a term that it already knows, we assume that it compares the two terms and only proceeds with the protocol if they are equal. In addition, we use the signal $verify(X, p)$, where X is a term and p is a predicate, to explicitly indicate that a role checks whether the predicate p holds. In the protocol's traces, the *verify* signal is recorded as $verify(X, p, b)$, with $b \in \{true, false\}$ indicating whether the predicate p is satisfied. This allows us to refer both to the terms that are evaluated in the predicate and the predicate's truth value. For example, $verify(X, fst(\langle m_1, m_2 \rangle) = m_3, true)$ occurs in a trace if the instantiations of the terms m_1 and m_3 are equal. If the predicate is not satisfied, the agent stops its role execution.

The signal $sel(A, p)$, where p is a predicate, indicates whether a voter A believes that he is in the sample group. As with *verify*, *sel* is recorded in the traces as $sel(A, p, b)$, where $b \in \{true, false\}$ indicates whether p is satisfied. The signal $BB(x)$ indicates that the term x is posted on *BB*, $voter(A)$ that A is a voter, $device(D, A)$ that D is A's device, and $Vote(A, v)$ that v is A's vote. When necessary, we use literal indices to distinguish between different signals of the same type. Finally, $recv(A, m)$ and $send(A, m)$ indicate that a role A receives and sends the message m, respectively, but we do not explicitly include them in the message sequence charts.

We next describe the protocol phases. For readability, we only describe the protocol from the perspective of one voter. The same protocol is executed between each voter H, his device D, the platform P he uses, and the unique server S and bulletin board *BB*. Similarly, we describe one auditor role A, which can be instantiated by the voters or an external auditor.

B. Selection phase

In this phase, as depicted in Figure 2, the server generates for each voter a unique voter code and randomly selects the sample group from the set of all voter codes.

First, the server S publishes a description of a random event ev in the future. For example, S publishes a future date, time, and the name of a stock market index from which it will draw its publicly verifiable randomness. The source of the randomness is chosen such that, at the time that S publishes ev, it cannot predict the randomness that will result from this event.[3] Then, S generates for each voter a random secret x and computes the voter code y as the hash of the voter's identity and this secret. S posts the list $[y]$ of all voter codes to *BB*. S sends the voter secret x encrypted for H's device D over an authentic channel. Then D computes $y = h(H, x)$ and displays it to the voter.

An alternative to the above is for the voter code y to be directly sent from S to H on an authentic channel, for example by letter. However, having D do part of the computation has two advantages over protocols where the code is directly sent to the voter. First, if we send a code directly to the voter, we must send it in plain text because humans cannot, in general, decrypt data themselves (without auxiliary devices). This means, however, that an intruder who intercepts this message can learn the voter code.[4] If we use a device D that can perform decryptions, the message can be sent encrypted and protected from eavesdroppers. Second, we assume that the server S can be compromised, but the honest D performs computations according to the protocol. If the voter learns the voter code from D, he knows that it is of the form $h(H, x')$ for some x' that D received from S. Of course, S can send a wrong secret x', therefore the voter must verify that the received code y_D is in the list of voter codes on *BB*. As h is collision resistant, S cannot send an $x' \neq x$ for which $h(H, x')$ computed by D is equal to a y that has been computed differently. Also, S cannot make two voters believe that they have the same voter code because, for any x_1 and x_2, $H_1 \neq H_2 \implies h(H_1, x_1) \neq h(H_2, x_2)$.

After all voters have learned their respective voter codes, a subset of the voter codes are selected to form the sample group. To ensure that this selection is random and cannot be influenced by S, it is based on the random event that was previously posted to *BB*. We denote by the role E that the environment produces the randomness r from the event ev and assume that the agent instantiating E is honest. For example, ev denotes a certain stock value, date, and time, and r is the random stock value at this defined date and time. The authentic sending of r from E to A and S denotes that both A and S can observe the randomness. Furthermore, by sending ev from E to A we model that A can observe that r was generated according to the event ev.

Based on r, S computes the sample group $[y_{SG}]$, which consists of a random subset of all voter codes in $[y]$. The sample group is published on *BB* and each voter can check if

[3] In practice, one can combine different randomness sources so that an adversary can only effectively influence the randomness by controlling all sources.

[4] In the scenario where the voter code is sent from S to H by mail, the postman, or anyone with access to the mail box, could learn the voter code.

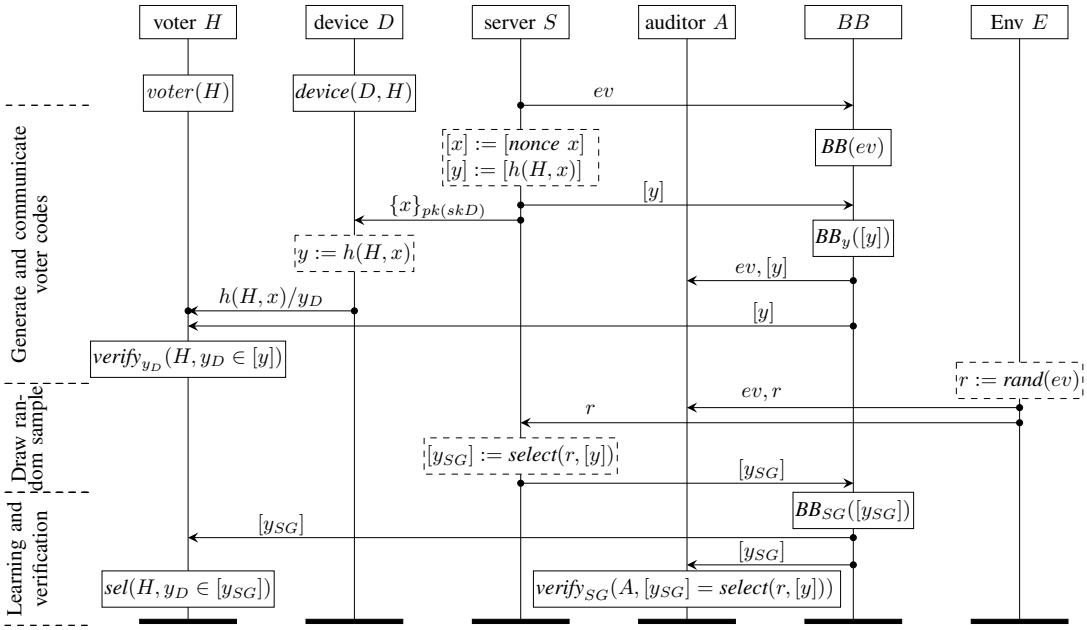

Fig. 2. Selection phase

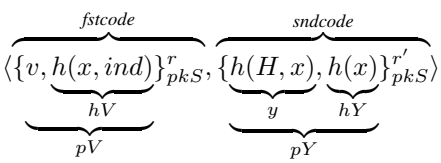

Fig. 3. Terms of the ballot code

his voter code is $[y_{SG}]$ (i.e. if he is chosen to vote), which is denoted by the signal sel. The auditor reads the sample group on the bulletin board and verifies that it was drawn according to the function *select* and the random number r.

C. Voting phase

The sample voters, which were selected in the selection phase, can cast their vote in the voting phase. To do this, a voter sends an encrypted version of the vote to S, the ballot code *code*. We first describe how the code is constructed by the voter's device D. Afterwards, we present the detailed steps of the voting phase.

1) Construction of the ballot code: If the voter H casts the vote v, his device D constructs the ballot code as

$$code := \langle \{v, h(x, ind)\}^r_{pkS}, \{h(H,x), h(x)\}^{r'}_{pkS} \rangle.$$

Figure 3 depicts the code's subterms. The code is built from two concatenated encryptions, which we denote by *fstcode* and *sndcode*. Both encryptions use the public key of the server S.

The first part, *fstcode*, encrypts the vote. In fact, it encrypts a pair pV consisting of the vote v and a hash hV of the voter secret x and an index ind. The voter secret binds together the two parts *fstcode* and *sndcode* and serves to authenticate the voter H, since only H's personal device D knows x. The index ind is a number that is chosen by the device D each time the voter enters a vote. When casting the vote, the voter sends the index, along with the code, to the server. We explain the advantage offered by this construction in Section V-B.

The code's second part, *sndcode*, also encrypts a pair, *pY*, and authenticates the voter by his voter code $y = h(H, x)$. As everyone knows the list of voter codes corresponding to sample voters, this enables auditors to check that the recorded votes were cast by sample voters. The second hash in pY, $hY = h(x)$, again ensures the authenticity and binds the two parts of the code together, as only D knows x.

Recall from Section III-A2 that cp has the property that the holder of the secret key can decrypt and permute a list of encryptions and construct a zero knowledge proof that the decrypted messages correspond to the plain texts in the encryptions, up to permutation. The construction of the codes thus allows S to verifiably decrypt the votes and voter codes.

