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

brought to you by CORE

provided by KITopen

Security Proofs for Participation Privacy and Stronger Verifiability for Helios

David Bernhard[1], Oksana Kulyk[2], Melanie Volkamer[2,3]

[1] University of Bristol, Bristol, United Kingdom
surname@cs.bris.ac.uk
[2] Technische Universität Darmstadt, Darmstadt, Germany
name.surname@secuso.org
[3] Karlstad University, Karlstad, Sweden

Abstract. The Helios voting scheme is well studied including formal proofs for verifiability and ballot privacy, but it does not provide participation privacy (i.e. it reveals who participated in the election). Kulyk, Teague and Volkamer proposed an extension to Helios that is claimed to provide ballot privacy as well as participation privacy while providing stronger verifiability than Helios. However, the authors did not prove their claims. Our contribution is to provide a formal definition for participation privacy and to prove that their claims hold.

1 Introduction

The Helios voting scheme [1] and its use in IACR elections has led to a large field of research into formal notions of security such as ballot privacy [7, 6] and verifiability [10, 20]. One open issue of Helios is a lack of particpation privacy (i.e. the public available election information should not reveal whether a honest voter cast a vote or abstained). Correpsondingly, no formal definition for participation privacy has been proposed, yet.

Kulyk, Teague and Volkamer proposed an extension to Helios [21] (henceforth referred to as KTV-Helios) that adds participation privacy to the Helios voting scheme while, at the same time, it still allows everyone to verify that only eligible voters cast a vote. KTV-Helios claims to achieve both security properties at the same time by introducing posting trustees who can cast dummy ballots for any voter, that do not contribute to the election result. Furthermore, KTV-Helios makes use of a public-key infrastructure (PKI): voters have signing keys and ballots contain zero-knowledge proofs that they have either been signed by an eligible voter or they contain a dummy vote. The authors of [21] did not prove that the KTV-Helios scheme actually provides the security properties it aims to achieve, i.e. participation privacy and verifiability as well as ballot privacy.

Our contributions. In order to evaluate these claims, we first provide a formal definition for probabilistic participation privacy. Furthermore, we apply the proposed definition to prove that KTV-Helios ensures probabilistic participation privacy and propose an algorithm to compute the adversarial advantage depending on the election parameters. In addition, we prove that KTV-Helios ensures ballot privacy according to the definition in [6] in the random oracle model. We further prove that the KTV-Helios scheme provides verifiability against malicious bulletin board based on the definition in [10], which is a stronger form of verifiability compared to the level of verifiability proven to hold for Helios in [10]. With this, the claims of the KTV-Helios security in [21] are evaluated and substantiated.

Verifiability: The system should provide for every honest[4] voter the possibility to verify that their ballot is properly stored on the bulletin board. It further should enable everyone to verify that only ballots from the eligible voters are included in the tally, and that each ballot cast by eligible voters on the bulletin board is correctly processed during tallying. These verifications should not require any security assumptions other that the register of eligible voters and the PKI is trustworthy.

Ballot privacy: Given the public data of the election (incl. the election result), the adversary should be incapable of gaining more information about an individual honest voter's vote than is leaked by the election result. This should not require further security assumptions othat that the following ones: (1) a majority of entities responsible for tallying does not divulge their secret key shares to the adversary; (2) the honest voter does not divulge private information used for encrypting her vote to the adversary; (3) the bulletin board acts according to its specification by not removing the ballots submitted to it.

Participation privacy: Given the public data of the election, the adversary should be incapable to tell, whether a given honest voter has cast her ballot in the election. Participation privacy should be ensured given only the following security assumptions: (1) the majority of entities responsible for the tallying do not divulge their secret key shares to the adversary, (2) the adversary is incapable of observing the communication channel between the voter, posting trustees and the voting system, (3) at least one of the posting trustees does not divulge private information to the adversary, (4) the bulletin board acts according to its specification, (5) The honest voters decide to participate or to abstain in the election independently from each other.

Overview of this paper. In Section 2 we provide a formal description of KTV-Helios. Next we provide definitions of existing security properties and prove them for KTV-Helios, verifiability in Section 3 and ballot privacy in Section 4. In Section 5 we define participation privacy and prove it for KTV-Helios, showing how to compute the privacy level for given election parameters.

2 Description of the KTV-Helios scheme

In this section, we provide a formal description of the KTV-Helios scheme. For this, we rely on the description in [21], while providing some additional specifications and minor modifications. As such, we specify the procedure for the posting trustees for determining the number and time of casting of dummy ballots for

[4] We refer to a voter as *honest*, if she is not under adversarial control, and *corrupted* otherwise.

each voter. An additional modification that we make is in the well-formedness proof for the ballot: in the original paper, the authors suggested that the voters prove the knowledge of their DSA secret key, if the DSA-based PKI is used for the election. This, however, can lead to practicability and security issues[5]. In order to avoid these issues we suggest that voters prove the knowledge of a signature on their encrypted vote instead by using the techniques outlined in [3].

2.1 Overview

The KTV-Helios scheme involves the following entities:

- *Election organizers*, responsible for setting up the bulletin board and publishing the general information for the election, incl. the voting options.
- *Registration authority*, responsible for keeping the list of eligible voters,
- *Bulletin board (*BB), acting as a reliable broadcast channel,
- *Posting trustees*, responsible for adding dummy ballots for the voters,
- *Tabulation tellers*, responsible for anonymising the ballots and decrypting the election result.

For the sake of simplicity, we assume a single tabulation teller and a single posting trustee. Then, the KTV-Helios scheme can be described as follows: *Before the election*, the PKI with digital signatures for the voters and voting system entities is established[6] by the registration authority, and the pair of asymmetric keys using additively homomorphic cryptosystem for the election is generated by the tabulation tellers. During the *voting phase*, the voter encrypts the chosen voting option and submits it to the bulletin board together with the accompanying well-formedness proof. In case she later wants to update her vote for option v with the vote for another voting option v', the voter submits another ballot that encrypts $v' - v$. The ballots, composed of the encrypted vote and proof, are published on the bulletin board next to the voter's name.

During the whole voting phase, the posting trustee also casts a number of dummy ballots on behalf of each voter, that are published next to that voter's name. Each dummy ballot consists of an encryption of 0 accompanied with the well-formedness proof, that is constructed in the same way as the proofs for non-dummy ballots. Before the tallying, for each voter the ballots that are published next to the voter's name are aggregated into the final ballot. Due to the homomorphic property of the cryptosystem, and due to the fact that the dummy ballots contain the encryption of 0, this final ballot encrypts the sum of all non-dummy votes cast by the voter. The final ballots of all voters are being further anonymised via shuffling. Afterwards, each anonymised ballot is assigned to a valid voting option, or discarded without revealing its plaintext value.

[5] If secure hardware, such as a tamper-resistant smartcard is used for implementing the PKI, this hardware either has to be reprogrammed for the election in order to use the secret key for the proofs, or simply divulge the secret key to the voting software.

[6] Alternatively, an existing PKI can be used, such as eID smartcards.

2.2 Building blocks of the KTV-Helios scheme

We now describe the building blocks of the KTV-Helios scheme in more details. The scheme uses the following cryptographic primitives:

- Signed ElGamal [7], a NM-CPA secure encryption scheme (the same one is used in Helios). Its algorithms are $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$. The encryption of a message $m \in \mathbb{Z}_q$ with a public key $(g, h) \in \mathbb{G}^2$ is $((g^r, g^m h^r), \pi_{PoK})$ where $r \leftarrow_{\$} \mathbb{Z}_q$ is randomly sampled and π_{PoK} is a Schnorr proof of knowledge of r. To decrypt a ciphertext $((c^{(1)}, c^{(2)}), \pi_{PoK})$ with a secret key sk, first check the PoK and if successful set $m = c^{(2)} \cdot (c^{(1)})^{(-sk)}$.
- An existentially unforgeable digital signature scheme consisting of algorithms $\mathsf{SigKeyGen}$, Sign and Verify, for example Schnorr signatures.
- The Chaum-Pedersen NIZK proof $\mathsf{EqProof}(g_1, g_2, h_1, h_2)$ that proves the equality of discrete logarithms $\log_{g_1} h_1 = \log_{g_2} h_2$ as described in [8]. This proof can be simulated in the random oracle model, for which we write $\mathsf{SimEqProof}(g_1, g_2, h_1', h_2')$ (see e.g. [6]).
- A NIZK disjunctive proof $\mathsf{DisjProof}(\mathsf{pk}_{id}, \mathsf{sk}_{id'} \in \{\mathsf{sk}_{id}, 0\}, g_1, g_2, h_1, h_2, t)$ that given $(\mathsf{pk}_{id}, \mathsf{sk}_{id}) \leftarrow_{\$} \mathsf{SigKeyGen}$ and $g_1, g_2, h_1, h_2 \in \mathbb{G}_q$ and timestamp t proves either the knowledge of $s = \mathsf{Sign}(\mathsf{sk}_s, g_1 || g_2 || h_1 || h_2 || t)$[7], or the equality of discrete logarithms $\log_{g_1} h_1 = \log_{g_2} h_2$.
- A re-encryption mix-net for ElGamal ciphertexts $\mathsf{Mix}(c_1, ..., c_N)$, for example the one of Wikström and Terelius [29].
- A plaintext equivalence test (PET) to decrypt ElGamal ciphertexts. On input a ciphertext c, a secret key sk and a message m it creates a decryption factor d that is 1 if c is an encryption of m under sk and random in $\mathbb{Z}_q$ if not. It also creates a proof π_{PET} that it operated correctly (this is another Chaum-Pedersen $\mathsf{EqProof}$).

The task of the **posting trustee** is to cast a random number of dummy ballots at random times next to each voter's *id*. In order to specify the dummy ballot casting algorithm for the posting trustee, we use two probability distributions $\mathbb{P}_{dummy}$ and $\mathbb{P}_t$. The first probability distrubition $\mathbb{P}_{dummy}$ is used to sample a number of dummy ballots for each voter. This distribution therefore has a support $[x, y]$ with x, y as the minimal and maximal number of dummy ballots that the posting trustee is going to cast for each voter (i.e., $x \in \mathbb{N}_0$, $y \in \mathbb{N}_o \cup \{\infty\}$). The parameters x and y, as well as the exact $\mathbb{P}_{dummy}$ needs to be defined by the election authorities when setting up a corresponding system, i.e. their optimal trade-off between security and efficiency. For further information which influence the selection of $\mathbb{P}_{dummy}$ has to the level of security and the efficiency of the tallying algorithms, see Section 5.2. The second probability distribution $\mathbb{P}_t$ is used to determine the time to cast each dummy ballot. Thus, this distribution has a support $[T_s, T_e]$ with T_s denoting the timestamp at the

[7] Methods for proving the knowledge of a digital signatures via Σ-proof are described by Asokan et al. [3] for common signature schemes; the general method of constructing NIZK disjunctive proofs is described by Cramer et al. in [13].

start of the voting phase, and T_e the timestamp at the end of the voting phase. In order to obfuscate the ballots cast by voters, $\mathbb{P}_t$ should be chosen so that this distribution resembles the distribution of times at which the voters cast their ballots. For this, e.g. the information from the previous elections could be used.

