Security Proofs for Participation Privacy, Receipt-Freeness, Ballot Privacy, and Verifiability Against Malicious Bulletin Board for the Helios Voting Scheme

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. However, depending on its version, the scheme provides either participation privacy (hiding who participated in the election) or verifiability against malicious bulletin board (preventing election manipulation by ballot stuffing), but not both at the same time. It also does not provide receipt-freeness, thus enabling vote buying by letting the voters contstruct receipts proving how they voted. Recently, an extension to Helios, further referred to as KTV-Helios, has been proposed that claims to provide these additional security properties. However, the authors of KTV-Helios did not prove their claims. Our first contribution is to provide formal definition for participation privacy and receipt-freeness, that can be applied to KTV-Helios. These definitions were used to also prove the corresponding claims of KTV-Helios. Our second contribution is to use the existing definitions of ballot privacy and verifiability against malicious bulletin board as applied to Helios in order to prove that both security properties also hold for KTV-Helios.

1 Introduction

The Helios voting scheme has been introduced in [1] and subsequently implemented and used in several real-world elections such as the IACR elections [2]. Moreover, the research conducted on Helios led to the development of several extensions for the scheme [3–9], formal security definitions and proofs [3, 10–12] and usability evaluations [13, 14]. Due to these numerous scientific extensions and evaluations, the Helios scheme can be cosnidered one of the most eveolved e-voting scheme which provides ballot privacy and end-to-end verifiability. However, the current implementation of Helios does not provide verifiability against malicious bulletin board that can add or modify ballots on behalf of the voters who do not perform the necessary verification procedures. The extension proposed in [3], called Belenios, solves this issue by introducing digital signatures

thus providing such verifiability against malicious bulletin board. Belenios, however, does not ensure participation privacy, meaning that the public available election data reveals whether a honest voter cast a vote or abstained. Although this information is usually potentially available in traditional paper-based elections, whereby anyone can observes people going into a polling station, an Internet voting system without participation privacy reveals the identities of the voters who cast their vote in an election on a much larger scale by publishing them online. Hence, the lack of participation privacy in Interent voting is a violation of voter privacy that is more serious in comparison to paper-based elections. A further issue with voter privacy in Helios is the lack of receipt-freeness, that enables voters constructing receipts that prove to a third party which candidate the voter has voted for. Thus, such receipts could be used for vote buying.

Recently an extension to Helios has been proposed [15] (henceforth referred to as KTV-Helios) that adds probabilistic participation privacy and probabilistic receipt-freeness to the Helios voting scheme while, at the same time, ensuring verifiability against malicious bulletin board, assuming a reliable public-key infrastructure is in place. However, despite their conceptual contributions to the Helios scheme, the authors of [15] did not actually formally prove the security of their scheme. Furthermore, providing such proofs for KTV-Helios requires introducing new formal definitions for participation privacy as well as receipt-freeness: Although the existing formal definitions of ballot privacy can be extended and applied for evaluating participation privacy in some voting systems, no definition that adresses participation privacy specifically has been proposed, yet. The available definitions of receipt-freeness, on the other hand, do not fully encompass the available e-voting schemes and security models that ensure receipt-freeness.

Our contributions. The main contributions of our paper are new formal definitions for probabilistic participation privacy (see Section 3) and probabilistic receipt-freeness (see Section 4), that we use to apply to KTV-Helios and evaluate its security claims. In addition, we prove that KTV-Helios ensures ballot privacy according to the definition in [11] in the random oracle model (see Section 5). We further prove that the KTV-Helios scheme provides verifiability against malicious bulletin board based on the definition in [3] (see Section 6).

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, the voting devices used by the voters are trustworthy and that the bulletin board provides a consistent review to all the voters and the voting system entities.

Ballot privacy: Given the public data of the election (incl. the election result), the adversary should be incapable of gaining more information about an

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

individual honest voter's vote than is leaked by the election result. This should not require further security assumptions other 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.

2 Description of KTV-Helios

We first describe the version of the *Helios* scheme, based upon the improvements in [3, 4, 10], that KTV-Helios extends upon. In this version, the eligible voters exchange their public signing keys with the registration authority, who then publishes these keys. In the setup phase, the tabulation tellers generate a pair of ElGamal keys used for encrypting the votes. During the voting, the voters encrypt and sign their chosen voting option, also computing the well-formedness proof. The voters then have an option either to audit the encrypted vote, or to submit it to the bulletin board. During the tallying, after the voting is finished, the encrypted votes are being anonymised, either via mix net shuffle or homomorphic tallying. The result of the anonymisation is being jointly decrypted by the tabulation trustees, and published as the outcome of the election.