2) Detailed voting phase: Figure 4 depicts the voting phase. First, the voter H enters his vote v on the device D. D then computes the code as described above, and displays it to the voter, together with the voter's identity H and the index ind. Using a trusted device to compute the code has advantages similar to the ones described in the selection phase. To cast the vote, the voter enters this code and the index on P, which sends it on to S. A possible realization of this communication from D to P that is triggered by H is that D displays a QR code which is scanned by H with P.

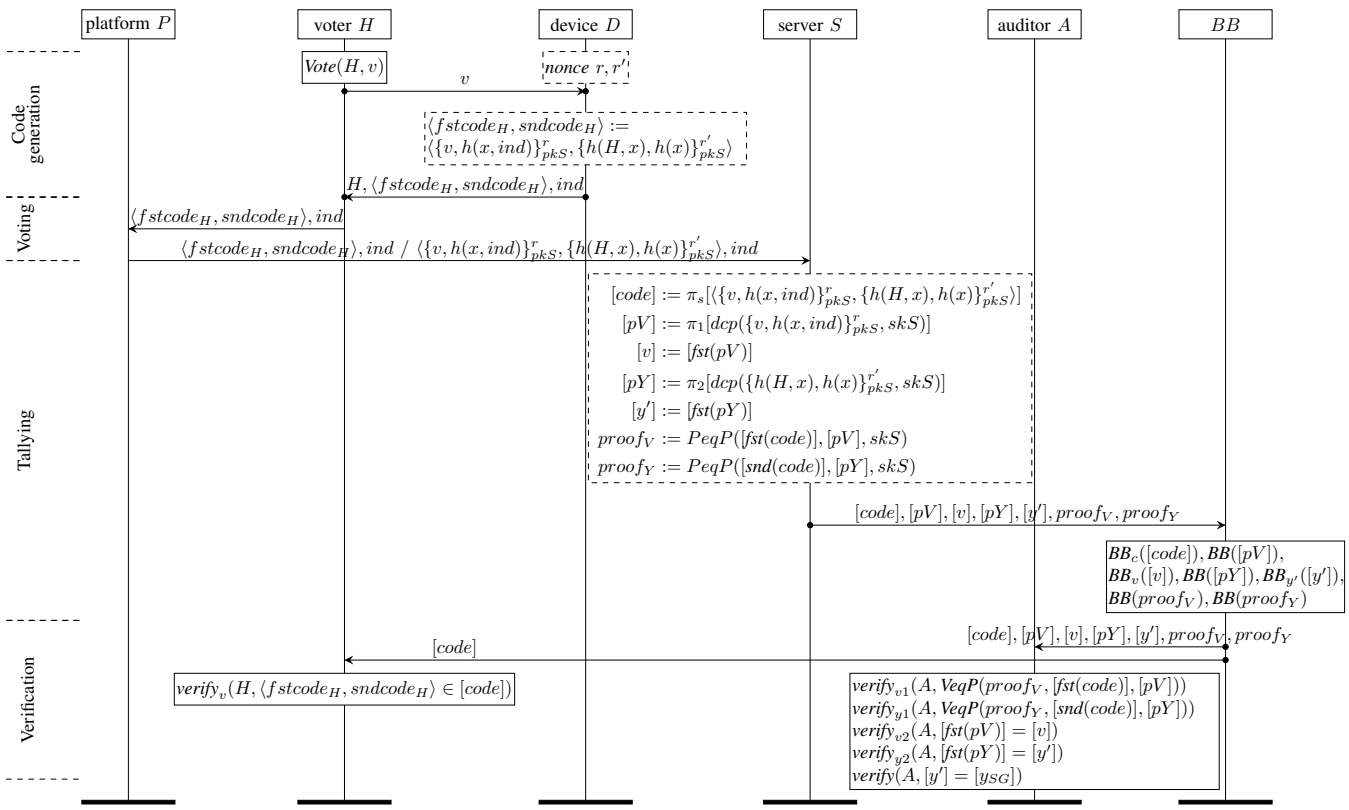

Fig. 4. Voting phase

The server can interpret the code with the help of the received index ind and only accepts it if it has the right form. S collects all codes in the list $[code]$. To avoid revealing private information, the codes are sorted in this list, denoted by the permutation π_S. S then decrypts the first parts of the codes to the list of pairs $[pV]$. The first elements of $[pV]$ are the votes defining the final tally. Also, S decrypts the codes' second parts into $[pY]$, which contains the voter codes $[y']$. The permutations π_1 and π_2 ensure that the orders of the elements do not reveal which decryptions correspond to which encryptions. Additionally, S constructs zero knowledge proofs that both parts have been decrypted correctly and posts all these lists and the proofs on the bulletin board.

An auditor A then verifies the following properties: (1) The two parts of the codes have been correctly decrypted to the published lists of pairs pV and pY, as the proofs of plain text equivalence are verified. (2) The first part of the elements in $[pV]$ and $[pY]$ correspond to the published votes $[v]$ and voter codes $[y']$, respectively. (3) The published voter codes $[y']$ correspond to the voters in the sample group. Each sample voter also reads the list of codes from the bulletin board and verifies that it includes his code.

3) Voting with abstention: We do not require that each sample voter casts a vote. However, even voters who abstain may perform the individual verifiability check that no vote was recorded for them. If a voter decides not to cast a vote, signaled by *Vote*$(H,$ *'empty'*$)$, a constant string *'empty'* is recorded as his vote. Compared to the standard voting protocol in Figure 4, the difference is that the voter never sends anything to S over P and therefore these steps are omitted. To learn the code corresponding to the empty vote, the voter enters a designated code word, or presses a designated *'empty'* button on D.

As the voter does not cast a vote, it is impractical for him to send ind to S. We therefore assume a predefined default index ind_0 and secret nonces r_0 and r'_0, known to both D and S. When the voter enters *'empty'*, D uses ind_0, r_0, and r'_0 to compute the code. After the vote casting phase has ended, for each voter that has not sent a vote, S computes the empty ballot code also using ind_0, r_0, and r'_0.

D. Complexity

We briefly summarize the roles' time complexities in terms of the number of voters n and the number of sample voters $m \leq n$. We define the size of a term in our term algebra to be the number of function symbols and names that occur in the term. We consider the application of functions in the signature Σ as well as the comparison, sending, and reception of terms as basic operations, thus requiring time proportional to the size of the terms. We prove the following lemma in Appendix A.

Lemma 1. The time complexity of all roles is linear in the number of voters n, except for the auditor role, which has complexity $O(n) + O(m \log(m))$.

IV. Analysis

We assume that the voting phase only starts after the selection phase has finished. Consequently, messages from the two phases cannot be interleaved. We thus analyze the two phases separately and use in the voting phase the fact that a successful selection phase has already established certain common knowledge shared between the agents.

We establish three main kinds of properties for *Alethea*: verifiability properties, global properties, and privacy properties. Verifiability properties are trace properties that are conditional on an agent performing a particular protocol step, typically a verification step. We say that a protocol satisfies a trace property if every trace of the protocol satisfies the property.

Our verifiability properties guarantee that a cheating server is caught provided the voters and auditors follow the protocol specification and perform the necessary verification steps. In practice, however, not all voters will perform these checks. Our global properties account for this and quantify the probability that a server is not caught cheating when it manipulates a number of voter codes or ballot codes. As these properties are novel and require introducing additional notation, we describe them in more detail in the next section.

Privacy properties express that an adversary cannot learn certain relations, for example who voted for whom. We define privacy properties as observational equivalence properties that express that an adversary cannot distinguish between two systems: a *left system* and a *right system*. For example, the adversary cannot distinguish one system (left) where a voter votes according to the adversary's will from another system (right) where the voter votes according to his own choice. More specifically, we define a set $\mathcal{S}$ of trace pairs (tr_L, tr_R) where tr_L is from the left system and tr_R from the right system. A protocol satisfies a privacy property if for all its traces in one system there exists a trace in the other system such that the pair of traces is contained in $\mathcal{S}$.

We use the Tamarin tool to verify verifiability and privacy properties (see [24] for the Tamarin input files). For the latter, we use Tamarin's built in support for observational equivalence [2]. We consider two models, one with an honest and one with a compromised server S. As some of the universal verifiability properties cannot be efficiently verified with Tamarin for a large number of voters, we model two voters in Tamarin. Whenever we prove properties for an arbitrary number of voters, we use hand-written proofs for this general case. In some cases, we need permutations of lists, for example to state that a zero knowledge proof of plain text equivalence is verified. To state that each permutation of the inputs is valid, we explicitly model each possible permutation in Tamarin.