The plaintext tally function of the KTV-Helios scheme, that takes the plaintext votes cast by voters and the posting trustee as input and outputs the election result, is informally described in the following way: The valid votes cast by registered eligible voters are included in the tally. If the voter casts multiple votes, they are added together to form a final vote before the final tally if the result of this addition is a valid voting option, or replaced with a null vote otherwise. If the voter abstains, their final vote is counted as a null vote[8]. The votes cast by the posting trustee are not included in the result.

The formalised description of the plaintext tally function is as follows: Let $\mathbb{G}_q$ be the plaintext space of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. Then, let $\mathbb{V}_{valid} = \{o_1, ..., o_L\} \subset \mathbb{G}_q^L$, $0 \notin \mathbb{V}_{valid}$ be a set of valid voting options, so that the voter is allowed to select one of these options as her vote. Let then $\rho' : (\mathbb{V}_{valid} \cup \{0\})^N \rightarrow \mathbb{N}_0^L$ be the function that, given the plaintext votes cast within the election, outputs a vector of values with the sum of cast votes for each candidate and the number of abstaining voters. Let $I = \{id_1, ..., id_N\}$ be a set of registered eligible voters, and $\hat{id} \notin I$ denote the posting trustee. Further, let N_T be the total number of votes cast within the election. We define the tally function for the KTV-Helios scheme $\rho(\mathbb{V}_{cast}) : (I \cup \{\hat{id}\} \times G_q)^* \rightarrow \mathbb{R}$ as follows:

1. Initialise a set $\mathbb{V}_{final} = \{(id_1, 0), ..., (id_N, 0)\}$
2. For every $(id, v) \in \mathbb{V}_{cast}$, if $id \in I$, replace the tuple $(id, v') \in \mathbb{V}_{final}$ with $(id, v' + v)$. If $id = \hat{id}$, discard the vote.
3. For every $(id_i, v_i) \in \mathbb{V}_{final}$, if $v_i \notin \mathbb{V}_{valid}$, replace (id_i, v_i) with $(id_i, 0)$
4. Output $\rho'(v_1, ..., v_N)$.

The function ρ provides *partial counting* defined as follows: Given the sets I_1,...,I_k that partition $I \cup \{\bar{id}\}$, define lists $\mathbb{V}_{cast}^{(1)}, ..., \mathbb{V}_{cast}^{(k)} \subset \mathbb{V}_{cast}$ so that for each $(id, v) \in \mathbb{V}_{cast}$ holds $(id, v) \in \mathbb{V}_{cast}^{(i)} \iff id \in I_i$, $i = 1, ..., k$. Then it holds:

$$\rho(\mathbb{V}_{cast}) = \sum_{i=1}^{k} \rho(\mathbb{V}_{cast}^{(i)})$$

2.3 Formal Description of KTV-Helios

We are now ready to provide the specification of the functions that constitute the formal description of the KTV-Helios scheme. This description is based upon the syntax proposed in [6], adjusted where needed to the context of the KTV-Helios scheme.

[8] Note, that the function does not make a distinction between abstaining voters, and voters that cast a null vote.

- $\mathsf{Register}(1^\lambda, id)$ is run by the registration authority. Given a register I of eligible voters, the function returns a pair of keys $(\mathsf{pk}_{id}, \mathsf{sk}_{id}) \leftarrow_\$ \mathsf{SigKeyGen}(1^\lambda)$ if $id \in I$ and adds (id, pk_{id}) to the list of registered voters' public keys I_{pk}.
- $\mathsf{Setup}(1^\lambda)$ is run by the tabulation teller. It runs $(\mathsf{pk}, \mathsf{sk}) = \mathsf{KeyGen}$ to create the election keys and returns the public key pk.
- $\mathsf{Vote}((id', \mathsf{sk}_{id'}), id, v, t)$ creates a ballot $b = (id, c, \pi_{PoK}, \pi, t)$ for voter $id \in I$ and voting option v, that is cast at a timestamp[9] t. If $id = id'$ (a voter casting her own ballot) then it computes $(c, \pi_{PoK}) = \mathsf{Enc}(\mathsf{pk}, v)$ where $c = (c^{(1)}, c^{(2)})$ and $\pi = \mathsf{DisjProof}(\mathsf{pk}_{id}, \mathsf{sk}_{id'}, g, h, c^{(1)}, c^{(2)}, t)$ using a signature $\mathsf{Sign}(\mathsf{sk}_{id'}, g||h||c||t)$. If $id' = \hat{id}$ (the posting trustee is casting a ballot on behalf of voter id) then $\mathsf{sk}_{id'}$ is not required but v must be 0.
- $\mathsf{Validate}(b)$ parses the ballot b as $(id, c = (c^{(1)}, c^{(2)}), \pi_{PoK}, \pi, t)$ and returns 1 if π and π_{PoK} are valid proofs, $id \in I$ and $t \in [T_s, T_e]$, and $\perp$ otherwise.
- $\mathsf{VerifyVote}(\mathsf{BB}, b)$ is used by the voter to ensure that her ballot b is properly stored on the bulletin board. It outputs 1 if $b \in \mathsf{BB}$ and $\mathsf{ValidateBB}(\mathsf{BB})$ holds, otherwise $\perp$.
- $\mathsf{VoteDummy}(id)$ is used by the posting trustee to cast dummy ballots for a given voter id. The posting trustee samples a random number $m \leftarrow_\$ \mathbb{P}_{dummy}$ and random timestamps $t_1, ..., t_m \leftarrow_\$ \mathbb{P}_t$, and returns a set of ballots

$$(\mathsf{Vote}((\hat{id}, 0), id, 0, t_1), ..., \mathsf{Vote}((\hat{id}, 0), id, 0, t_m))$$

- $\mathsf{Valid}(\mathsf{BB}, b)$ is run by the board before appending a new ballot. It checks that $\mathsf{Validate}(b) = 1$ and that the ciphertext c in b does not appear in any ballot already on the board. If this holds it returns 1, otherwise $\perp$.
- $\mathsf{ValidateBB}(\mathsf{BB})$ checks that a board is valid. It is run by the tabulation tellers as part of the tallying process and by voters verifying the board. It creates an empty board B' and for each ballot $b \in \mathsf{BB}$ runs "if $\mathsf{Valid}(B', b)$ then append b to B'". If any ballot gets rejected it returns $\perp$, otherwise 1.
- $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk})$ is used by the tabulation teller to calculate the election result. It returns a tuple (R, Π) where R is the election result and Π is auxiliary data (proofs of correct tallying). In more detail:

  1. Run $\mathsf{ValidateBB}(\mathsf{BB})$ and return $\perp$ if this fails.
  2. Parse each ballot $b \in \mathsf{BB}$ as $(id, c, \pi_{PoK}, \pi, t)$.
  3. For each id appearing in the ballots, set $c_{id} = \prod_{c \in C(id)} c$ where $C(id)$ is the set of ciphertexts c in ballots belonging to voter id.
  4. Mix the ballots $(c_1, \ldots, c_N)$ (where N is the number of distinct identities who cast a ballot) to get a new list of ballots $(\bar{c}_1, \ldots, \bar{c}_N)$ and a proof π_{mix} of correct mixing.
  5. For each $i \in \{1, \ldots, N\}$ and each valid voting option $v \in \mathbb{V}_{valid}$, use the PET to create a decryption factor $d_{i,v}$ and proof $\pi_{PET,i,v}$.
  6. The result R is the number of times each voting option was chosen, i.e. $R(v) = |\{i : d_{i,v} = 1\}|$ for all $v \in \mathbb{V}_{valid}$. The auxiliary data Π

[9] As the timestamp t denotes the time at which b is submitted to the bulletin board, we assume that it is chosen in $[T_s, T_e]$.

contains the mixing proofs π_{mix}, the mixed ciphertexts $(\bar{c}_1, \ldots, \bar{c}_N)$, the decryption factors $d_{i,v}$ and the PET proofs $\pi_{PET,i,v}$ for $i \in \{1, \ldots, N\}$ and $v \in \mathbb{V}_{valid}$.

- $\mathsf{ValidateTally}(\mathsf{BB}, (R, \Pi))$ takes a bulletin board BB and the output (R, Π) of Tally and returns 1 if $\mathsf{ValidateBB}(\mathsf{BB}) = 1$ and all the proofs π_{mix} and π_{PET} are valid, otherwise $\perp$. It is used to verify an election.

The election is then organized as follows:

Setup phase: The election organizers set up an empty bulletin board BB and publish a set of valid non-null voting options $\mathbb{V}_{valid} = (v_1, ..., v_L)$ with $0 \notin \mathbb{V}_{valid}$. If there is no existing PKI encompassing the eligible voters, the eligible voters from the voting register I are being registered via the registration authority running $\mathsf{Register}(1^\lambda, id)$ for each voter and publishing the list of registered voters $I_{pk} = \{(id_1, \mathsf{pk}_{id_1}), ..., (id_N, \mathsf{pk}_{id_N})\}$. The tabulation teller runs $\mathsf{Setup}(1^\lambda)$.

Voting phase: The posting trustee runs $\mathsf{VoteDummy}(id)$ for each registered eligible voter $id \in I$. The posting trustee then submits each resulting dummy ballot $b = (id, c, \pi_{PoK}, \pi, t)$ to the bulletin board at a time corresponding to the timestamp t. The bulletin board appends b to BB. The voter id runs $\mathsf{Vote}((id, \mathsf{sk}_{id}), id, v, t)$ in order to cast her ballot for a voting option v at a time denoted by timestamp t. The bulletin board appends b to BB. Then, the voter can run $\mathsf{VerifyVote}(\mathsf{BB}, b)$ to check whether her ballot is properly stored.

Tallying phase: The tabulation teller runs $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk})$ on the contents of the bulletin board, and publishes the resulting output (R, Π). Everyone who wants to verify the correctness of the tally runs $\mathsf{ValidateTally}(\mathsf{BB}, (R, \Pi))$.