The basic idea of *KTV-Helios* is the introduction of so-called dummy ballots, that are meant to obfuscate the presence of ballots cast by the voters[5]. 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 a null vote 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 a null vote, this final ballot encrypts the sum of all non-dummy votes cast by the voter. The final ballots of all voters are being anonymised via shuffling. Afterwards, each anonymised ballot is assigned to a valid voting option, or discarded without revealing its plaintext value.

[5] A similar concept of dummy ballots has also been used in [16] which extends the JCJ/Civitas voting scheme [17]

For the sake of simplicity, we assume a single tabulation teller and a single posting trustee.

2.1 Building Blocks of KTV-Helios

In this section, we describe the building blocks (i.e. the cryptographic primitives, probability distributions and plaintext tally function) of the KTV-Helios scheme. The scheme uses the following *cryptographic primitives*:

- Signed ElGamal [10], 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 [18]. 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. [11]).
- 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 $s = \mathsf{Sign}(\mathsf{sk}_s, g_1||g_2||h_1||h_2||t)$[6], 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 [21].
- 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 next building blocks are the *probability distributions*. They are used by the posting trustees in order 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 $\mathbb{P}_d$ and $\mathbb{P}_t$. The first probability distrubition $\mathbb{P}_d$ 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., $x \in \mathbb{N}_0$, $y \in \mathbb{N}_0 \cup \{\infty\}$). The parameters x and y, as well as the exact $\mathbb{P}_d$ needs to be defined by the election authorities when setting up a corresponding system, i.e. their optimal trade-off between

[6] Methods for proving the knowledge of a digital signatures via Σ-proof are described by Asokan et al. [19] for common signature schemes; the general method of constructing NIZK disjunctive proofs is described by Cramer et al. in [20].

security and efficiency. For further information which influence the selection of $\mathbb{P}_d$ has to the level of security and the efficiency of the tallying algorithms, see Section 3. 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 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[7]. 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 $I \cup \{\hat{id}\}$, 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:

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

2.2 Formal Description of KTV-Helios

We are now ready to provide the formal description of the KTV-Helios scheme. This description is based upon the syntax proposed in [11], adjusted to the

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

context of the KTV-Helios scheme. For the sake of simplicity, we assume a single tabulation teller and a single posting trustee[8]. We first specify the various functions in place, i.e.:

- $\mathsf{RegisterVoter}(1^\lambda, id)$ is run by the voter id. The voter id generates a pair of keys $(\mathsf{pk}_{id}, \mathsf{sk}_{id}) \leftarrow_\$ \mathsf{SigKeyGen}(1^\lambda)$ and sends the public key pk_{id} to the registration authority.
- $\mathsf{RegisterRA}(1^\lambda, id, \mathsf{pk}_{id})$ is run by the registration authority. The registration authority adds (id, pk_{id}) to the list of registered voters' public keys I_{pk} if $id \in I$, and returns $\perp$ otherwise.
- $\mathsf{Setup}(1^\lambda)$ 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. Note, that the challenges used in π_{PoK} and π should include the statements and commitments from both π_{PoK} and π in order to prevent that the voter signs and casts the ballot she did not compute herself.
- $\mathsf{Validate}(b)$ 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 $m \leftarrow_\$ \mathbb{P}_d$ 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.

[8] We discuss extending the proofs towards several of those entities in Appendix A.

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

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 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.

These functions are combined in order to build the KTV Helios scheme. The corresponding description of the KTV Helios scheme is given in the following paragraphs along the line of the three phases of an election.

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 register themselves by running $\mathsf{RegisterVoter}(1^\lambda, id)$. After the voters have registered, or if there is an existing PKI already established among the voters, the registration authority prepares the list of the eligible voters' public keys by running $\mathsf{RegisterRA}(id, \mathsf{pk}_{id})$ for each voter id and publishing the list $I_{pk} = \{(id_1, \mathsf{pk}_{id_1}), ..., (id_N, \mathsf{pk}_{id_N})\}$. 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 Participation Privacy

We first provide a cryptographic definition of probabilistic participation privacy. In order to enable the evaluation of participation privacy in KTV-Helios, we chose to propose a quantitative definition. The definition is inspired by the coercion resistance definition in [22] and the verifiability definition in [23]. Similar

to the notion of (γ_k, δ)-verifiability with quantitative goal γ_k in [23], we speak of (δ, k)-participation privacy, where δ denotes the advantage of the adversary who tries to tell whether a given voter has abstained from casting her vote in the election, or cast her vote at most k times.