Finally, we prove that S cannot influence the sample group and that the sample group is random. In the symbolic model, these properties follow directly from the assumptions on the selection function *select*. However, as these properties are crucial to the proper functioning of random sample voting, we give details of how *select* could be realized and prove them at this lower level of abstraction.

A. Global verifiability properties

To prove that a protocol satisfies a property, we consider all possible interleavings of the agents instantiating their roles with the adversary. In practice, humans do not follow their role specification precisely. For example, it is unlikely that every voter performs all the necessary checks to detect manipulation by the server S. This implies that there will always be traces where S successfully manipulates the election's outcome. Therefore, in addition to reasoning about the *possibility* of attacks, our global properties are used to reason about the *probability* that manipulations by S are not detected.

We define a probability space $(\Omega, \mathcal{F}, Prob)$, where Ω is the set of all possible outcomes (the sample space), $\mathcal{F} \subseteq 2^{\Omega}$ is the set of events, and $Prob : \mathcal{F} \to [0, 1]$ is the probability measure. $\mathcal{F}$ is a σ-algebra, i.e., closed under complementation and countable unions. In addition to the set of traces $\mathit{TR}(\wp)$, we include in Ω those traces that arise from voters skipping their verification steps. Formally, we consider the protocol $\wp'$ obtained from $\wp$ by adding three voter roles that are identical to role H except that the first, second, or both *verify* signals specified for H in $\wp$ are missing. A voter then has the choice of executing one of these four roles and this choice determines which verification steps are skipped. Skipping verification steps means that the voter will not notice a manipulation. We set $\Omega = \mathit{TR}(\wp')$.

We assume that the probability measure $Prob$ is such that the event of a voter performing a check is independent of the event that the voter's check succeeds.

Definition 1. Let $(\Omega, \mathcal{F}, Prob)$ be a probability space of a protocol $\wp$ as defined above. Let $V_s(H)$, $V_v(H)$ be the events that H is a voter and H is a member of the sample group, respectively. The indices s and v denote that an event is associated with the selection and voting phase, respectively. Let $X_s(H), X_v(H)$ be the events that H's voter code or ballot code was manipulated. Finally, let $Y_s(H), Y_v(H)$ be the events that H checks that his voter code and his ballot code is on *BB*, respectively. Formally,

$$
\begin{aligned}
V_s(H) &= \{tr \in \Omega \mid voter(H) \in tr\} \\
V_v(H) &= \{tr \in \Omega \mid \exists v.\ \mathit{Vote}(H, v) \in tr\} \\
X_s(H) &= \{tr \in \Omega \mid \exists D.\ \forall x, [y], k.\ device(D, H) \in tr \wedge \\
&\quad \neg(\mathit{recv}(D, \{x\}_k) \in tr \wedge \mathit{BB}_y([y]) \in tr \wedge h(H, x) \in [y])\} \\
X_v(H) &= \{tr \in \Omega \mid \exists c_H, ind.\ \forall [code]. \\
&\quad (\mathit{send}(H, \text{'empty'}) \in tr \wedge \mathit{recv}(H, \langle H, c_H, ind\rangle) \in tr \\
&\quad \vee \mathit{send}(H, \langle c_H, ind\rangle) \in tr) \wedge \neg(\mathit{BB}_c([code]) \wedge c_H \in [code])\} \\
Y_s(H) &= \{tr \in \Omega \mid \exists p, b.\ \mathit{verify}_{y_D}(H, p, b)\} \\
Y_v(H) &= \{tr \in \Omega \mid \exists p, b.\ \mathit{verify}_v(H, p, b)\}
\end{aligned}
$$

We say that *verification is independent of manipulation* in the probability space of a voting protocol $\wp$, if for all H,

$X_s(H)$ and $Y_s(H)$ are conditionally independent events given $V_s(H)$ and if $X_v(H)$ and $Y_v(H)$ are conditionally independent events given $V_v(H)$.

We define a global property for an event $X \subseteq \Omega$, random variables $X_1, \ldots, X_n : \Omega \to \mathbb{N}$, a function $F : \mathbb{N}^n \to [0,1]$, and relations $\sim_1, \ldots, \sim_n, \sim \in \{=, \leq, \geq\}$. A global property has the form $\{tr \in \Omega \mid Prob(X|X_1(tr) \sim_1 j_1, \ldots, X_n(tr) \sim_n j_n) \sim F(j_1, \ldots, j_n)\}$. The global property is satisfied if for every trace tr of the sample space the conditional probability of X given the event characterized by the relations on $X_1, \ldots, X_n$, satisfies the indicated relation.

In our global properties, X indicates that the voting server's manipulation is not detected. The random variables count the number of voters, voters that perform verifiability checks, and manipulated codes, and F gives a bound on the probability that the voting server's manipulation is not detected given those counts. See e.g., Definition 5.

B. Analysis of selection phase

1) Verifiability properties: It is essential that S publishes exactly one voter code for each voter. Otherwise, some voters are never considered for the sample group. We therefore require the verifiability of the voter codes. As the correspondence between voters and voter codes must remain secret, we define this property as an individual verifiability property. For notational simplicity, when using set comprehension notation like $\{a \mid F(a)\}$, all free variables b different from a in F are implicitly universally quantified.

Definition 2.

Individual verifiability of voter code :=

$$\{tr \mid \mathit{verify}_{y_D}(H, y_D \in [y], \mathit{true}) \in tr \implies \exists [y'], x. \mathit{BB}_y([y']) \in tr \wedge y_D \in [y'] \wedge y_D = h(H, x)\}$$

The definition states that whenever a voter H verifies that his voter code y_D is included in the list of voter codes $[y]$, then y_D is really part of the published voter code list and was correctly computed for the voter H. We establish the following lemma with Tamarin.

Lemma 2. *Alethea* satisfies individual verifiability of voter code, even with a compromised server S.

Additionally, a voter must be able to verify whether he was selected into the sample group. We define that whenever H, who received voter code y_D, concludes that he is selected, then his voter code is included in the list of sample voters and is correctly computed for H.

Definition 3.

Individual verifiability of the selection :=

$$\{tr \mid \mathit{sel}(H, y_D \in [y_{SG}], \mathit{true}) \in tr \implies \exists [y'_{SG}], x. \mathit{BB}_{SG}([y'_{SG}]) \in tr \wedge y_D \in [y'_{SG}] \wedge y_D = h(H, x)\}$$

Tamarin verifies the following lemma.

Lemma 3. *Alethea* satisfies individual verifiability of the selection, even with a compromised server S.

In addition to the individual verifiability properties, it must be universally verifiable that S computed the sample group from the list of eligible voters according to the protocol specification.

Definition 4.

Universal verifiability of the selection :=

$$\{tr \mid \mathit{verify}_{SG}(A, [y_{SG}] = \mathit{select}(r, [y]), \mathit{true}) \in tr \implies \mathit{BB}_y([y]) \in tr \wedge \mathit{BB}_{SG}([y_{SG}]) \in tr \wedge [y_{SG}] = \mathit{select}(r, [y])\}$$

The first part states that an auditor A verifies that the sample group $[y_{SG}]$ is correctly computed by the function *select* taking as inputs the randomness r from the environment and the list of voter codes $[y]$. The definition states that whenever this check holds, *BB* contains the same lists $[y]$ and $[y_{SG}]$ and thus the sample voters $[y_{SG}]$ are correctly computed from r and $[y]$. We prove the following lemma for two voters in Tamarin and for arbitrary many voters in Appendix B by hand.

Lemma 4. *Alethea* satisfies universal verifiability of the selection, even with a compromised server S.

2) Global properties: We have established that each voter can verify that there is a voter code included for him. However, not all voters will perform this check. Therefore, it is important to examine what global property holds when only a fraction of the voters verify the inclusion of their voter code.

Definition 5. Let $V, X, Y : \Omega \to \mathbb{N}$ be random variables that count, respectively, the number of registered voters in a trace, the number of voters whose voter code is not installed on their device or incorrectly recorded on the bulletin board, and the number of voters that check that their voter code is included on the bulletin board. Let Z be the event that no manipulation is detected. Formally,

$$V(tr) = |\{H \mid voter(H) \in tr\}|$$
$$X(tr) = |\{H \mid \exists D. \forall x, [y], k. \mathit{device}(D, H) \in tr \wedge \neg(\mathit{recv}(D, \{x\}_k) \in tr \wedge \mathit{BB}_y([y]) \in tr \wedge h(H, x) \in [y])\}|$$
$$Y(tr) = \left|\{H \mid \exists p, b. \mathit{verify}_{y_D}(H, p, b) \in tr\}\right|$$
$$Z = \{tr \in \Omega \mid \forall H, p. \mathit{verify}_{y_D}(H, p, \mathit{false}) \notin tr\}$$

Global individual verifiability of voter codes is defined by the following set of traces:

$$\{tr \mid Prob(Z \mid V(tr) = n, X(tr) = o, Y(tr) = a) \leq \left(\tfrac{n-o}{n}\right)^a\}.$$

The definition states that the probability that no manipulation is detected is at most $(\frac{n-o}{n})^a$, when there are n registered voters, o voter codes that are manipulated, and a voters check that their voter code is on the bulletin board. For example, for 1000 voters, if 100 voter codes are manipulated and 50 voters check whether their voter code is on *BB*, then the probability that no one detects the manipulation is at most .005. But if only one code is manipulated then, even when 900 voters check their codes, the probability that the manipulation is not detected is at most .406. In this case, however, a cheating voting authority cannot substantially influence the selection.