3 Verifiability

Our goal was to prove that the scheme provides verifiability, i.e. cast as intended and stored as cast verifiability is provided for every honest voter; and that everyone can verify that only ballots from the eligible voters are included in the tally, and that each ballot cast by eligible voters is correctly tallied. It is hence required, that a successful verification ensures, that the tally result consists of the ballots of all the honest voters who run $\mathsf{VerifyVote}(\mathsf{BB}, b)$, a subset of ballots of honest voters who did not do this, and a subset of ballots of voters corrupted by the adversary. Note, we accept the following assumptions: The register of eligible voters and the PKI is trustworthy. Furthermore, honest voters' secret keys are not leaked to the adversary

For the actual proof, we rely on the 'verifiability against a malicious bulletin board' framework definition for Helios alike schemes of [10] which matches the verifiability definition in our introduction. We adjust the definition in [10] to the KTV-Helios scheme by applying the following experiment $\mathsf{Exp}^{\mathsf{ver-b}}_{\mathcal{A},\mathcal{S}}$: The challenger runs the setup phase as outlined in Section 2.3 on behalf of the election organizer and registration authority. The tabulation teller, which might be controlled by the adversary, runs $\mathsf{Setup}(1^\lambda)$. The challenger further initialises an empty set I^C and HVote, which would correspond to the set of corrupted voters and to the votes cast by honest voters correspondingly. The adversary is given access to the following queries:

- $O\mathsf{Cast}(b)$: appends the ballot b to the bulletin board BB.
- $O\mathsf{Vote}(id', id, v, t)$: If $id \in I \cup \{\hat{id}\}$ and $id \notin I^C$, appends b to BB where $b = \mathsf{Vote}((id', \mathsf{sk}_{id'}), id, v, t)$, and adds a tuple (id', v, b) to HVote. Note, that as opposed to the definition in [10], the tuples $(id', *, *)$ already present in HVote are not removed, since the tally function takes all the valid cast ballots as the input. Otherwise, the query returns $\perp$.
- $O\mathsf{Corrupt}(id)$: if called for a corrupt voter identity id $\in I^C$, the oracle immediately returns $\perp$.Otherwise, it adds id to I^C and returns the voter's secret key sk_{id} to the adversary. In contrast to the definition in [10], we require that each tuple $(id, *, *) \in \mathsf{HVote}$ is removed from HVote, meaning that the previous ballots cast for the voter id using the $O\mathsf{Vote}$ query no longer count as ballots of an honest voter.

In addition to these queries, the adversary also has the capabilities of adding, modifying and removing the ballots on the bulletin board. Additionally, a set of voters $\mathsf{Checked} \subset I$ is defined, so that for each query $O\mathsf{Vote}(id, id, v, t)$, it is assumed that the corresponding voter $id \in \mathsf{Checked}$ has run $\mathsf{VerifyVote}(\mathsf{BB}, b)$ on the resulting ballot at the end of the election, and complained to the authorities in case the verification result was negative. At the end of the experiment, the adversary produces the tally output (R, Π). The experiment outputs $\mathsf{Exp}^{\mathsf{ver-b}}_{\mathcal{A},\mathcal{S}} = 0$ if one of the following cases holds:

- There were no manipulation, i.e. the output result R corresponds to the votes from honest voters who checked that their ballot is properly stored on the bulletin board, a subset of votes from honest voters who did not perform this check, and a subset of votes from corrupted voters: i.e.

$$\begin{aligned} R = {} & \rho((id_{E,1}, v_{E,1}), ..., (id_{E,n_E}, v_{E,n_E})) \\ & + \rho((id_{A,1}, v_{A,1}), ..., (id_{A,n_A}, v_{A,n_A})) \\ & + \rho((id_{B,1}, v_{B,1}), ..., (id_{B,n_B}, v_{B,n_B})) \end{aligned}$$

  holds; while the list of tuples $(id_{E,i}, v_{E,i})$ were cast by honest voters (i.e. $(id_{E,i}, v_{E,i}, *) \in \mathsf{HVote}$ for all $i = 1, ..., n_E$) who verified that their ballot is properly stored on the bulletin board (i.e. $id_{E,i} \in \mathsf{Checked}$ for all $i = 1, ..., n_E$); the list of tuples $\{(id_{A,1}, v_{A,1}), ..., (id_{A,n_A}, v_{A,n_A})\}$ were cast by honest voters (i.e. $(id_{A,i}, v_{A,i}, *) \in \mathsf{HVote}$ for all $i = 1, ..., n_A$) but who did not verify (i.e. $id_{A,i} \notin \mathsf{Checked}$ for all $i = 1, ..., n_A$); and the list of tuples $\{(id_{B,1}, v_{B,1}), ..., (id_{B,n_B}, v_{B,n_B})\}$ represents those votes cast by the adversary so that the number of unique IDs in a list $\{id_{B,1}, ..., id_{B,n_B}\}$ is at most the number of corrupted voters $|I^C|$.
- A manipulation was detected, i.e either there were complains from the voters who run the $\mathsf{VerifyVote}$ check with $\mathsf{VerifyVote}(\mathsf{BB}, b) = \perp$, or the tally output does not pass the validity check: $\mathsf{ValidateTally}(BB, (R, \Pi)) = 0$.

The experiment $\mathsf{Exp}^{\mathsf{ver-b}}_{\mathcal{A},\mathcal{S}}$ serves as a basis for the definition of verifiability[10] against a malicious bulletin board.

Definition 1. *A voting scheme $\mathcal{S}$ ensures verifiability, if the success probability* $\Pr\left[\mathsf{Exp}^{\mathsf{ver-b}}_{\mathcal{A},\mathcal{S}} = 1\right]$ *is negligible for any PPT adversary.*

We are now ready to prove the verifiability against a malicious bulletin board for the KTV-Helios scheme.

Theorem 1. *The voting scheme defined in Section 2.3 provides verifiability against a malicious bulletin board.*

The proof is based on similar ideas as the proof in [10]. We provide the proof in Appendix C.

4 Ballot privacy

In this section we prove ballot privacy (BPRIV) for the KTV-Helios scheme following the defintion in [6]. Since the original definition also uses two auxiliary properties called strong correctness and strong consistency, we prove these as well. Together these definitions imply that an adversary does not get more information from an election scheme as they would from the election result alone. Put differently, the election data — ballots on the board, proofs of correctness, proofs of correct tallying — do not leak any information about the votes. We assume like in [6] that both the tabulation teller and the bulletin board are honest, which corresponds to our informal definition in the introduction of this paper.

4.1 Purpose and Definition of BPRIV

We adjust the definition proposed by Bernhard et al. [6] – more precisely the definition in the random oracle model – to the KTV-Helios scheme by including additional parameters required for casting a ballot. We also omit the $\mathsf{Publish}$ algorithm as our boards do not store any non-public data (our $\mathsf{Publish}$ would be the identity function). Recall that a scheme satisfies BPRIV [6] if there exists an algorithm $\mathsf{SimProof}$ such that no adversary has more than a negligible chance of winning the BPRIV game; the game itself uses the $\mathsf{SimProof}$ algorithm in the tallying oracle.

The purpose of BPRIV is to show that one does not learn anything more from the election data (including the bulletin board and any proofs output by the

[10] Note, that the definition can be further cast into the verifiability framework by Kuesters, Trudering and Vogt [23] in order to enable uniform treatment of verifiability. The casting of the definition by Cortier et al. [10] has been described in [11]. Since the scheme and the security in [10] are similar to the KTV-Helios scheme and the definition of verifiability used in this paper, the casting into the framework can also be done in a similar manner.

tallying process) than from the election result alone. In other words, the election data does not leak information about the votes, at least in a computational sense[11]. For example, if Alice, Bob and Charlie vote in an election and the result is "3 yes" then the result alone implies that Alice must have voted yes, which is not considered a privacy breach. But if Charlie votes yes and the result is "2 yes, 1 no" then Charlie should not, without any further information, be able to tell whether Alice voted yes or no as this does not follow from the result.

The BPRIV notion is a security experiment with two bulletin boards, one of which (chosen at random by sampling a bit β) is shown to the adversary. For each voter, the adversary may either cast a ballot themselves or ask the voter to cast one of two votes v_0, v_1 in which case a ballot for v_0 is sent to the first board and a ballot for v_1 is sent to the second board. The adversary thus sees either a ballot for v_0 or a ballot for v_1 and a scheme is BPRIV secure if no PPT adversary has better than a negligible chance of distinguishing the two cases. At the end of the election, the adversary is always given the election result for the first board. This disallows trivial wins if the adversary makes the results on the two boards differ from each other. If the first board was the one shown to the adversary, it is tallied normally; if the adversary saw the second board but the first result then the experiment creates fake tallying proofs to pretend that the second board had the same result as the first one. This is the role of the $\mathsf{SimProof}$ algorithm that must be provided as part of a BPRIV security proof.

The experiment $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A},\mathcal{S}}$ for the scheme $\mathcal{S}$ is formally defined as follows: The challenger sets up two empty bulletin boards BB_0 and BB_1, runs the setup phase as outlined in Section 2.3 and publishes the election public key pk. The challenger also chooses a random $\beta \in \{0, 1\}$. The adversary can read the board BB_β at any time and can perfomr the following oracle queries:

- $O\mathsf{Cast}(b)$: This query lets the adversary cast an arbitrary ballot b, as long as b is valid for the board BB_β that the adversary can see. If $\mathsf{Valid}(\mathsf{BB}_\beta, b) = 1$, the challenger runs $\mathsf{Append}(\mathsf{BB}_0, b)$ and $\mathsf{Append}(\mathsf{BB}_1, b)$ to append the ballot b to both bulletin boards.
- $O\mathsf{VoteLR}(id', id, v_0, v_1, t)$: This lets the adversary ask a voter to vote for either v_0 or v_1 depending on the secret β. First, if $id \in I$ and $id' = id$ the challenger computes $b_0 = \mathsf{Vote}((id, \mathsf{sk}_{id}), id, v_0, t)$ and $b_1 = \mathsf{Vote}((id, \mathsf{sk}_{id}), id, v_1, t)$. If $id \in I$ and $id' = \hat{id}$ then the challenger computes two[12] ballots $b_0 = \mathsf{Vote}((id', \mathsf{sk}_{id'}), id, 0, t)$ and $b_1 = \mathsf{Vote}((id, \mathsf{sk}_{id}), id, 0, t)$. If none of these cases applies, the challenger returns $\perp$.
  Secondly, the challenger checks if $\mathsf{Valid}(\mathsf{BB}_\beta, b_\beta) = 1$ and returns $\perp$ if not. Finally the challenger runs $\mathsf{Append}(\mathsf{BB}_0, b_0)$ and $\mathsf{Append}(\mathsf{BB}_1, b_1)$.

[11] In an information-theoretic sense, an encrypted ballot does of course contain information about a vote, otherwise one could not tally it. But since ballots are encrypted, they should not help anyone who does not have the election secret key to discover the contained vote.

[12] Vote is a randomised algorithm so the effect of calling it twice on the same inputs is to create two distinct ballots.

– $O\mathsf{Tally}()$: The adversary calls this to end the voting and obtain the results. They may call this oracle only once and after calling it, the adversary may not make any more $O\mathsf{Cast}$ or $O\mathsf{VoteLR}$ calls.
  The challenger computes a result and auxiliary data for BB_0 as $(R, \Pi) = \mathsf{Tally}(\mathsf{BB}_0, \mathsf{sk})$. If $\beta = 1$, the challenger also computes simulated auxiliary data for BB_1 as $\Pi = \mathsf{SimProof}(\mathsf{BB}_1, R)$, overwriting the previous auxiliary data Π. The challenger then returns (R, Π) to the adversary.