Defining (δ, k)-participation privacy. We define participation privacy via modeling the experiment, where the actions of two honest voters id_0, id_1 (that is, their decision to abstain or to cast a vote) are swapped. Namely, we consider the following experiment $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},k}$ 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. The adversary only sees the public output for one of these bulletin boards BB_β, $\beta \leftarrow_\$ \{0,1\}$. Let Q_S be a set of oracle queries which 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 the votes for the voters id_0, id_1: given a number of voting options $v_1, ..., v_{k'}$ chosen by the adversary, $k' \leq k$, for each $i = 0, 1$, the bulletin board BB_i contains the votes for $v_1, ..., v_{k'}$ on behalf of id_i and an abstention from the election is modelled for the voter id_{1-i}.

The oracle computes the tally result R on BB_0. In case a voting scheme provides auxiliary output Π for the tally, the oracle returns (R, Π) in case $\beta = 0$, and simulates the auxiliary output $\Pi' = \mathsf{SimProof}(\mathsf{BB}_1, R)$, returning the tuple (R, Π') in case $\beta = 1$[10]. The adversary further outputs the public output of BB_β. The goal of the adversary is to guess whether the provided output corresponds to BB_0 or to BB_1, i.e. to output β. The definition of (δ, k)-participation privacy is then as follows:

Definition 1. *The voting scheme $\mathcal{S}$ achieves (δ, k)-participation privacy given a subset of PPT adversaries C_S, if for any adversary $\mathcal{A} \in C_S$, $k \in \mathbb{N}$ and two honest voter id_0, id_1 holds*

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

is negligible in the security parameter.

As an example of applying the definition, consider a voting scheme that assigns a random unique pseudonym to each voter and publishes the encrypted cast votes next to the voters pseudonyms. The assignment of pseudonyms is assumed to be known only to a trustworthy registration authority, and only the registration authority and an honest voter knows what her pseudonym is. Hence, as the adversary who only has access to the public output cannot establish a link between the pseudonym and the voter's identity, she has no advantage in

[10] The tally result should be the same, if the vote of each voter is equally included in the result. However, in order to be able to model the voting schemes where the weight of the vote might depend on the voter's identity, we chose to simulate the auxiliary output in our definition.

distinguishing between the output of $\mathsf{Exp}^{\mathsf{ppriv},0}_{\mathcal{A},\mathcal{S},k}$ and $\mathsf{Exp}^{\mathsf{ppriv},1}_{\mathcal{A},\mathcal{S},k}$ for any value of k. Thus, the scheme provides $(0,k)$-participation privacy.

(δ,k)-participation privacy in KTV-Helios. In order to evaluate (δ,k)-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. We make following assumptions regarding adversarial capabilities: the tabulation teller does not divulge her secret key to the adversary, the adversary is incapable of observing the communication channel between the voter, the posting trustee and the voting system, the posting trustee does not divulge private information to the adversary, the bulletin board acts according to its specification, and the honest voters decide to participate or to abstain in the election independently from each other.

Hence, we define C_S as a set of adversaries that are given access to the following queries in the experiment $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},k}$:

$\underline{\mathcal{O}\mathsf{VoteAbstain}(v_1,...,v_{k'})}$:
if $k' > k$ then
  return $\perp$
endif
$b_{0,1},...,b_{0,m_0} \leftarrow_{\$} \mathsf{VoteDummy}(id_0)$
$b_{1,1},...,b_{1,m_1} \leftarrow_{\$} \mathsf{VoteDummy}(id_1)$
for $j = 1,...,k'$ do
  $t'_j \leftarrow_{\$} \mathbb{P}_t$
  $b'_{0,j} = \mathsf{Vote}((id_\beta, \mathsf{sk}_{id_0}), id_0, v_j, t'_j)$
  $b'_{1,j} = \mathsf{Vote}((id_\beta, \mathsf{sk}_{id_1}), id_1, v_j, t'_j)$
endfor
Append $b_{0,1},...,b_{0,m_0}$ to BB_0
Append $b_{1,1},...,b_{1,m_1}$ to BB_1
Append $b'_{0,1},...,b'_{0,k'}$ to BB_0
Append $b'_{1,1},...,b'_{1,k'}$ to BB_1

$\underline{\mathcal{O}\mathsf{Cast}(b)}$:
if $\mathsf{Valid}(\mathsf{BB}_\beta, b)$ then
  Append b to BB_0
  Append b to BB_1
endif

$\underline{\mathcal{O}\mathsf{Tally}()}$:
if $\beta = 0$ then
  return $\mathsf{Tally}(\mathsf{sk}, \mathsf{BB}_0)$
else
  $(R, \Pi) = \mathsf{Tally}(\mathsf{sk}, \mathsf{BB}_0)$
  $\Pi' = \mathsf{SimTally}(\mathsf{BB}_1, R)$
endif
return (R, Π')