We prove the following lemma in Appendix B.

Lemma 5. Let the probability measure $Prob$ be such that verification is independent of manipulation (Definition 1). Then *Alethea* satisfies global individual verifiability of voter codes.

3) Privacy properties: In random sample voting, the set of eligible voters is usually considerably smaller than in conventional voting. Therefore, to change the election outcome, an adversary needs to compromise fewer voters. To counter this threat, it is crucial that the adversary cannot learn which voters are in the sample group. We denote this property by *sample group anonymity* and model it as a privacy property.

We define sample group anonymity as the property that an adversary cannot distinguish the following two systems with two voters, A and B. In both systems, S generates A's and B's voter code as $y_A = h(A, x_A)$ and $y_B = h(B, x_B)$. In the left system, the voter code y_A is selected, thus A is in the sample group, and in the right system B's code y_B is selected instead. The next lemma states that the adversary cannot distinguish whether A or B is in the sample group.

We write $t_1 \approx t_2$ to denote that two traces are indistinguishable for an adversary. We use this definition informally here and refer to [2] for a formal definition of $\approx$.

Definition 6. Let $\mathit{TR}(\wp_L)$ be the set of all traces of $\wp$ with two voters A and B where A is selected for the sample group. Let $\mathit{TR}(\wp_R)$ be defined similarly, except that B is selected instead to be the sample voter.

$$\textit{Anonymity of the sample group} := \{(tr_L, tr_R) \in \mathit{TR}(\wp_L) \times \mathit{TR}(\wp_R) | tr_L \approx tr_R\}$$

The set defines the pairs of traces that are observationally equivalent, where the first traces are from the left and the second traces are from the right system. If a protocol fulfills this property, for each trace in the left system, where A is selected, there exists an indistinguishable trace in the right system, where B is chosen, and vice versa. An outside adversary thus cannot determine if a given voter is in the sample group. The following lemma is verified by Tamarin.

Lemma 6. *Alethea* satisfies anonymity of the sample group for an honest server S.

C. Analysis of voting phase

1) Verifiability properties: In the voting phase, only the voter must know his intended vote. Therefore, to check that his vote was recorded as intended, each voter must carry out an individual verifiability check. Our definition of individual verifiability is based on [17].

Definition 7.

$$\begin{aligned}\textit{Individual verifiability} := \{tr \mid (\mathit{Vote}(H, v_H) \in tr \wedge \\ \mathit{verify}_v(H, \langle \mathit{fstcode}_H, \mathit{sndcode}_H\rangle \in [\mathit{code}], \mathit{true}) \in tr) \\ \implies \exists hV, pkS, r, [\mathit{code}'].\mathit{BB}_c([\mathit{code}']) \in tr \wedge \\ \langle \mathit{fstcode}_H, \mathit{sndcode}_H\rangle \in [\mathit{code}'] \wedge \mathit{fstcode}_H = \{v_H, hV\}^r_{pkS}\}\end{aligned}$$

This states that whenever a voter verifies that his code $\langle \mathit{fstcode}_H, \mathit{sndcode}_H\rangle$ is in the list of all recorded codes $[\mathit{code}]$, then one of the recorded codes on *BB* corresponds to his vote v_H. The following lemma is established with Tamarin.

Lemma 7. *Alethea* satisfies individual verifiability, even with a compromised server S.

We next establish the universal verifiability property that any auditor can verify that the votes are counted as recorded. This property is essential since we do not trust S with respect to the integrity of the voting result. Recall that all the checks can be done by the voter H himself who can also instantiate the auditor role.

Definition 8.

$$\begin{aligned}\textit{Universal verifiability of the tally} := \\ \{tr \mid (\mathit{verify}_{v1}(A, \mathit{VeqP}(\mathit{proof}_V, [\mathit{fst}(\mathit{code})], [pV]), \mathit{true}) \in tr \wedge \\ \mathit{verify}_{v2}(A, [\mathit{fst}(pV)] = [v], \mathit{true}) \in tr) \\ \implies \mathit{BB}_c([\mathit{code}]) \in tr \wedge \mathit{BB}_v([v]) \in tr \wedge \\ \exists [hV], [\mathit{sndcode}], [r], k, \pi.\ \pi[\mathit{code}] = [\langle \{v, hV\}^r_k, \mathit{sndcode}\rangle]\}\end{aligned}$$

The left side of the implication denotes that an auditor verifies that the first parts of the codes were correctly decrypted into the pairs $[pV]$ and the first elements of these pairs correspond to the published votes. If these checks are verified, then the bulletin board contains the same lists of codes and votes, such that the codes are correct encodings of the votes, but their order can be permuted. We establish the following lemma for two voters in Tamarin and complete the proof for arbitrarily many voters in Appendix C.

Lemma 8. *Alethea* satisfies universal verifiability of the tally, even with a compromised server S.

End-to-end verifiability is the conjunction of individual and universal verifiability. The following theorem therefore follows from Lemmas 7 and 8.

Theorem 1. *Alethea* satisfies end-to-end verifiability, even with a compromised server S.

For random sample voting, it must also be verifiable that only the selected sample voters cast votes. For this purpose, each ballot code includes the sender's voter code in the second encryption. We establish that it is verifiable that S correctly decrypts this part of the codes. As it is public which voter codes correspond to sample voters, auditors can then easily check that all votes were sent by sample voters.

Definition 9.

$$\begin{aligned}\textit{Universal verifiability of voter codes} := \{tr \mid \\ (\mathit{verify}_{y1}(A, \mathit{VeqP}(\mathit{proof}_Y, [\mathit{snd}(\mathit{code})], [pY]), \mathit{true}) \in tr \wedge \\ \mathit{verify}_{y2}(A, [\mathit{fst}(pY)] = [y'], \mathit{true}) \in tr) \\ \implies \mathit{BB}_c([\mathit{code}]) \in tr \wedge \mathit{BB}_{y'}([y']) \in tr \wedge \\ \exists [hY], [\mathit{fstcode}], [r'], k, \pi.\ \pi[\mathit{code}] = [\langle \mathit{fstcode}, \{y', hY\}^{r'}_k\rangle]\}\end{aligned}$$

Similarly to universal verifiability of the tally, the definition states that if an auditor verifies that the second part of the

codes have correctly been decrypted to the pairs $[pY]$ and that each first part of a pair pY corresponds to a unique voter code $y' \in [y']$, then all voter codes $[y']$ on *BB* are contained in a unique *code* on *BB*. We automatically prove the following lemma for two voters in Tamarin and manually prove it for arbitrarily many voters in Appendix C.

Lemma 9. *Alethea* satisfies universal verifiability of voter codes, even with a compromised server S.

2) Global properties: As in the selection phase, we examine the probability of an undetected manipulation by the server, given the number of individual verifiability checks made. We only considers the sample voters, as only they can cast a vote, and denote the number of sample voters by m.

Definition 10. Let $V, X, Y : \Omega \to \mathbb{N}$ be random variables that count, respectively, the number of sample voters in a trace, the number of voters whose ballot code is not correctly included on the bulletin board, and the number of voters that check whether their code is included on the bulletin board. Let Z be the event that no manipulation is detected. Formally,

$$V(tr) = |\{H \mid \exists v.\ \mathit{Vote}(H, v) \in tr\}|$$
$$X(tr) = |\{H \mid \exists c_H, ind. \forall [code].$$
$$(\mathit{send}(H, \text{'empty'}) \in tr \wedge \mathit{recv}(H, \langle H, c_H, ind\rangle) \in tr$$
$$\vee \mathit{send}(H, \langle c_H, ind\rangle) \in tr) \wedge \neg(\mathit{BB}_c([code]) \wedge c_H \in [code])\}|$$
$$Y(tr) = |\{H \mid \exists p, b.\ \mathit{verify}_v(H, p, b) \in tr\}|$$
$$Z = \{tr \in \Omega \mid \forall H, p.\ \mathit{verify}_v(H, p, \mathit{false}) \notin tr\}$$

Global individual verifiability of votes is defined by the set

$$\{tr | Prob(Z | V(tr) = m, X(tr) = o, Y(tr) = a) \leq \left(\tfrac{m-o}{m}\right)^a\}.$$

The definition gives an upper bound for the probability that no one detects that S has manipulated some ballot codes. The probability is computed under the assumption that there are m sample voters, a voters perform their individual checks, and S did not correctly include o ballots. We prove the following lemma in Appendix C.