At the end, the adversary has to output a guess $g \in \{0, 1\}$. We say that the adversary wins an execution of the experiment if $g = \beta$.

Definition 2. *A voting scheme $\mathcal{S}$ satisfies ballot privacy (BPRIV) if there exists a PPT simulation function* $\mathsf{SimProof}(\mathsf{BB}, R)$ *so that for any PPT adversary the quantity*

$$\mathsf{Adv}^{\mathsf{bpriv}}_{\mathcal{A},\mathcal{S}} := \left| \Pr\left[\mathsf{Exp}^{\mathsf{bpriv},0}_{\mathcal{A},\mathcal{S}} = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{bpriv},1}_{\mathcal{A},\mathcal{S}} = 1\right] \right|$$

is negligible (in the security parameter).

4.2 Proof for the KTV-Helios Scheme

The core of a BPRIV proof is a simulator $\mathsf{SimTally}$ that, when $\beta = 1$, takes as input the board BB_1 and the result R from BB_0 and outputs simulated data Π that the adversary cannot distinguish from real auxiliary data, such as proofs of correct tallying. This proves that the auxiliary data Π does not leak any information about the votes, except what already follows from the result.

Recall that the tallying process in KTV-Helios is as follows:

1. Remove any invalid ballots from the board using $\mathsf{ValidateBB}$.
2. Homomorphically aggregate the ballots from each voter.
3. Shuffle the remaining ballots (one per voter) in a mix-net.
4. Match each shuffled ballot against each valid vote $v \in V$ with a PET.
5. Compute the number of voters who chose each vote $v \in V$ by counting the successful PETs. This gives the election result R.
6. The auxiliary data Π comprises the proofs of correct mixing Π_{mix} from stage 3 and the data and proofs Π_{PET} forming the PETs in stage 4.

The additional PET stage compared to (non-KTV) Helios actually makes the ballot privacy proof easier. The simulator $\mathsf{SimProof}(\mathsf{BB}, R)$ works as follows:

1. Remove any invalid ballots from the board BB using $\mathsf{ValidateBB}$.
2. Homomorphically aggregate the ballots from each voter.
3. Shuffle the remaining ballots (one per voter) in a mix-net. Note, we do not need to simulate the mix-net; we can just run a normal mix (and store the auxiliary data Π_{mix} that this creates).
4. Simulate the PETs (we will describe this in detail below) to get simulated data Π'_{PET}.

5. Return (Π_{mix}, Π'_{PET}).

The following lemma is useful to construct the PET simulator.

Lemma 1. *In any execution of the BPRIV game, if we tallied both boards then with all but negligible probability, both boards would end up with the same number of ballots.*

Note that both the $O\mathsf{VoteLR}$ and the $O\mathsf{Cast}$ oracles either add one ballot to both boards each or do not add any ballots at all. Therefore we have the invariant that the number of ballots before tallying is the same on both boards with probability 1.

The first stage of the tallying algorithm runs $\mathsf{ValidateBB}$ to remove possibly invalid ballots. On the visible board BB_β, since all ballots were already checked in the oracles before placing them on the board, we conclude that $\mathsf{ValidateBB}$ does not remove any ballots. On the invisible board $\mathsf{BB}_{(1-\beta)}$, if any ballot b gets removed then we consider the query (VoteLR or Cast) where it was created. The only way a ballot b can get removed again is if at the time it was added, it was valid on BB_β (or it would never have got added at all) but invalid on $\mathsf{BB}_{(1-\beta)}$ (or it would not get removed again later). But this means that the ciphertext c in the ballot b in question must be a copy of an earlier ciphertext on $\mathsf{BB}_{(1-\beta)}$ but not on BB_β, as this is the only other case when Valid declares a ballot invalid, and the only such ballots are those created by $O\mathsf{VoteLR}$. Therefore we conclude that either two ballots created by $O\mathsf{VoteLR}$ have collided, the probability of which is certainly negligible, or the adversary has submitted in a $O\mathsf{Cast}$ query a copy of a ciphertext that $O\mathsf{VoteLR}$ previously placed on the invisible board $\mathsf{BB}_{(1-\beta)}$. Since the adversary never saw this ciphertext, and since the encryption scheme is NM-CPA so ciphertexts must be unpredictable, the probability of this event is negligible too. This concludes the proof of Lemma 1.

We now describe how to simulate the PET. Our inputs are a number n of ballots (the output of the mix-net), a result R that was correctly computed on a different board that also had n ballots (after stage 1 of tallying) by Lemma 1 and a set V of valid votes.

Since the PETs in a real tally are taken over ballots that have just come out of a mix-netm the distribution of votes in these ballots is a uniformly random permutation of votes subject to the tally being R. For example, if R indicates that there was one vote for $v = 1$ and $n - 1$ votes for $v = 2$ then the probability of the 1-vote being in the i-th ballot is $1/n$, irrespective of the order in which the ballots were cast (for example the adversary might know that the first person to vote was the one that cast the 1-vote). This is because the ballots are uniformly permuted in the mix-net.

Our simulation strategy is therefore to emulate this random permutation. The result R gives us a mapping $f_R : V \cup \{\bot\} \to \{0, 1, \ldots, n\}$ where for example $f_R(v) = 3$ means that three voters voted for v and $f_R(\bot)$ is the number of voters who cast an invalid vote. We have $f_R(\bot) + \sum_{v \in V} f_R(v) = n$, i.e. the number of invalid votes plus the totals for each valid option sum to the number n of ballots that came out of the mix-net. We simulate as follows:

1. Create a list $L = (L_1, \ldots, L_n)$ such that each vote $v \in V$ appears $f_R(v)$ times in L and the symbol $\perp$ appears $f_R(\perp)$ times. Then permute L randomly.
2. Create an $n \times |V|$ matrix d of PET results: if $L[i] = v$, which means that we pretend voter i voted for $v \in V$, then set $d_{i,v} = 1$. Otherwise set $d_{i,v}$ to be a random element of $\mathbb{Z}_q$.
3. For each (i, v) pair create a simulated PET proof as follows. For each ciphertext $c_i = (c_i^{(1)}, c_i^{(2)})$ and each valid voting option $v \in V$ pick a random $r_{i,v} \leftarrow_{\$} \mathbb{Z}_q$ and set $s_{i,v} = ((c_i^{(1)})^r, (c_i^{(2)}/v)^r)$. Then compute proofs
$$\pi_{i,v} = \mathsf{SimEqProof}(g, s_{i,v}^{(1)}, h, s_{i,v}^{(2)}/d_{i,v}) \cup \mathsf{EqProof}(c_i^{(1)}, c_i^{(2)}/v, s_{i,v}^{(1)}, s_{i,v}^{(2)})$$
4. Return the mix-net proofs Π_{mix} and the PET proofs/data Π_{PET} consisting of the values $d_{i,v}$, $s_{i,v}$ and the associated proofs $\pi_{i,v}$.

The $\mathsf{EqProof}$ part proves that the $s_{i,v}$ are correct rerandomisations of the c_i for the votes $v \in V$, which they are. The $\mathsf{SimEqProof}$ are fake proofs that the $d_{i,v}$ are the decryptions of the $s_{i,v}$ which is generally false since we chose the $d_{i,v}$ values randomly. As the encryption scheme in question is NM-CPA secure, no PPT adversary has more than a negligible change of telling a correct d-value from a false one without any proofs (indeed, this is why we have the proofs of correct decryption in the real tally) and since the proofs are zero-knowledge, we can assume that a PPT adversary cannot tell a real from a simulated proof. Therefore the proofs $\pi_{i,v}$ do not help in distinguishing real from fake $d_{i,v}$ either.

The adversary does know the result R (since the challenger in the BPRIV game outputs that and $\mathsf{SimTally}$ cannot change it) but the simulated decryptions $d_{i,v}$ are consistent with R and follow the same distribution as the real ones. Therefore we can claim that the output of the tallying oracle in case $\beta = 1$ is indistinguishable to PPT adversaries from the output in the case $\beta = 0$. The other information that the adversary sees are the ballots on the board (in particular the $\mathcal{O}\mathsf{VoteLR}$ ones which have a dependency on β) but these are ciphertexts in an NM-CPA secure encryption scheme so we can assume that they are indistinguishable to PPT adversaries too. We therefore conclude that KTV-Helios satisfies BPRIV and have proven the following.

Theorem 2. *KTV-Helios satisfies the BPRIV security definition.*

4.3 Strong Correctness and Strong Consistency

Together with BPRIV, [6] contains two auxiliary properties called strong correctness and strong consistency that are also required for a voting scheme to guarantee privacy. We define and check these properties here for the KTV scheme.

The Valid algorithm can reject new ballots based on the information already on the board (for example, it can reject a duplicate of an existing ballot). Strong correctness ensures that the rejection algorithm is not too stong, in particular that dishonest voters cannot manipulate the board to the point where it would prevent an honest voter from casting her ballot. To model this we let the adversary choose a bulletin board and test if an honest ballot, for which the adversary can choose all inputs, would get rejected from this board.

Since the original definition did not contain timestamps or a list of registered voter identities, we adapt the syntax of the original definition [6, Def. 9] to include these elements.

Definition 3. *A voting scheme S has strong correctness if no PPT adversary has more than a negligible probability of winning in the following experiment.*

1. *The challenger sets up the voting scheme and publishes the election public key pk and the list of voter identities and public keys I.*
2. *The adversary generates a board BB, a voter identity $id \in I$, a vote $v \in V$ and a timestamp $t \in [T_s, T_e]$.*
3. *The challenger creates a ballot $b = \mathsf{Vote}((id, \mathsf{sk}_{id}), id, v, t)$.*
4. *The adversary loses if there is already a ballot with timestamp $t' \geq t$ on BB.*
5. *The adversary wins if $\mathsf{Valid}(\mathsf{BB}, b)$ rejects the honest ballot b.*

We have made the following changes compared to the original definition: we have added identities id to match the syntax of our voting scheme and demanded that the adversary choose an $id \in I$ since otherwise the ballot b will quite legitimately be rejected. We have also added timestamps and the restriction that the adversary must choose a timestamp t satisfying both $t \leq T_e$ and $t > t'$ for any timestamp t' of a ballot already on the board BB. Otherwise one could trivially stop any more ballots from being accepted by putting a ballot with timestamp T_e on the board.

Lemma 2. *The voting scheme described in Section 2.3 satisfies strong correctness.*

Proof. If $\mathsf{Valid}(\mathsf{BB}, b)$ fails on a ballot then one of two things must have happened: $\mathsf{Validate}(b) = 0$ or the ciphertext c in b is already on the board somewhere.

$\mathsf{Validate}(b)$ only fails if the identity id in b is not in I, one of the proofs in b does not verify or the timestamp is out of its domain. Since we are considering a honestly generated ballot b in the strong correctness experiment, correctness of the proof schemes involved means that the proofs are correct.