One source of information that can be used by the adversary for guessing β is k' additional ballots on the bulletin board BB_1 as the output of $\mathcal{O}\mathsf{VoteAbstain}(v_1,...,v_{k'})$. In order to account for the adversarial advantage gained from this number, we define the following experiment $\mathsf{Exp}^{\mathsf{num},\beta}_{\mathcal{A},\mathcal{S},\mathbb{P}_d,\mathbb{P}_t,k'}$, with $\beta = i \in \{0,1\}$ if the voter id_i abstains and the voter id_{1-i} casts k' ballot in the election.

$\underline{\mathsf{Exp}^{\mathsf{num},\beta}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t,k}}$:
$m \leftarrow_{\$} \mathbb{P}_d$
$m_\beta = m + k$
$m_{1-\beta} \leftarrow_{\$} \mathbb{P}_d$
$t_{0,1},...,t_{m_0}, t_{m_0+1},...,t_{m_0+m_1} \leftarrow_{\$} \mathbb{P}_t$
return $m_0, m_1, t_{0,1},...,t_{m_0}, t_{m_0+1},...,t_{m_0+m_1}$

Let $\delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t}$ denote an advantage in this experiment, so that $|\Pr\left[\mathsf{Exp}^{\mathsf{num},0}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t,k}=0\right]-\Pr\left[\mathsf{Exp}^{\mathsf{num},1}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t,k}=0\right]-\delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t}|$ is negligible[11]. We are now ready to evaluate (δ,k)-participation privacy, for KTV-Helios.

Theorem 1. *KTV-Helios, instantiated with the probability distributions $\mathbb{P}_d,\mathbb{P}_t$ achieves (δ,k)-participation privacy for a given $k>0$ given the subset of adversaries C_S, with $\delta=\max_{k'\leq k}\delta^{\mathsf{num}}_{k',\mathbb{P}_d,\mathbb{P}_t}$. It further does not achieve (δ',k)-participation privacy for any $\delta'<\delta$.*

We provide the proof for this theorem in Appendix D.

4 Receipt-Freeness

The KTV-Helios scheme ensures probabilistic receipt-freeness via deniable vote updating. The principle of deniable vote updating has also been proposed in other e-voting schemes [24–26] in order to prevent a voter from constructing receipts that show how the voter has voted. As such, the voter can cast her vote for the voting option the adversary instructs to vote for, but due to deniable vote updating the voter can change her vote without the adversary knowing it. The variant of deniable vote updating used in KTV-Helios is also characterised by enabling the so-called preliminary deniable vote updating. Given two ballots $b_{\mathcal{A}}$, b_v, with $b_{\mathcal{A}}$ as the ballot with the vote for a candidate demanded by the adversary, and b_v the ballot that changes $b_{\mathcal{A}}$ to a vote for a candidate chosen by the voter, the voter can cast $b_{\mathcal{A}}$ and b_v in any order. This approach prevents an attack, where the voter succeeds to cast $b_{\mathcal{A}}$ as the last ballot in the election, thus making sure that her vote has not been updated. However, in KTV-Helios, constructing b_v requires the knowledge of a vote that was cast with $b_{\mathcal{A}}$.

Defining δ-receipt-freeness. We propose a formal definition for probabilistic receipt-freeness for e-voting schemes with deniable vote updating. Hereby we employ the δ-notation similar to the definition of (δ,k)-participation privacy and define δ-receipt-freeness. We base our definitions on the definition of receipt-freeness by Cortier et al. [5]. However, as opposed to their definition, and similar to a probabilistic definition of coercion resistantce by Kuesters et al. in [22], we consider vote buying from a single voter, while considering an extension towards multiple voters in future work. We further adjust the definition by Cortier et al. by enabling the voter to apply a counter-strategy against an adversary that demands a receipt, namely, to deniably update her vote.

We define an experiment $\mathsf{Exp}^{\mathsf{rfree},\beta}_{\mathcal{A},\mathcal{S}}$ for a voting scheme $\mathcal{S}$ as follows. The challenger sets up two bulletin boards BB_0, BB_1 and simulates the election setup. The challenger further sets $\beta=0$ to represent the voter following the adversarial instructions and $\beta=1$ to represent the voter deniably updating her vote. The adversary has access to following queries, whereby she is allowed to query

[11] We show how to calculate $\delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t}$ for some choices of $\mathbb{P}_d$ and $\mathbb{P}_t$ in Appendix G.