Lemma 10. If the probability measure $Prob$ is such that verification is independent of manipulation (Definition 1), then *Alethea* satisfies global individual verifiability of votes.

3) Privacy properties: Privacy denotes that an adversary cannot link voters to their votes. We define it as the property that an adversary cannot distinguish a left system where voter A votes v_1 and voter B votes v_2 from a right system where A votes v_2 and B votes v_1 [6], [12]. We use the notation $\wp_{m_1 \leftarrow m_1', m_2 \leftarrow m_2'}$ to denote the specification of the protocol $\wp$ where each occurrence of the terms m_1 and m_2 is replaced by m_1' and m_2', respectively.

Definition 11. Let v_A and v_B be the term that denotes A's and B's vote, respectively, and let v_1 and v_2 be message terms.

Vote privacy :=

$$\{(tr_L, tr_R) \in \mathit{TR}(\wp_{v_A \leftarrow v_1, v_B \leftarrow v_2}) \times \mathit{TR}(\wp_{v_A \leftarrow v_2, v_B \leftarrow v_1}) \mid tr_L \approx tr_R\}$$

This set contains all indistinguishable trace pairs, such that the first trace is from the left system, where A votes v_1 and B votes v_2, and the second trace is from the right system, where A votes v_2 and B votes v_1. Recall from Section II-B that we establish privacy properties with respect to an honest server S. The following lemma is proven by Tamarin.

Lemma 11. *Alethea* satisfies vote privacy with an honest server S.

Even though we assume honest voters (Trust Assumption 3), it is important that a protocol preserves privacy even if voters are forced by an adversary to reveal private information. A protocol is *receipt-free*, if it is not possible for a voter to generate a receipt for how he voted, even if he reveals his secrets to the adversary.

To model receipt-freeness, we consider two voters, A and B, and change the original protocol $\wp$ to $\wp'$ where A sends all his secrets to the adversary except that A always claims that his vote is v_1. As with vote privacy, A votes v_1 in the left and v_2 in the right system. Voter B votes in each system opposite to A so that the end result is the same in both systems [12]. We define receipt-freeness as privacy, but with respect to $\wp'$.

Definition 12. Let $\wp'$ be the protocol obtained from $\wp$ as described above, let v_A and v_B be the term that denotes A's and B's vote, respectively, and let v_1 and v_2 be message terms. Receipt-freeness of the protocol $\wp$ is defined as follows.

Receipt-freeness :=

$$\{(tr_L, tr_R) \in \mathit{TR}(\wp'_{v_A \leftarrow v_1, v_B \leftarrow v_2}) \times \mathit{TR}(\wp'_{v_A \leftarrow v_2, v_B \leftarrow v_1}) \mid tr_L \approx tr_R\}$$

This set defines all indistinguishable trace pairs such that the traces are from two systems where A and B vote the opposite way. The difference to privacy is that A reveals all secrets, except he claims in both systems that he votes v_1, which is only true in the left system. We establish with Tamarin that the following lemma holds.

Lemma 12. *Alethea* satisfies receipt-freeness with an honest server S.

D. Randomness of the sample group

In this section we assume that the bulletin board contains exactly one voter code per voter. We established by global individual verifiability of voter codes the probability that no one detects a manipulation if this does not hold. Under this assumption and assuming that the output produced by the environment E based on event ev is random, we show next that S cannot influence the sample group's selection, no matter how S chooses the voter codes for the voters. Moreover, we show that the sample group is indistinguishable from random. To prove these properties on a more detailed level than the symbolic model allows, we first define pseudo-random permutations and explain how we construct the function *select* based on them.

Let $\mathcal{G}$ be the group of all permutations of lists of length n. A random permutation (RP) π^R is chosen from $\mathcal{G}$ uniformly at

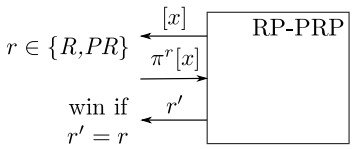

Fig. 5. Game PR-PRP

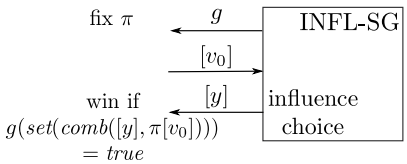

Fig. 6. Game INFL–SG

random. A pseudo-random permutation (PRP) π^{PR} is chosen at random from a subset of $\mathcal{G}$'s elements, denoted by $subset(\mathcal{G})$.

As is standard, we assume that a PRP is indistinguishable from a RP for the adversary. More precisely, we define indistinguishability by the security game depicted in Figure 5. In the game, the adversary provides the environment with an input $[x]$. The environment chooses $r \in \{R, PR\}$ and outputs $\pi^r[x]$. The adversary must output a decision r'. The adversary's *advantage* is defined by $|\Pr[r' = PR \mid r = PR] - \Pr[r' = PR \mid r = R]|$ and we assume that it is negligible as a function of n.

To define the function *select* that samples m voters from n eligible voters, we first define the initialization list $[v_0] = [1, 1, ..., 1, 0, 0, ..., 0]$, where the first m entries are 1 and the remaining $n - m$ entries are 0.

Definition 13. Let $comb([vc], [b]) : voterList \times selectionList \rightarrow voterAssignment$ be a function of two inputs, a list of voter codes and a list of bits. The function's output is a list of voter codes and zeros and all three lists are the same length. The function combines the lists element-wise as follows.

$$comb([vc], [b]) = \left[\begin{cases} vc_i & \text{if } b_i = 1 \\ 0 & \text{if } b_i = 0 \end{cases} \right]_{i \in \{1,\ldots,n\}}$$

Let $set([l])$ be a function that takes a list $[l]$ as input and outputs the set of its *nonzero* elements. We define the sample group as the set $SG = set(comb([y], \pi[v_0]))$, where the voter codes $[y]$ can be chosen by S, but such that each voter has exactly one corresponding voter code. The randomness r in the function $select(r, [y])$ can be understood to determine π. Given this construction, we first assume a random permutation $\pi \in \mathcal{G}$ and show that S cannot influence SG in this optimal case. We prove the lemma in Appendix D.

Lemma 13. For each list of voter codes $[y]$, if $\pi \in \mathcal{G}$ is selected uniformly at random, then $SG = set(comb([y], \pi[v_0]))$ is a uniformly random m-element subset of $set([y])$.

We next show that, even if π is a PRP, S cannot influence the sample group by its choice of $[y]$. For this purpose, we define a game *INFL–SG* that an adversary wins if she can successfully influence the sample group as determined by a predicate g that denotes the adversary's goal. Given a set s, $g(s)$ is *true* if the adversary has reached her goal on the selection s. We are only interested in goals for which a computationally bounded adversary can judge whether she succeeded. Therefore, we only consider polynomial time functions g.

Definition 14. The game INFL–SG is depicted in Figure 6 and runs as follows. The environment fixes a permutation π and the adversary outputs a goal g. Then the environment gives the list $[v_0]$ to the adversary and the adversary outputs the voter codes $[y]$. She wins the game if $g(set(comb([y], \pi[v_0])))$ holds.

Let $|SG_g| = |\{SG \mid \exists [y], \pi.\ SG = set(comb([y], \pi[v_0])) \wedge g(SG)\}|$ be the number of valid sample group choices that meet the goal g and let $SG_\#$ be the total number of possible sample groups. We have shown in Lemma 13 that if π is random, the choice of sample group SG is random. Thus, the probability that the result meets the goal g is given by $p_R = \frac{SG_g}{SG_\#}$, independent of S's choice of $[y]$. The following lemma states that even if π is a PRP, S cannot significantly improve the probability of influencing the sample group. We define the adversary's advantage in winning the INFL–SG game by $|\Pr[g = true \mid r = PR] - \Pr[g = true \mid r = R]|$. We refer to Appendix D for the proof.

Lemma 14. The adversary's advantage in winning the INFL–SG game with a PRP over an RP is bounded by her advantage in the RP–PRP game.

If follows from the preceding lemma that the selection of the sample group is indistinguishable from random for the adversary even if a PRP π is used.

V. Extensions

We next present different protocol extensions and explain informally how they achieve even stronger security guarantees.