Since the ballot b in question is created by Vote which picks a fresh random $r \leftarrow_{\$} \mathbb{Z}_q$, the probability of c colliding with a previous ciphertext (even an adversarially created one) is negligible. (To be precise, since we are assuming a PPT adversary, the board BB created by the adversary can only contain a polynomially bounded number of ciphertexts and since the probability of a collision with any of these is negligible individually, so is the sum of these probabilities for a union bound.) This proves Lemma 2. □

The definition of strong correctness may seem tautological (and the proof trivial) but it prevents the following counter-example from [6, Section 4.4]: an adversary can set a particular bit in a ballot of its own that causes the board to reject all further ballots. Assuming that either Alice wants to vote for (candidate)

1 and Bob wants to vote for 2 or the other way round, in a private voting scheme we would not expect the adversary to be able to tell who voted for 1. Without strong correctness, the adversary could let Alice vote then submit their "special" ballot to block the board, then ask Bob to vote. Since Bob's ballot now gets rejected, the result is exactly Alice's vote, so the adversary discovers how she voted.

Strong consistency prevents the Valid algorithm from leaking information in scenarios such as the following: the adversary can submit a special ballot that gets accepted if and only if the first ballot already on the board is a vote for 1. Of course this is mainly of interest where Valid has access to non-public information, either because it has access to a secret key or the board contains non-public information.

Strong consistency formally says that the election result is a function of the votes and that each valid ballot must be uniquely associated with a vote. In particular, the vote in one ballot cannot depend on the other ballots on the board.

Definition 4. *A voting scheme has strong consistency relative to a result function ρ if there are two algorithms*

- $\mathsf{Extract}(\mathsf{sk}, b)$ *takes an election secret key and a ballot and returns either a pair (id, v) containing an identity $id \in I$ and a vote $v \in V$, or the symbol $\perp$ to denote an invalid ballot.*
- $\mathsf{ValidInd}(\mathsf{pk}, b)$ *takes an election public key and a ballot and returns* 0 *(invalid ballot) or* 1 *(valid ballot).*

such that the following conditions hold.

1. *The extraction algorithm returns the identity and vote for honestly created ballots: for any election keypair $(\mathsf{pk}, \mathsf{sk})$ created by* Setup*, any voter registration list I and any ballot b created by $\mathsf{Vote}((id, \mathsf{sk}_{id}), id, v, t)$ where $id \in I$, $t \in [T_s, T_e]$ and $v \in V$ we have $\mathsf{Extract}(\mathsf{sk}, b) = (id, v)$.*
2. *Ballots accepted onto a board are also accepted by* ValidInd*: for any board* BB*, if $\mathsf{Valid}(\mathsf{BB}, b)$ holds then $\mathsf{ValidInd}(\mathsf{pk}, b)$ holds too.*
3. *For any PPT adversary A, the probability of winning the following game is negligible:*
   (a) *Create an election keypair $(\mathsf{pk}, \mathsf{sk})$ with* Setup *and set up user registration list I.*
   (b) *Give A all public* and secret *keys of the election and the users and let A return a board* BB*. Let n be the number of ballots on this board.*
   (c) *A loses if there is any ballot b on the board* BB *for which $\mathsf{ValidInd}(b) = 0$.*
   (d) *Let $(r_1, \Pi) = \mathsf{Tally}(\mathsf{sk}, \mathsf{BB})$. The adversary loses if the tallying function returns $\perp$.*
   (e) *Let $e_i = \mathsf{Extract}(b_i)$ for $i = 1, \ldots, n$ and the b_i are the ballots on the board* BB*. Let $r_2 = \rho(e_1, \ldots, e_n)$.*
   (f) *The adversary wins if $r_1 \neq r_2$.*

We prove that KTV-Helios satisfies strong consistency. This means that we have to check that the tally function really counts the votes in the ballots.

For $\mathsf{Extract}(\mathsf{sk}, b)$ we parse the ballot as $b = (id, c, \pi_{PoK}, \pi, t)$ and check the two proofs; if either of them fail then we return $\perp$. Then we decrypt c with sk to get a vote v. If $v \in V$ then we return (id, v) otherwise we return $\perp$. For $\mathsf{ValidInd}(\mathsf{pk}, b)$ we just run $\mathsf{Validate}(b)$. We can assume that the list I of voter identities and public keys is public. We check the three conditions:

1. This follows from correctness of the encryption and proof schemes. If we encrypt a vote $v \in V$ to get ciphertext c then we also get v back when we decrypt c with the matching key and the correct voting algorithm produces correct proofs too.
2. Since Valid runs $\mathsf{Validate}$, it must hold that ballots accepted onto the board are valid.
3. In fact the probability of an adversary winning this game is zero. Consider an execution of the experiment in which $r_1 \neq r_2$ in the last stage. We know that Tally did not return $\perp$ or we would not have got this far, therefore all ballots on the board passed $\mathsf{Validate}$ individually and the board as a whole passed $\mathsf{ValidBB}(\mathsf{BB})$. In particular $\mathsf{ValidateBB}$ did not cause tallying to abort.

   In this case, by the definition of Tally, the result r_1 is obtained by homomorphically adding the ciphertexts of each voter, mixing (which does not change the votes) and then PET-decrypting the resulting ballots which for all valid votes produces the same result as normal decryption whereas invalid ones are discarded.

   The extraction to get r_2 on the other hand first decrypts each ciphertext individually, then (to evaluate ρ) sums the decrypts for each voter, discards invalid sums and then reports the number of votes for each option. By the homomorphic property of the encryption scheme, these two methods of tallying must return the same result r (strong consistency does not deal with the proofs Π of correct tallying).

This concludes the proof of strong consistency. □

5 Participation privacy

We first provide a cryptographic definition of probabilistic participation privacy (see Section 5.1). The definition is inspired by the coercion resistance definition in [22]. Similar to the notion of δ-coercion resistance in [22], we speak of δ-participation privacy, where δ denotes the advantage of the adversary who tries to tell whether a given voter has participated in the election. In Section 5.2, we instantiate this definition for the KTV-Helios and provide the optimal value of δ, so that KTV-Helios satisfies δ-participation privacy.

5.1 Defining δ-participation privacy

Let (N, n_h, L, p) be the parameters characterising an election, so that N is a total number of eligible voters, n_h the number of honest voters, L the total number of valid voting options, and p a vector of probabilities $p_0, ..., p_L$. p_i for $i = 1, ..., L$ denotes the probability that an honest voter cast a vote for a valid voting option o_i in this election. p_0 denotes the probability that an honest voter abstained.

We define δ-participation privacy for a single honest voter id^A (similar to the authors of [22] who considered coercion of a single voter). δ-participation privacy for all voters is then ensured if the individual voters' decisions whether to cast a vote or not are independent. Thus, knowing whether an honest voter id^B abstained, does not give the adversary any additional information on whether the id^A abstained.

We consider the following experiment $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},id^A}$ given the adversary $\mathcal{A} \in C_S$, so that C_S is a set of PPT adversaries, defined according the adversarial model for a particular scheme. There are two bulletin boards BB_0, BB_1, which are filled by the challenger modelling the voting phase. Let Q_S be a set of oracle queries, the adversary has access to. Using these queries, the adversary fills both of the bulletin boards with additional content, so that BB_0 and BB_1 contain the same cast ballots except that BB_1 also contains a ballot cast for the voter id^A.

The experiment computes the tally result for BB_β ($\beta = 0$ if the voter id^A abstains and $\beta = 1$ otherwise) and provides it to the adversary together with the public output of BB_β. The goal of the adversary is to guess whether the provdided tally result corresponds to BB_0 or to BB_1, i.e. to output β.

The definition of δ-participation privacy is then as follows:

Definition 5. *The voting scheme $\mathcal{S}$ in the election with the parameters (N, n_h, L, p) achieves δ-participation privacy given a subset of PPT adversaries C_S, if for any adversary $\mathcal{A} \in C_S$ and an honest voter id^A holds*

$$\Pr\left[\mathsf{Exp}^{\mathsf{ppriv},0}_{\mathcal{A},\mathcal{S},id^A} = 0\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ppriv},1}_{\mathcal{A},\mathcal{S},id^A} = 0\right] - \delta$$

is negligible in the security parameter.

5.2 δ-participation privacy in the KTV-Helios scheme

In order to evaluate δ-participation privacy in the KTV-Helios scheme according to the aforementioned definition, we first need to specify the adversary $\mathcal{A} \in C_S$ we aim to protect against. Afterwards we consider the information sources that would help the adversary $\mathcal{A} \in C_S$ to correctly guess β at the end of the experiment. This is done in order to determine the optimal value of δ, so that the KTV-Helios scheme satisfies δ-participation privacy with $\mathcal{A} \in C_S$ according to Definition 5. We conclude the subsection by showing how to calculate this optimal value of δ depending on the information leakage from those sources.

Specification of $\mathcal{A} \in C_S$. We make following assumptions regarding adversarial capabilities: (1) the tabulation teller does not divulge her secret key to the adversary, (2) the adversary is incapable of observing the communication channel between the voter, the posting trustee and the voting system, (3) the posting trustee does not divulge private information to the adversary, (4) the bulletin board acts according to its specification, (5) the honest voters decide to participate or to abstain in the election independently from each other. Thus, we asumme that the adversary is only able to cast dummy ballots on behalf of any voter and non-dummy ballots on behalf of corrupted voters.

Hence, we define C_S as a set of adversaries that are given access to the queries $Q_S = \{O\mathsf{Cast}, O\mathsf{CastDummy}, O\mathsf{VoteDummy}, O\mathsf{Tally}\}$ in the experiment $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},id^A}$. These queries are defined as follows:

- $O\mathsf{Cast}(b)$: the adversary casts a ballot on behalf of a corrupted voter by appending b to both of the bulletin boards BB_0 and BB_1. If the ballot b is invalid (namely, $\mathsf{ValidBB}(\mathsf{BB}_\beta, b) =\perp$), the query terminates and returns $\perp$.
- $O\mathsf{CastDummy}(id, t)$: the adversary casts a dummy vote next to a specified id by computting $b = \mathsf{Vote}((\hat{id}, 0), id, 0, t)$, which is then appended to both of the bulletin boards.
- $O\mathsf{VoteDummy}(v)$: the oracle appends a series of dummy ballots $b_1, ..., b_m \leftarrow\!\!\$\ \mathsf{VoteDummy}(id^A)$ to both of the bulletin boards next to the voter id^A, with m sampled by the oracle according to the probability distribution $\mathbb{P}_{dummy}$ defined as in Section 2.2. Additionally, the oracle appends a ballot $b' = \mathsf{Vote}((id^A, \mathsf{sk}_{id^A}), id^A, v, t')$ for the voter id^A, with a random timestamp $t' \leftarrow\!\!\$\ \mathbb{P}_t$. The adversary is allowed to query $O\mathsf{VoteDummy}(v)$ only once.
- $O\mathsf{Tally}$: The oracle returns the tally result R with the validity proofs Π, $(R, \Pi) = \mathsf{Tally}(\mathsf{BB}_\beta, \mathsf{sk})$. The adversary is allowed to query $O\mathsf{Tally}$ only once.