A. Improved verifiability of voter codes

We established in Lemma 2 that for each voter who verifies a check, there is a voter code on *BB*. Furthermore, Lemma 5 established that even if not all voters make a check, the probability that no manipulation is detected is low. Next we describe how to further decrease this probability by a factor $\frac{1}{d}$ for any positive integer d.

We propose a standard cut-and-choose mechanism where the server S produces d sets of voter codes. That is, for each voter, S computes d many secrets x and corresponding voter codes $y = h(H, x)$. S then posts commitments to all these values on *BB*. Based on a second publicly verifiable random event ev_2, a random number $k := rand(ev_2) \in \{1, \ldots, d\}$ is drawn, which decides that the kth set of voter codes is the one to be used in the election. S then posts the voter codes of the kth computation on *BB*. Furthermore, S reveals all values x and y from the remaining $d - 1$ sets. An auditor can verify that all published values match the previous commitments

and that S computed y correctly from x in all revealed sets. An incorrect computation of S is not detected only with probability $\frac{1}{d}$.

B. Towards coercion resistance

We analyzed our protocol against a remote adversary who can only learn those terms the voters send to the network. We introduce an extension that allows an adversary to be with the voter, except for the moment when the voter enters his vote on the device. Recall the voting phase from Figure 4. A voter can enter votes as often as he wants on his device D and, each time, D computes a new code based on a fresh index ind. This ensures that if a voter learns the ballot code in the absence of the adversary, he cannot later reproduce a receipt for the vote. Each produced code will be new, no matter what vote-choice is entered.

Similarly, if a voter chooses to abstain from voting, he enters on his device that he wants to cast the empty vote. The first time he does so, the voter learns the correct code, which is based on the index ind_0. Each subsequent time he enters *'empty'*, the device will use a new index. Thus, if the adversary is not present at the time of the first check, she cannot learn that a voter abstained from voting. This is similar to the assumption that the adversary cannot be present at the time when a voter who casts a vote learns his code.

Finally, note that a voter could video record his interaction with D to prove to an adversary how he voted. This scenario is similar to the case where the adversary is always with the voter and we do not attempt to solve this. However, to mitigate the motivation of a voter to engage in such an attack, the device displays the voter's identity H together with the code. If H thus records the process and sends a video of it to somebody, he risks that legal entities learn that he tried to sell his vote. For this to work, the voter's identity must be displayed in a way that it is impossible for H to hide its identity and display the code.

VI. Related work

We discuss related work on random sample voting, (classical) voting protocols, and security properties.

A. Random sample voting

Chaum's [8] account of random sample voting is the closest related work. In contrast to our analysis, Chaum does not formally prove his claimed security properties. Another difference is that Chaum considers a much weaker adversary. Chaum's adversary is rational and only performs attacks that benefit her. We consider a standard Dolev-Yao adversary, who always attacks, regardless of the benefit.

Alethea satisfies receipt-freeness in the Dolev-Yao adversary model, as a voter cannot reveal any secrets to the adversary that constitute a proof of how he voted. Chaum suggests to avoid coercion by introducing *decoy ballots*. These are ballots that can be sold to an adversary because they are indistinguishable from real ballots; however they are not counted in the final tally. Thus, Chaum's protocol does not satisfy receipt-freeness in a Dolev-Yao adversary model since a legitimate voter can prove to the adversary how he voted. Nevertheless, the adversary does not know if a given receipt is for a real or for a decoy ballot. Therefore, a rational adversary is not motivated to engage in vote buying.[5]

Chaum's protocol and *Alethea* also differ in how the sample group is published. With *Alethea* each voter learns whether he is selected by checking the information on the *BB*. In contrast, in Chaum's protocol the sample voters learn that they have been chosen to vote by the fact that they receive a ballot without requesting it. To verify that each sample voter indeed received his ballot, they must be explicitly asked by an external auditor at the end of the protocol.

B. Voting protocols

Alethea's voting phase is a remote voting protocol that achieves both end-to-end verifiability and receipt-freeness. There are many voting protocols that satisfy similar properties. Several of these protocols, e.g. [9], [15], [22], are designed for poll-site voting which is impractical for random sample voting due to the small number of voters and the need to keep their identity anonymous.

Helios [1] is a remote voting protocol that satisfies verifiability but not receipt-freeness. Two voting protocols that, like *Alethea*, achieve both verifiability and receipt-freeness are *BeleniosRF* [7] and *Civitas* [10]. Voters cast encrypted ballots that are processed by homomorphic tallying in BeleniosRF and by mixnets in Civitas. In both protocols, it is universally verifiable that the ballots are tallied as recorded but nevertheless impossible for a voter to prove to an adversary how he voted. In order for these protocols to work, a voter must encrypt his ballot with the help of a machine.

For random sample voting, we must assume that a general purpose platform is compromised. This necessitates that the protocol specification contains separate roles for the human voter and his devices. However, many existing remote voting protocols do not make such a separation [7] or require that the voter's platform must be trusted [10].

We have therefore developed *Alethea* as new voting protocol separating the human voter role and his devices from the start and using [4] and [5] for guidance. This enabled us to examine what properties hold under the realistic assumption that a voter's platform is compromised while the voter himself is honest and casts his vote with the platform's help. In addition to a general purpose computing platform, we propose a specialized trusted device. This device only needs limited computing capabilities and need not be connected to the Internet. We argue that it is more realistic to trust such a device than a general purpose platform.

C. Security properties

We focus on definitions and formal models of standard security properties for voting protocols. *Delaune et al.* [12] introduce the first symbolic definition of receipt-freeness and

[5]The economic justification that decoy ballots are effective and stop vote buying is extremely subtle and does not necessarily hold [3].

coercion resistance in the applied pi calculus. Similarly to their work, we define privacy and receipt-freeness as observational equivalence properties. We model receipt-freeness by modifying the protocol such that one voter communicates all his secrets to the adversary in one system and only pretends to do so in the other system. To better understand different notions of privacy, receipt-freeness, and coercion resistance, *Dreier et al.* [14] consider a new family of privacy properties that includes attacks such as vote-copying and forced-abstention. They also model these properties in the applied-pi calculus.

A review of existing definitions of verifiability is presented by *Cortier et al.* [11]. They cast all definitions in the same framework to analyze and compare them. In particular, they analyze the definitions of individual and universal verifiability by *Kremer et al.* [17] which are similar to our definitions but defined in the applied pi calculus. *Kremer et al.* also define *eligibility verifiability* as the property that everyone can check that each vote in the final tally was cast by an eligible voter and no voter could vote more than once. Even though this property is defined with respect to all eligible voters, it is comparable to our definition of *universal verifiability of voter codes*, which ensures that all tallied votes were cast by sample voters.

VII. Conclusion

As new forms of democracy are an active research area in political economy, it is important to demonstrate the feasibility and limitations of protocols that support them from a security perspective. This work is a first step to better understand the formal properties of random sample voting.

We have introduced *Alethea*, the first random sample voting protocol that satisfies receipt-freeness. *Alethea* is also the first formally verified random sample voting protocol.

To verify *Alethea*, we formulated and proved new as well as standard security properties. In particular, we showed that the voting server cannot influence the selection of the sample group and that the selection is random and universally verifiable. Moreover, we established that the sample voters remain anonymous and we proved that *Alethea* satisfies end-to-end verifiability and receipt-freeness.

A premise of end-to-end verifiability is that the voters perform a number of verifications. With our new global properties we make the realistic assumption that many voters will skip some verification steps and we quantify the probability that manipulations are not detected under this assumption. If this probability is sufficiently small, the global property justifies that proving end-to-end verifiability in a possibilistic trace model is worthwhile. We have also defined and proved an analogous global property for the selection phase of *Alethea*.

As future work, we intend to build on *Alethea* to develop protocols for other alternative forms of democracy that require a random group, such as *Co-Voting* [16].

References

[1] Ben Adida. Helios: Web-based Open-audit Voting. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 335–348, Berkeley, CA, USA, 2008. USENIX Association.

[2] David Basin, Jannik Dreier, and Ralf Sasse. Automated Symbolic Proofs of Observational Equivalence. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1144–1155, New York, NY, USA, 2015. ACM.

[3] David Basin, Hans Gersbach, Akaki Mamageishvili, Lara Schmid, and Oriol Tejada. Election Security and Economics: It's All About Eve. In *Electronic Voting. E-Vote-ID 2017*, pages 1–20. Springer, 2017.

[4] David Basin, Saša Radomirović, and Michael Schläpfer. A Complete Characterization of Secure Human-Server Communication. In *28th IEEE Computer Security Foundations Symposium (CSF 2015)*, pages 199–213. IEEE Computer Society, 2015.

[5] David Basin, Saša Radomirović, and Lara Schmid. Modeling Human Errors in Security Protocols. In *29th IEEE Computer Security Foundations Symposium (CSF 2016)*, pages 325–340. IEEE Computer Society, 2016.

[6] Josh Cohen Benaloh and Moti Yung. Distributing the power of a government to enhance the privacy of voters (extended abstract). In *Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing*, pages 52–62, 1986.

[7] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1614–1625, New York, NY, USA, 2016. ACM.

[8] David Chaum. Random-Sample Voting. http://rsvoting.org/whitepaper/white_paper.pdf, Accessed: 2017-07-07.

[9] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-End Voter-Verifiable Optical-Scan Voting. *IEEE Security Privacy*, 6(3):40–46, May 2008.

[10] Michael R Clarkson, Stephen Chong, and Andrew C Myers. Civitas: Toward a Secure Voting System. In *2008 IEEE Symposium on Security and Privacy (S&P 2008)*, pages 354–368.

[11] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Mueller, and Tomasz Truderung. Sok: Verifiability Notions for E-Voting Protocols. In *IEEE Symposium on Security and Privacy, S&P 2016, San Jose, CA, USA, May 22-26, 2016*, pages 779–798.

[12] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *19th IEEE Computer Security Foundations Workshop (CSF 2006)*, pages 28–42. IEEE, 2006.

[13] Danny Dolev and Andrew C. Yao. On the Security of Public Key Protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[14] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. A Formal Taxonomy of Privacy in Voting Protocols. In *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*, pages 6710–6715. IEEE, 2012.

[15] Aleksander Essex, Jeremy Clark, Urs Hengartner, and Carlisle Adams. Eperio: Mitigating Technical Complexity in Cryptographic Election Verification. In *Proceedings of the 2010 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'10, pages 1–16. USENIX Association, 2010.

[16] Hans Gersbach. Co-voting Democracy. *Economics of Governance*, 18(4):337–349, 2017.

[17] Steve Kremer, Mark Ryan, and Ben Smyth. Election Verifiability in Electronic Voting Protocols. In *15th European Symposium on Research in Computer Security (ESORICS 2010)*, volume 10, pages 389–404. Springer, 2010.

[18] Oksana Kulyk, Stephan Neumann, Karola Marky, Jurlind Budurushi, and Melanie Volkamer. Coercion-Resistant Proxy Voting. In *ICT Systems Security and Privacy Protection*, pages 3–16. Springer, 2016.

[19] Ueli Maurer and Pierre Schmid. A Calculus for Secure Channel Establishment in Open Networks. In *European Symposium on Research in Computer Security*, pages 173–192. Springer, 1994.

[20] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *25th International Conference on Computer Aided Verification (CAV 2013)*, volume 8044 of *LNCS*, pages 696–701. Springer, July 2013.

[21] C. Andrew Neff. A Verifiable Secret Shuffle and Its Application to e-Voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, CCS '01, pages 116–125, New York, NY, USA, 2001. ACM.

[22] Peter YA Ryan, David Bismark, James A Heather, Steve A Schneider, and Zhe Xia. The prêt à voter verifiable election system. *IEEE transactions on information forensics and security*, 4(4):662–673, 2009.

[23] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *25th IEEE Computer Security Foundations Symposium (CSF 2012)*, pages 78–94, 2012.

[24] Alethea Tamarin Input Files. In Tamarin repository https://github.com/tamarin-prover/tamarin-prover, folder `examples/csf18-alethea`, 2018. Accessed: 2018-04-30.

APPENDIX

We give proofs for the lemmas in Sections III-D to IV-D.

A. Proof of complexity

Proof of Lemma 1. We consider each role's time complexity for the two protocol phases. Recall that n is the number of voters and m is the number of sample voters, where $m \leq n$.

Voters: H's time complexity is dominated by searching the lists $[y]$ and $[y_{SG}]$ in the selection phase and the list $[\mathit{code}]$ in the voting phase, which requires $O(n) + O(m) + O(m)$ operations and is dominated by $O(n)$.

Server: In the selection phase, S requires $O(n)$ time to compute a voter code for each voter and to send them to the devices and to *BB*. If *select* is realized as proposed in Section IV-D, it includes the permutation of a list of length n and the element-wise combination of two lists of length n. Thus the selection phase's time complexity is $O(n)$. In the voting phase, S has to process lists of length m, including unpairing all elements, decrypting all elements, and permuting the lists, which requires $O(m)$ steps. If *cp* is implemented as in [21], the generation of the zero knowledge proofs requires $O(m)$ steps. S's total time complexity is thus dominated by $O(n)$.

Auditors: In the selection phase, A computes the *select* function ($O(n)$) and receives lists of sizes n and m ($O(n)$). In the voting phase, A receives five lists of length m and two proofs. If *cp* is realized as in [21], the verifications of the zero knowledge proofs require $O(m)$ time. Finally, A has to test if the lists $[y']$ and $[y_{SG}]$ are equal, which are both of length m. This can be done by sorting then comparing the lists in $O(m \log(m))$ steps. A's total time complexity is thus $O(n) + O(m \log(m))$.

Bulletin Board: BB only receives and displays messages. The time complexity of the selection phase is dominated by the processing of the voter code list, $O(n)$, and of the voting phase by handling five lists of length m and to proofs of lengths $O(m)$. This results in $O(n)$ total time complexity.

Device, platform, and environment: In both phases, D, P, and E send, receive and compute a constant number of messages. Their total time complexity is $O(1)$.

Thus all roles have time complexity in $O(n)$, except for the auditor role, which has time complexity $O(n)+O(m \log(m))$. □

B. Proofs for selection phase

Proof of Lemma 4. Let $tr \in \mathit{TR}(\wp)$ be a trace. Since A is honest, it only performs a verification if it has previously received all necessary messages. Therefore

$$\begin{aligned}
&\mathit{verify}_{SG}(A, [y_{SG}] = \mathit{select}(r, [y]), \mathit{true}) \in tr \\
&\overset{(1)}{\Longrightarrow} [y_{SG}] = \mathit{select}(r, [y]) \wedge \mathit{recv}(A, [y_{SG}]) \in tr \wedge \\
&\qquad\qquad \exists e.\mathit{recv}(A, \langle e, [y] \rangle) \in tr \\
&\overset{(2)}{\Longrightarrow} \exists \mathit{BB}, e.\mathit{send}(\mathit{BB}, [y_{SG}]) \in tr \wedge \mathit{send}(\mathit{BB}, \langle e, [y] \rangle) \in tr \\
&\overset{(3)}{\Longrightarrow} \mathit{BB}_{SG}([y_{SG}]) \in tr \wedge \mathit{BB}_y([y]) \in tr.
\end{aligned}$$

Implication (2) holds as A is honest and only accepts the received values $[y_{SG}]$ and $[y]$ over an authentic channel and from *BB*. By the properties of an authentic channel, we conclude that *BB* has sent $[y_{SG}]$ and $[y]$. Implication (3) holds as *BB* is honest and only sends the values $[y_{SG}]$ and $[y]$ after the signals that these values are posted on the bulletin board. □

Proof of Lemma 5. By Lemma 2, for each voter who verifies the individual verifiability of the voter code, there exists a corresponding voter code on the bulletin board.

Consider a trace with n voters. By the lemma's hypothesis for each voter, the event that the voter code was manipulated is independent of the event that the voter code was verified. Hence, we compute the probability that no manipulation is detected when there are n voters, o manipulated voter codes, and a voters who verified their voter codes as follows.

$$\begin{aligned}
&\mathit{Prob}(Z | V(tr) = n, X(tr) = o, Y(tr) = a) \\
&\quad = \tfrac{n-o}{n} \cdot \tfrac{n-o-1}{n-1} \cdots \tfrac{n-o-(a-1)}{n-(a-1)} \leq (\tfrac{n-o}{n})^a
\end{aligned}$$