We now consider the **sources of information** that would help the adversary $\mathcal{A} \in C_S$ to correctly guess β at the end of the experiment $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},id^A}$: The first source of such information is the tally result that consists of a vector $v = (v_0, v_1, ..., v_L)$, with v_0 denoting the number of abstaining voters and v_i as the number of votes for each voting option o_i. Given the vector p representing the probabilities that an honest voter would abstain or vote for a particular voting option, we denote by δ_{ideal} the adversarial advantage, so that the ideal scheme, that outputs only this vector v, is δ_{ideal}-participation private, but is not δ'_{ideal}-participation private for any $\delta'_{ideal} < \delta_{ideal}$. The methods for calculating δ_{ideal} given the number of honest voters n_h, the number of voting options L and the probability vector p are given in [22]. Namely, δ_{ideal} can be computed using the formula for δ^0_{min} provided in [22].

The second source that can be used by the adversary is one additional ballot on the bulletin board BB_1 as the output of $O\mathsf{VoteDummy}(v)$. In order to account for the adversarial advantage gained from the number of ballots next to voter's id on the bulletin board, we define the following experiment $\mathsf{Exp}^{\mathsf{num},\beta}_{\mathcal{A},\mathbb{P}_{dummy},\mathbb{P}_t}$: Again, the challenger sets $\beta = 0$ if the voter id^A abstains and $\beta = 1$ if she casts a ballot in the election. She then outputs the number $m + \beta$, with $m \leftarrow\!\!\$\ \mathbb{P}_{dummy}$, and the set of timestamps $t_1, ..., t_m, t_{m+\beta}$ that are independently sampled from $\mathbb{P}_t$ to the adversary. The adversary outputs β. Let $\delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t}$ denote an advantage in this experiment, so that

$\Pr\left[\mathsf{Exp}^{\mathsf{num},0}_{\mathcal{A},\mathbb{P}_{dummy},\mathbb{P}_t} = 0\right] - \Pr\left[\mathsf{Exp}^{\mathsf{num},1}_{\mathcal{A},\mathbb{P}_{dummy},\mathbb{P}_t} = 0\right] - \delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t}$ is negligible. This advantage $\delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t}$ can be calculated given the probability distribution $\mathbb{P}_{dummy}$ used by the posting trustee to determine a number of dummy ballots for each voter. For the sake of brevity, we limit our analysis on a particular class of probability distributions $\mathbb{P}_{dummy}$, namely, unimodal distributions with support in $\mathbb{N}_0$, and show that $\delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t}$ equals the maximal value of $\mathbb{P}_{dummy}$.

Theorem 3. *Let $\mathbb{P}_{dummy}$ be a discrete unimodal distribution with support $[x, \infty)$, $x \geq 0$, and $\mathbb{P}_t$ chosen such that it corresponds to the distribution of times at which the voters cast their ballots. It holds, for the adversarial advantage in the experiment $\delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t} = \mathsf{Exp}^{\mathsf{num},\beta}_{\mathcal{A},\mathbb{P}_{dummy},\mathbb{P}_t}$:*

$$\delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t} = P(X = m_{max})$$

for X as a random variable X distributed in $\mathbb{P}_{dummy}$, and m_{max} as the mode of $\mathbb{P}_{dummy}$ (i.e. $P(X = m') \leq P(X = m)$ for all $m' \neq m_{max}$).

We give the proof of this theorem in Appendix E.

Determining the optimal value for $\boldsymbol{\delta}$. We are now ready to determine an optimal value δ for the KTV-Helios scheme as $\delta = \delta_{ideal}$ and $\delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t}$. Hence we show, that for such δ, the KTV-scheme achieves δ-participation privacy, but does not achieve δ'-participation privacy for any lower values of δ. We do this by proving the following theorem.

Theorem 4. *The voting scheme described in Section 2.3 that is instantiated with the probability distributions $\mathbb{P}_{dummy}, \mathbb{P}_t$ in the election characterised by (N, n_h, L, p) achieves δ-participation privacy given the subset of adversaries C_S, with $\delta = \delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t} + \delta_{ideal}$. It further does not achieve δ'-participation privacy for any $\delta' < \delta$.*

We briefly sketch the proof idea (the detailed proof is provided in Appendix D): We prove, that the distinguishing advantage in $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},id^A}$, defined as is negligibly larger than δ. Namely, we prove, that the only sources of information available to the adversary for distinguishing between $\mathsf{Exp}^{\mathsf{ppriv},0}_{\mathcal{A},\mathcal{S},id^A}$ and $\mathsf{Exp}^{\mathsf{ppriv},1}_{\mathcal{A},\mathcal{S},id^A}$ with non-negligible advantage are the ones described above: the election result (characterised by the distinguishing advantage δ_{ideal}) and the number of ballot next to the voter id^A (characterised by the distinguishing advantage $\delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t}$). It holds, that the rest of the output of $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},id^A}$ provides only negligible distinguishing advantage due to the ballot privacy property of the KTV-Helios scheme. It follows that the distinguishing advantage of $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},id^A}$ is negligibly larger than δ, hence, the KTV-Helios scheme achieves δ-participation privacy, but not δ'-participation privacy for any $\delta' < \delta$.

6 Related Work

Several *concrete and abstract definitions* of security requirements and underlying assumptions have been developed applying various formalization approaches: An overview of game-based ballot privacy definitions was proposed in [6], and a framework that proposes a uniform treatment of the verifiability definitions from [23, 5, 10, 18, 28] is described in [11]. Other approaches for defining and evaluating the security of voting schemes include applied pi-calculus [19, 4, 14], process algebra [24] or k-resilience terms [27]. These appraoches have been applied to evaluate various voting schemes [20, 2, 14, 27]. In particular, the formal security analysis of Helios has been the topic of [6, 10, 20].

Due to its versatility, *various extensions and modifications of the original Helios scheme from [1]* have been proposed: A modification has been proposed by [7] to fix vulnerabilities in [1]. Zeus [30], Selene [26] or the work by Guasch et al. [16] propose alternatives to the cast-as-intended mechanism used in Helios. Improvements of various security properties were proposed in [15] (ensuring long-term ballot privacy), in [10] (ensuring verifiability against malicious bulletin board), in [9] (introducing threshold decryption for better robustness); and in [12] (ensuring receipt-freeness). Other research focused on improving the usability of the Helios system, in particular on making the cast as intended verification more usable [25, 17].

7 Conclusion and future work

We have evaluated the security properties of the KTV-Helios extension proposed in [21]. Namely, we have proven that the KTV-Helios extension satisfies ballot privacy and verifiability against malicious bulletin board according to the definitions from [6, 10] adjusted to the context of KTV-Helios. Furthermore, we have proposed an abstract definition for participation privacy. Similar to the definition of coercion resistance in [22], we proposed a definition of δ-participation privacy, with δ representing the adversarial advantage in distinguishing whether a particular honest voter has cast her ballot in the election. We further proposed an instantiation of the definition for KTV-Helios and determined the value of δ as the adversarial advantage for the adversary.

Future work. Another security property that the KTV-Helios extension proposes is receipt-freeness. This property is ensured in a similar way to participation privacy: due to the dummy ballots that obfuscate the real ballots, the voters have an option to deniably update their vote, thus making it impossible to construct a receipt for voting for a particular voting option. However, none of the existing definitions of receipt-freeness considers this case (as opposed to the definitions of a stronger security property, coercion resistance), which like δ-participation privacy depends on a (possibly non-negligible) factor δ that models the indistinguishability of real from dummy ballots. We aim to formalise and prove this property for KTV-Helios in future work.

We further plan to extend the proofs for ballot privacy and verifiability for the case where the tabulation teller and the posting trustee are implemented in a distributed way. In particular, we aim to prove that the aforementioned security properties hold if the tallying is implemented via distributed decryption.

Acknowledgements

We would like to thank Vincent Cheval, Véronique Cortier, Marc Fischlin, Vanessa Teague and Tomasz Trudering for their helpful feedback at various stages.

The research that led to these results has been partially funded from a project in the framework of Hessen ModellProjekte (HA project no. 435/14-25), financed with funds of LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben (State Offensive for the Development of Scientific and Economic Excellence).

References

1. Ben Adida. Helios: Web-based open-audit voting. In *17th Conference on Security Symposium*, SS'08, pages 335–348. USENIX, July 2008.
2. Mathilde Arnaud, Véronique Cortier, and Cyrille Wiedling. Analysis of an electronic boardroom voting system. In *E-Voting and Identify: 4th International Conference, Vote-ID 2013*, pages 109–126. Springer, July 2013.
3. N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. In *Advances in Cryptology — EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques*, pages 591–606. Springer, June 1998.
4. M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *21st IEEE Computer Security Foundations Symposium*, pages 195–209. IEEE, June 2008.
5. Josh Daniel Cohen Benaloh. Verifiable secret-ballot elections. *PhD thesis, Yale University, Department of Computer Science*, 1987.
6. D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. In *2015 IEEE Symposium on Security and Privacy*, pages 499–516. IEEE, May 2015.
7. David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *Advances in Cryptology – ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security*, pages 626–643. Springer, December 2012.
8. David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Advances in Cryptology — CRYPTO' 92: 12th Annual International Cryptology Conference*, pages 89–105. Springer, August 1993.
9. Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Distributed elgamal à la pedersen: Application to helios. In *12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, pages 131–142. ACM, November 2013.
10. Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election verifiability for helios under weaker trust assumptions. In *Computer Security - ESORICS 2014: 19th European Symposium on Research in Computer Security, Part II*, pages 327–344. Springer, September 2014.