The inequality follows as $\frac{x-1}{y-1} < \frac{x}{y}$ when $1 < x < y$. □

C. Proofs for voting phase

Proof of Lemma 8. Let $tr \in \mathit{TR}(\wp)$ and suppose that $q = \mathit{verify}_{v1}(A, \mathit{VeqP}(\mathit{proof}_V, [\mathit{fst}(\mathit{code})], [pV]), \mathit{true}) \in tr \wedge \mathit{verify}_{v2}(A, [\mathit{fst}(pV)] = [v], \mathit{true}) \in tr$ holds. Then

$$\begin{aligned}
&\overset{(1)}{\Longrightarrow} q \wedge \exists [x], [r], k, \pi.\ [pV] = [\langle v, x \rangle] \wedge \pi[\mathit{fst}(\mathit{code})] = [\{pV\}_k^r] \\
&\overset{(2)}{\Longrightarrow} q \wedge \exists [x], [r], k, \pi.\ \pi[\mathit{fst}(\mathit{code})] = [\{v, x\}_k^r] \\
&\overset{(3)}{\Longrightarrow} q \wedge \exists [x], [r], [\mathit{sndcode}], k, \pi.\ \pi[\mathit{code}] = [\langle \{v, x\}_k^r, \mathit{sndcode} \rangle] \\
&\overset{(4)}{\Longrightarrow} \exists [pY], [y'], \mathit{proof}_Y. \\
&\qquad \mathit{recv}(A, \langle [\mathit{code}], [pV], [v], [pY], [y'], \mathit{proof}_V, \mathit{proof}_Y \rangle) \in tr \\
&\overset{(5)}{\Longrightarrow} \exists \mathit{BB}, [pY], [y'], \mathit{proof}_Y. \\
&\qquad \mathit{send}(\mathit{BB}, \langle [\mathit{code}], [pV], [v], [pY], [y'], \mathit{proof}_V, \mathit{proof}_Y \rangle) \in tr \\
&\overset{(6)}{\Longrightarrow} \mathit{BB}_c([\mathit{code}]) \in tr \wedge \mathit{BB}_v([v]) \in tr
\end{aligned}$$

Implication (1) holds because the verifications succeed as indicated by the third argument (true) in the verify signal: The list of votes corresponds to the list of the first elements of the pairs $[pV]$. This denotes a one-to-one correspondence between votes v and vote pairs pV. The verification of the proof proof_V ensures that each element of $[\mathit{fst}(\mathit{code})]$ corresponds to an encryption of a unique element of $[pV]$, but the lists can be permuted differently. (2) holds by the facts that each pair pV

corresponds to a unique vote v and each $\mathit{fst}(\mathit{code})$ encrypts a unique pair pV. Thus, each $\mathit{fst}(\mathit{code})$ corresponds to the encryption of a unique vote. (3) holds because for each vote there is also a unique *code* that consists of the $\mathit{fst}(\mathit{code})$ and some second part *sndcode*. In (4) we use the assumption that the auditor A is honest and follows its role specification. We conclude that A uses in the verifications only terms that it has previously received. (5) holds because A only accepts the receiving of these messages over an authentic channel from *BB*. By the properties of an authentic channel, *BB* must thus have sent these messages. (6) holds because *BB* is honest, so it only sends these messages after having included the same message terms in the signals.

We have thus verified universal verifiability of the tally. Note that none of the implications rely on the server's honesty. Therefore, the proof holds for a compromised server S. □

We only give a high level proof sketch of Lemma 9 as the proof is analogous to the proof of Lemma 8.

Proof Sketch of Lemma 9. Assume a trace $tr \in \mathit{TR}(\wp)$ and that the given verifications are true in the trace. Because the verifications hold, we conclude that each voter code y' is contained in a unique pair in $[pY]$ and that each such pair pY is encrypted in a unique element in $[\mathit{snd}(\mathit{code})]$. As in the proof of Lemma 8, we conclude from these properties that for each voter code y' there is a unique code $\langle \mathit{fstcode}, \{y', hY\}_k^{r'} \rangle \in [\mathit{code}]$ that contains it.

Moreover, we observe that an honest auditor A must have received the terms $[\mathit{code}], [pY], [y']$, and proof_Y authentically from *BB*. Therefore *BB* must have sent these terms. Also, because *BB* is honest, the same terms are on the bulletin board. Together, we conclude that the terms $[\mathit{code}]$ and $[y']$, as used in A's verification signals, are present on *BB* and have the required relation. □

Proof of Lemma 10. By Lemma 7, if a voter verifies the individual verifiability check, his ballot code is correctly included in the list of ballot codes on *BB*.

Consider a trace with m voters in the sample group. By the lemma's hypothesis, for each voter the event that his ballot was manipulated is independent of the event that it was verified. Hence, we compute the probability that no manipulation is detected when there are m voters, o ballots are manipulated, and a ballots are verified as follows.

$$\begin{aligned} &\mathit{Prob}(Z|V(tr) = m, X(tr) = o, Y(tr) = a) \\ &\quad = \tfrac{m-o}{m} \cdot \tfrac{m-o-1}{m-1} \cdots \tfrac{m-o-(a-1)}{m-(a-1)} \leq \left(\tfrac{m-o}{m}\right)^a \end{aligned}$$

The inequality follows as $\frac{x-1}{y-1} < \frac{x}{y}$ when $1 < x < y$. □

D. Proofs for sample group

Proof of Lemma 13. Let $[y]$ be a list and *SG* a m-element subset of $\mathit{set}([y])$. Thus there is a vector $[v] \in \{0,1\}^n$ of Hamming weight $m = |\mathit{SG}|$ such that $\mathit{SG} = \mathit{set}(\mathit{comb}([y], [v]))$.

We first prove that there exist at least $m!(n-m)!$ many choices of $\pi \in \mathcal{G}$ such that $\mathit{SG} = \mathit{set}(\mathit{comb}([y], \pi[v_0]))$. Since

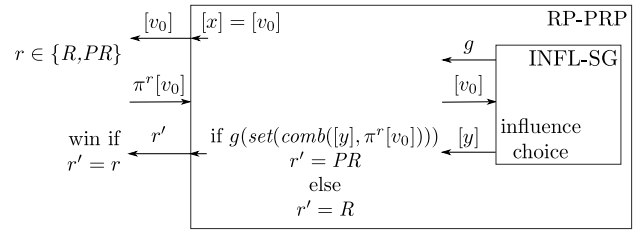

Fig. 7. Reduction from RP-PRP to INFL–SG

$\mathcal{G}$ contains all permutations of lists of length n, there exists $\pi \in \mathcal{G}$, such that $[v] = \pi[v_0]$. For each permutation π' that only permutes the ones and zeros of $[v_0]$, it also holds that $[v] = \pi\pi'[v_0]$. Since there are m many ones and $(n-m)$ many zeros in $[v_0]$, there are $m!(n-m)!$ many permutations π that map an input $[y]$ to *SG*.

We next prove that there are exactly $m!(n-m)!$ many choices of $\pi \in \mathcal{G}$ such that $\mathit{SG} = \mathit{set}(\mathit{comb}([y], \pi[v_0]))$. *SG* is a selection of m voters out of n voters, so there are $\binom{n}{m} = \frac{n!}{(n-m)!m!}$ many possible outputs *SG*. Since for each output *SG* there are at least $m!(n-m)!$ many choices of $\pi \in \mathcal{G}$ such that $\mathit{SG} = \mathit{set}(\mathit{comb}([y], \pi[v_0]))$ and there are $n!$ permutations in $\mathcal{G}$, it follows that for each *SG* there are exactly $m!(n-m)!$ many choices of $\pi \in \mathcal{G}$ such that $\mathit{SG} = \mathit{set}(\mathit{comb}([y], \pi[v_0]))$.

Since the $\pi \in \mathcal{G}$ are chosen uniformly at random it follows that the set of all possible outputs *SG* is a uniformly random m-element subset of $\mathit{set}([y])$. □

Proof of Lemma 14. Let $p_{PR} = \Pr[g = \mathit{true} \mid r = PR]$ and $p_R = \Pr[g = \mathit{true} \mid r = R]$ be the probability that the adversary wins the INFL–SG game given a PRP and an RP, respectively. Suppose the adversary's advantage in winning the INFL–SG game with a PRP over an RP is greater than ϵ, i.e., $|p_{PR} - p_R| > \epsilon$. We show that she can use this to distinguish an RP from a PRP in the RP–PRP game with an advantage greater than ϵ.

The reduction from the RP–PRP game to the INFL–SG game is shown in Figure 7. The adversary first outputs $[x] = [v_0]$ to the environment in the RP–PRP game. She then plays the INFL–SG. That is, she outputs g, takes $[v_0]$, and outputs $[y]$. Next, the adversary takes the input $\pi^r[v_0]$ from the environment in the RP–PRP game and the output $[y]$ from the INFL–SG game and computes $g(\mathit{set}(\mathit{comb}([y], \pi^r[v_0])))$. If the predicate g is satisfied, she outputs $r' = PR$ otherwise she outputs $r' = R$. In case $r = PR$, the adversary wins the inner game with probability p_{PR}. The output of the outer game is $r' = PR$ and the adversary wins the outer game in this case with the same probability. In case $r = R$, the adversary wins the inner game with probability p_R. Thus, with the same probability she incorrectly outputs $r' = PR$ in the outer game.

Thus her advantage in distinguishing an RP from a PRP is $|\mathit{Prob}[r' = PR \mid r = PR] - \mathit{Pr}[r' = PR \mid r = R]| = |\mathit{Prob}[g = \mathit{true} \mid r = PR] - \mathit{Pr}[g = \mathit{true} \mid r = R]| > \epsilon$. □