11. Veronique Cortier, David Galindo, Ralf Kuesters, Johannes Mueller, and Tomasz Truderung. Verifiability notions for e-voting protocols. Cryptology ePrint Archive, Report 2016/287, March 2016. http://eprint.iacr.org/.
12. Véronique Cortier, Georg Fuchsbauer, and David Galindo. Beleniosrf: A strongly receipt-free electronic voting scheme. Cryptology ePrint Archive, Report 2015/629, June 2015. http://eprint.iacr.org/.
13. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94: 14th Annual International Cryptology Conference*, pages 174–187. Springer, August 1994.
14. Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
15. Denise Demirel, Jeroen Van De Graaf, and Roberto Samarone dos Santos Araújo. Improving helios with everlasting privacy towards the public. In *2012 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX, August 2012.
16. Sandra Guasch and Paz Morillo. How to challenge and cast your e-vote. In *20th international conference on Financial Cryptography and Data Security*. IFCA, February 2016.
17. Fatih Karayumak, Michaela Kauer, M Maina Olembo, Tobias Volk, and Melanie Volkamer. User study of the improved helios voting system interfaces. In *Socio-Technical Aspects in Security and Trust (STAST), 2011 1st Workshop on*, pages 37–44. IEEE, September 2011.
18. Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *Advances in Cryptology - EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II*, pages 468–498. Springer, April 2015.
19. Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *Programming Languages and Systems: 14th European Symposium on Programming, ESOP 2005*, pages 186–200. Springer, April 2005.
20. Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *Computer Security – ESORICS 2010: 15th European Symposium on Research in Computer Security*, pages 389–404. Springer, September 2010.
21. Oksana Kulyk, Vanessa Teague, and Melanie Volkamer. Extending helios towards private eligibility verifiability. In *E-Voting and Identity: 5th International Conference, VoteID 2015*, pages 57–73. Springer, September 2015.
22. R. Küsters, T. Truderung, and A. Vogt. A game-based definition of coercion-resistance and its applications. In *2010 23rd IEEE Computer Security Foundations Symposium*, pages 122–136. IEEE, July 2010.
23. Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. In *17th ACM Conference on Computer and Communications Security*, CCS '10, pages 526–535. ACM, April 2010.
24. Murat Moran, James Heather, and Steve Schneider. Verifying anonymity in voting systems using csp. *Formal Aspects of Computing*, 26(1):63–98, December 2012.
25. Stephan Neumann, M. Maina Olembo, Karen Renaud, and Melanie Volkamer. Helios verification: To alleviate, or to nominate: Is that the question, or shall we have both? In *Electronic Government and the Information Systems Perspective: Third International Conference, EGOVIS 2014*, pages 246–260. Springer, September 2014.

26. Peter Y A Ryan, Peter B Roenne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. Cryptology ePrint Archive, Report 2015/1105, November 2015. http://eprint.iacr.org/.
27. Guido Schryen, Melanie Volkamer, Sebastian Ries, and Sheikh Mahbub Habib. A formal approach towards measuring trust in distributed systems. In *2011 ACM Symposium on Applied Computing*, SAC '11, pages 1739–1745. ACM, March 2011.
28. Ben Smyth, Steven Frink, and Michael R. Clarkson. Election verifiability: Cryptographic definitions and an analysis of helios and jcj. Cryptology ePrint Archive, Report 2015/233, March 2015. http://eprint.iacr.org/.
29. Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In *Progress in Cryptology – AFRICACRYPT 2010: Third International Conference on Cryptology in Africa*, pages 100–113. Springer, May 2010.
30. Georgios Tsoukalas, Kostas Papadimitriou, Panos Louridas, and Panayiotis Tsanakas. From helios to zeus. In *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX, August 2013.

A Multiple entities

In our proofs, we assumed that the tabulation teller and the posting trustee are implemented as a single entity each.

We first consider the case with a total of $N_p > 1$ posting trustees. We assume that each of them acts independently from the others. Hence, the function $\mathsf{VoteDummy}$ is run a total of N_p times for each voter. As mentioned in the introduction, our goal was to ensure participation privacy for the case, where at most $N_p - 1$ posting trustees are corrupted. In that case, in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ppriv},\beta}$ defined Section 5, the query $O\mathsf{VoteDummy}$ corresponds to a remaining honest posting trustee casting dummy ballots on behalf of the voter. Hence, the definition and proofs from Section 5 still hold.

There are several ways to implement multiple tabulation tellers. In particular, for the shuffling, each tabulation teller could act as a separate mix node, and the decryption could be implemented in a threshold distributed way. Similar to [6], the security proofs for such a case will be considered in future work.

B Proof of partial counting property

We show, that the plaintext tally function ρ described in Section 2.2 has the partial counting property. Let $I = \{id_1, ..., id_N\}$ be a set of voter ids, $\bar{id} \notin I$ the id denoting the posting trustee, $\{o_1, ..., o_L\} \in \mathbb{G}_q \setminus \{0\}$ a set of valid voting options, and let $\mathbb{V}_{cast}$ be a set of tuples (id, v) with $id \in I \cup \{\bar{id}\}$ and $v \in \mathbb{G}_q$.

Let $I_1, .., I_k$ be partitions of $I \cup \{\bar{id}\}$, so that $\bigcup_{i=1}^{k} I_i = I \cup \{\bar{id}\}$ and $I_i \cap I_j = \emptyset$ for all $i \neq j$. We further define the lists $\mathbb{V}_{cast}^{(i)} \subset \mathbb{V}_{cast}$ as a list of all the tuples $(id, v) \in \mathbb{V}_{cast}$, for which holds $id \in I_i$.

The partial counting property means, that the tally on $\mathbb{V}_{cast}$ can be expressed as a sum of tallies on all the lists $\mathbb{V}_{cast}^{(i)}$, $i = 1, ..., k$. Namely, it should hold

$$\rho(\mathbb{V}_{cast}) = \sum_{i=1}^{k} \rho(\mathbb{V}_{cast}^{(i)})$$

In order to prove this, consider the output of $\rho(\mathbb{V}_{cast}^{(i)})$. Let ρ' be the function, that, given the list of plaintext votes $v_1, ..., v_N$ outputs the number of votes for each voting option $o_1, ..., o_L$ and the number of abstaining voters. Namely, on the input of $v_1, ..., v_N \in (\{o_1, ..., o_L\} \cup 0)^{L+1} \subset \mathbb{G}_q^{L+1}$, ρ' returns a vector of values $R \in \mathbb{N}_0^{L+1}$. It holds, that ρ' supports partial counting. Namely, for two lists $S_1 = (v_{1,1}, ..., v_{N_1,1})$ and $S_2 = (v_{2,1}, ..., v_{N_2,2})$ with $S_1, S_2 \in (\{o_1, ..., o_L\} \cup 0)^{L+1}$, it holds

$$\rho'(S_1) + \rho'(S_2) = \rho'(S_1 \cup S_2)$$

As described in Section 2.2, with $\mathbb{V}$ as a set of tuples $(id, v) \in I \cup \{\bar{id}\} \times \mathbb{G}_1$, the function ρ outputs $R = \rho'(v_1, ..., v_N)$ with v_i, $i = 1, ..., N$ being either the sum of all votes cast by the voter $id_i \in I$, or $v_i = 0$ if there were no valid votes from the voter id_i in $\mathbb{V}$ (i.e. there is no tuple (id_i, v) with $v \in \{o_1, ..., o_L\}$ in $\mathbb{V}$).

it follows that $\rho(\mathbb{V}_{cast}^{(i)}) = \rho'(v_{1,i}, ..., v_{N,i})$ with $v_{j,i}$ denoting the sum of all cast votes by the voter id_j if $id_j \in I_i$, and $v_{j,i} = 0$ if $id_j \notin I_i$. Combined with the partial counting property of ρ' it follows that

$$\rho(\mathbb{V}_{cast}) = \sum_{i=1}^{k} \rho(\mathbb{V}_{cast}^{(i)})$$

□

C Proof of verifiability for the KTV-Helios scheme

We proceed with the proof as follows: (1) We first prove that each well-formed ballot $b_1, ..., b_n$ on the bulletin board was either cast by an honest voter who checked whether the ballot is properly stored on the bulletin board, by an honest voter who did not check this, by a corrupted voter, or the ballot corresponds to a null vote. (2) We then show that the plaintext tally result on all the votes corresponding to these ballots together correspond to all the votes cast by honest voters who checked that their vote is stored on the bulletin board, a subset of honest voters who did no such checks, at most $|I^C|$ votes cast by the adversary and the dummy votes. After this, we prove (3) that this plaintext tally result corresponds to the result output by the tally function Tally, if this function is applied according to its specification in Section 2.3. We conclude by proving (4), that the adversary is incapable of producing a tally result that passes the verification check, and yet is different from the tally result output by Tally.

Step 1. Let $b = (id, c, \pi_{PoK}, \pi, t)$ be a well-formed ballot (that passes Validate) on the board. We prove that b belongs to one of the following lists with overwhelming probability:

- $\mathbb{V}_{cast}^{HC} := ((id_{E,1}, v_{E,1}), ..., (id_{E,n_E}, v_{E,n_E}))$ the list of all tuples of honest voters and non-zero votes (i.e. $((id_{E,i}, v_{E,i}), *) \in \mathsf{HVote}$) who verified that their vote is properly stored on the bulletin board (i.e. $id_{E,i} \in \mathsf{Checked}$).
- $\mathbb{V}_{cast}^{HU} := ((id_{A,1}, v_{A,1}), ..., (id_{A,n_A}, v_{A,n_A}))$, the list of all tuples of honest voters and non-zero votes (i.e. $((id_{A,i}, v_{A,i}), *) \in \mathsf{HVote}$) who did not verify that their vote is properly stored on the bulletin board (i.e. $id_{E,i} \notin \mathsf{Checked}$).
- $\mathbb{V}_{cast}^{C} := ((id_{B,1}, v_{B,1}), ..., (id_{B,n_B}, v_{B,n_B}))$, the list of all tuples of corrupted voters with non-zero votes (i.e. $id_{B,i} \in I^C$) and their votes.
- $\mathbb{V}_{cast}^{D} := \{(*, 0)\}^{n_D}$: the list of all tuples that correspond to zero-votes.

From soundness of the proof π we conclude that the ciphertext c ballot b is signed by the voter's secret key or else c encrypts zero, in which case b is a zero-ballot and $(\hat{id}, 0)$ must be in $\mathbb{V}_{cast}^{D}$. If b is signed, by unforgeability of the signature scheme and the assumption that honest voters do not reveal their signing keys, either b was cast by a corrupt voter and so $(id, v) \in \mathbb{V}_{cast}^{C}$ where v is the vote in c or else b was cast by a honest voter and so (id, v) must lie in one of the other two lists (depending on whether $id \in \mathsf{Checked}$ or not).

Step 2. We prove that applying the tally function ρ to the lists in step 1 inlclues all votes by honest voters who checked their ballots, at most I^C votes by corrupt voters and a subset of the remaining honest votes (by voters who did not check).

If there were no complaints from the voters in $\mathsf{Checked}$, which would have caused the adversary to lose the security game, we know that all the ballots from these voters must be on the board so all their votes are in $\mathbb{V}_{cast}^{HC}$. The adversary's ballots are only the ones in $\mathbb{V}_{cast}^{C}$ whose identities are in I^C so the number of these ballots is at most $|I^C|$. All the remaining ballots are in $\mathbb{V}_{cast}^{HU}$ and so must have been cast by non-checking honest voters. Since ρ supports partial counting as expained in Section 2.2 we conclude, for $\mathbb{V}_{cast}$ the list of all votes in ballots on the board:

$$\rho(\mathbb{V}_{cast}) = \rho(\mathbb{V}_{cast}^{HC}) + \rho(\mathbb{V}_{cast}^{HU}) + \rho(\mathbb{V}_{cast}^{C}).$$

Step 3. We prove that applying $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk})$ to the ballots on the board tallies them correctly, i.e. the result R corresponds to $\rho(\mathbb{V}_{cast})$.

The homomorphic property of ElGamal means that the ciphertexts input to the mix contain the sum of all votes cast under the name of each voter. The mix does not change the encrypted values in the ciphertexts, it just permutes them around. Since ElGamal is a correct encryption scheme and the PET is sound, the decrypted values output in the PET correspond to the messages in the mixed ciphertexts. It follows that the result output by $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk})$ corresponds to the function ρ applied to the votes in the ballots on BB. (This step is essentially a proof of correctness for the KTV scheme.)

Step 4. We prove that the adversary cannot output a result/proof pair (R', Π') for a result $R' \neq R$ different from the result R that Tally would return, which passes $\mathsf{ValidateTally}$.

The homomorphic sum-ciphertexts for each voter are recomputed by $\mathsf{ValidateTally}$ to be able to check the mix. The mix is protected by the proof $\pi_{mix} \in \Pi'$ which is sound, so the mixed ciphertexts $(\bar{c}_i) \in \Pi'$ must be a valid permutation and rerandomisation of those on the board. The PET decryptions too are protected by a sound proof so the decryption factors d in Π must match the ballots on the board. From these, the result R can be recomputed. Therefore, unless one of the proofs in Π is invalid (which would contradict soundness) we conclude that if $\mathsf{ValidateTally}(\mathsf{BB}, (R', \Pi'))$ only outputs 1 when R is the correct result for BB.

Hence, the adversarial success probability $\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathcal{S}}^{\mathsf{ver-b}} = 1\right]$ is negligible. This proves verifiability against malicious bulletin boards.

□

D Proof of participation privacy for the KTV-Helios scheme

We base our proof on the idea, that the aforementioned two sources of information (i.e. the tally result and the number of ballots next to id^A) are the only ones that give advantage to the adversary. The rest of the public election data, as in case of ballot privacy (as shown in Section 4), does not provide any advantage to the adversary.

Our proof strategy is hence as follows. We consider a sequence of games, starting from $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ppriv},0}$ and ending with $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ppriv},1}$ and show, that the adversary $\mathcal{A}$ that is given access to the queries in Q_S distinguishes the transition through all those games with the advantage of at most $\delta_{num} + \delta_{ideal}$. We define $\mathsf{BB}_{0,i}$ as the content of the bulletin board and (R_i, Π_i) as the tally output at the end of the game G_i, $i = 1, ..., 4$. We define the sequence as follows:

- G_1**.** The first game G_1 is equivalent to the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ppriv},\beta}$ with $\beta = 0$ (hence, it is equivalent to the election where the voter id^A abstains). Thus, the content of $\mathsf{BB}_{0,1}$ and the tally output (R_1, Π_1) corresponds to the content of BB_0 and the output of $O\mathsf{Tally}$ at the end of $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ppriv},0}$.
- G_2**.** The second game G_2 is equivalent to the election, where the voter id^A cast a ballot with a null-vote. Thus, the content of the bulletin board $\mathsf{BB}_{0,2}$ is equivalent to the content of the bulletin board BB_1 at the end of $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ppriv},1}$ for the adversary using the query $O\mathsf{VoteDummy}(v)$ with $v = 0$.

We prove, that the adversary has an advantage of δ_{num} of distinguishing between the output of G_1 and G_2. The tally result does not change, hence the tally output (R_2, Π_2) is equivalent to the tally output (R_1, Π_1). The only difference between the contents of $\mathsf{BB}_{0,1}$ and $\mathsf{BB}_{0,2}$ is the presence of an additional ballot with the encryption of 0 on $\mathsf{BB}_{0,2}$. Therefore, we conclude that the challenge in distinguishing between the outputs of G_1 and G_2 is equivalent to the challenge in distinguishing between the output of $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{num},0}$ and $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{num},1}$, and therefore the adversarial advantage of distinguishing between the output of G_1 and G_2 is $\delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t}$.

- G_3**.** The third game G_3 is equivalent to the election, where the voter cast a vote for a non-null voting option $v \neq 0$, however, the tally result R does not include the vote of id^A, and the tally validity proofs Π are simulated as described in Section 4. Namely, the content of the bulletin board $\mathsf{BB}_{0,3}$ is equivalent to the content of the bulletin board BB_1 at the end of $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ppriv},1}$ for the adversary using the query $O\mathsf{VoteDummy}(v)$ with $v \neq 0$. The tally output (R_3, Π_3) is calculated as follows: let $\mathsf{BB}'_{0,3}$ contain the content of $\mathsf{BB}_{0,3}$ excluding the non-dummy ballot next to id^A. The tally output is calculated as $(R_3, \Pi) = \mathsf{Tally}(\mathsf{BB}'_{0,3}, \mathsf{sk})$, and the simulated proof of tally validity as $\Pi_3 = \mathsf{SimProof}(\mathsf{BB}_{0,3}, R)$.

We now prove, that the adversarial advantage in distinguishing between the output of G_2 and G_3 is negligible. Consider an adversary $\mathcal{B}$ in the ballot privacy experiment (Section 4) $\mathsf{Exp}_{\mathcal{A},\mathcal{S}}^{\mathsf{bpriv},\beta}$, who simulates the games G_2 and G_3 for the adversary $\mathcal{A}$. The adversary $\mathcal{B}$ returns the output of $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{bpriv},\beta}$ for the queries $O\mathsf{Cast}$ and $O\mathsf{Tally}$. For simulating the output of $O\mathsf{VoteDummy}(v)$, $\mathcal{B}$ proceeds as follows: first, she chooses a random value $m \leftarrow_{\$} \mathbb{P}_{dummy}$, and a set of and random timestamps $t_1, ..., t_m \leftarrow_{\$} \mathbb{P}_t$, and computes a set of ballots $b_1, ..., b_m$ with $b_i = \mathsf{Vote}((\hat{id}, 0), id, 0, t_i)$. She then uses the query $O\mathsf{VoteLR}(id^A, id^A, 0, v, t)$ for a random $t \in \mathbb{P}_t$ in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{bpriv},\beta}$ and returns its output together with the ballots $b_1, ..., b_m$ to $\mathcal{A}$. At the end, $\mathcal{B}$ returns the value β output by $\mathcal{A}$ as the guess in $\mathsf{Exp}_{\mathcal{A},\mathcal{S}}^{\mathsf{bpriv},\beta}$. Thus, it follows that the adversarial advantage in distinguishing G_2 from G_3 is at most equal to the adversarial advantage in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{bpriv},\beta}$, denoted as δ_{BPRIV}.

- G_4**.** The fourth game G_4 is equivalent to the G_3 except that this time the real tally result is output (i.e. the result that includes the vote from id^A). Thus, the contents of $\mathsf{BB}_{0,4}$ are the same as of $\mathsf{BB}_{0,4}$, and the tally output result is calculated as $(R_4, \Pi_4) = \mathsf{Tally}(\mathsf{BB}_{0,4}, \mathsf{sk})$. Hence, since the tally result is the only difference in the output of the games G_3 and G_4, the adversary distinghuishes between the outputs of two games with the same advantage as in the ideal scheme, namely δ_{ideal}.

It follows, that the in transition through the game sequence $G_1 \rightarrow G_2 \rightarrow G_3 \rightarrow G_4$, the outputs of each game are distinguished from the outputs of a previous game with the advantage either $\delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t}$ (for games G_1 and G_2), δ_{BPRIV} (for games G_2 and G_3) and δ_{ideal} (for games G_3 and G_4). Hence, the adversary distinguishes between the output in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ppriv},\beta}$ with the advantage $\delta = \delta_{ideal} + \delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t} + \delta_{BPRIV}$, with δ_{BPRIV} negligible as proven in Section 4. □

E Proof of Theorem 3

In this section we prove Theorem 3.

Since probability distribution for times of casting the dummy ballots $\mathbb{P}_t$ is chosen in such a way, that it corresponds to the distribution of times at which the voters cast their ballots, the timestamps on the ballots do not provide any additional information to the adversary. Hence, we only consider the adversary seeing the total number of cast ballots next to the voter.

We consider the upper bound δ_{num} of difference between the probability that the adversary guesses correctly that the voter has abstained, and the probability that the adversary decides incorrectly that the voter has abstained (i.e. $\delta_{num} = \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{num},0} = 0\right] - \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{num},1} = 0\right]$).

Let m denote the number of ballots cast next to the voter id^A. It holds, that either all m of them were cast by the oracle as dummy ballots (modelling the posting trustee) if the voter id^A has abstained (i.e. $\beta = 0$), or $m-1$ of them were cast by the oracle as dummy ballots and one as the ballot by id^A, if the voter id^A did not abstain (i.e. $\beta = 1$). Let M_c be the set of all values of m, so that the adversary decides that the voter abstained from the election and outputs $\beta = 0$ at the end of the experiment. Then, it holds for the adversarial advantage, and a random variable X distributed in $\mathbb{P}_{dummy}^{\theta}$:

$$\delta_{num} = \Pr\left[\mathsf{Exp}_{\mathcal{A},\mathbb{P}_{dummy},\mathbb{P}_t}^{\mathsf{num},0} = 0\right] - \Pr\left[\mathsf{Exp}_{\mathcal{A},\mathbb{P}_{dummy},\mathbb{P}_t}^{\mathsf{num},1} = 0\right] = \sum_{m \in M_c} (\Pr[X = m] - \Pr[X = m-1])$$

Let $M_0 := \{m \in [a, \infty) \colon \Pr[X = m] - \Pr[X = m-1] \geq 0\}$. In particular, for m_{max} as the mode of $\mathbb{P}_{dummy}^{\theta}$, it holds $M_0 = [x, m_{max}]$; then,

$$\begin{aligned} \delta_{num} &= \Pr\left[\mathsf{Exp}_{\mathcal{A},\mathbb{P}_{dummy},\mathbb{P}_t}^{\mathsf{num},0} = 0\right] - \Pr\left[\mathsf{Exp}_{\mathcal{A},\mathbb{P}_{dummy},\mathbb{P}_t}^{\mathsf{num},1} = 0\right] \\ &\leq \sum_{m=x}^{m_{max}} \Pr[X = m] - \Pr[X = m-1] \\ &= \Pr[X = m_{max}] =: \delta_{num}^{u} \end{aligned} \tag{1}$$

Hence, the upper bound δ^u of δ, so that the KTV-Helios scheme ensures δ^u-participation privacy, can be calculated as $\Pr[X = m_{max}]$.

It further follows, that an adversary who is instructed to always output $\beta = 1$ if for the output value of m it holds that $P(X = m) - P(X = m-1) > 0$, guesses β correctly with an advantage of δ^u. Hence, the adversarial advantage $\delta_{num,\mathbb{P}_{dummy},\mathbb{P}_t}$ in $\mathsf{Exp}_{\mathcal{A},\mathbb{P}_{dummy},\mathbb{P}_t}^{\mathsf{num},\beta}$ equals δ^u and thus equals $\Pr[X = m_{max}]$. $\square$