$\mathcal{O}\mathsf{Receipt}(id, v_0, v_1, t)$ and $\mathcal{O}\mathsf{Tally}()$ only once:

$\underline{\mathcal{O}\mathsf{Cast}(b)}$:
if $\mathsf{Valid}(\mathsf{BB}_\beta, b)$ then
    Append b to BB_0
    Append b to BB_1
endif

$\underline{\mathcal{O}\mathsf{Receipt}(id, v_0, v_1, t)}$:
if $v_0 \notin \mathbb{V}_{valid}$ or $v_1 \notin \mathbb{V}_{valid}$ then
    return $\perp$
endif
$b_{\mathcal{A}} = \mathsf{Vote}(id, \mathsf{sk}_{id}, v_0, t)$
Append $b_{\mathcal{A}}$ to BB_0
Append $b_{\mathcal{A}}$ to BB_1
$t_v \leftarrow\!\!\$\ \mathbb{P}_t$
$b_v = \mathsf{DeniablyUpdate}(id, \mathsf{sk}_{id}, v_0, v_1, t_v)$
Append b_v to BB_1
$\mathsf{Obfuscate}(BB_0, id)$
$\mathsf{Obfuscate}(BB_1, id)$

$\underline{\mathcal{O}\mathsf{VoteLR}(id, v_0, v_1, t)}$:
$b_0 = \mathsf{Vote}((id, \mathsf{sk}_{id}), id, v_0, t)$
$b_1 = \mathsf{Vote}((id, \mathsf{sk}_{id}), id, v_1, t)$
if $\mathsf{Valid}(\mathsf{BB}_\beta, b_\beta) = 0$ then
    return $\perp$
endif
Append b_0 to BB_0
Append b_1 to BB_1

$\underline{\mathcal{O}\mathsf{Tally}()}$:
if $\beta = 0$ then
    return $\mathsf{Tally}(\mathsf{sk}, \mathsf{BB}_0)$
else
    $(R, \Pi) = \mathsf{Tally}(\mathsf{sk}, \mathsf{BB}_0)$
    $\Pi' = \mathsf{SimTally}(\mathsf{BB}_1, R)$
endif
return (R, Π')

Intuitively, the definition encompasses the scenario of vote buying, whereby the adversary tells the voter the name of the candidate the voter has to provide a receipt for, and the voter is able to access the random coins used in creating an adversarial ballot $b_{\mathcal{A}}$. It, however, does not cover the scenarios where the adversary wants to make sure the voter did not cast a valid vote in the election, or to change the voter's vote to a random candidate (forced abstention and randomization as described in [27]). It also does not consider the information leakage from the election result. We now define δ-receipt-freeness for deniable vote updating:

Definition 2. *The voting scheme $\mathcal{S}$ achieves δ-receipt-freeness, if there are algorithms* SimProof*,* DeniablyUpdate*,* Obfuscate *so that*

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

is negligible in the security parameter.

δ-receipt-freeness in KTV-Helios. In order to evaluate δ-receipt-freeness for the KTV-Helios scheme, we define the algorithm $\mathsf{DeniablyUpdate}(id, \mathsf{sk}_{id}, v_0, v_1, t_v)$ as casting a ballot for v_1/v_0: that is,

$$\mathsf{DeniablyUpdate}(id, \mathsf{sk}_{id}, v_0, v_1, t_v) = \mathsf{Vote}((id, \mathsf{sk}_{id}), id, v_1/v_0, t_v)$$

The assumptions regarding adversarial capabilities for receipt-freeness in KTV-Helios are then as follows: the tabulation teller does not divulge her secret key to the adversary, the adversary is incapable of observing the communication

channel between the voter, the posting trustee and the voting system, the posting trustee does not divulge private information to the adversary, the bulletin board acts according to its specification, the voter is capable of casting a vote without being observed by the adversary, the voters who are required by the adversary to provide receipts act independent from each other and the adversary does not cast ballots on behalf of the voter, which plaintexts the voter does not know. The last assumption relies the voter not divulging her secret key to the adversary, which is justified if the secret key is also used for purposes other than voting, e.g. as a part of eID infrastructure, in which case divulging it to the adversary would incur larger losses to the voter than she would gain from selling her vote. It further relies on on the absense of two-way communication between the voter and the adversary during casting the ballot, which we consider unlikely in large-scale vote buying.

For finding an appropriate value of δ we need to account for the adversarial advantage gained from the number of ballots next to voter's id on the bulletin board. For this purpose, we define the following experiment $\mathsf{Exp}^{\mathsf{rfnum},\beta}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t}$, where the challenger sets $\beta = 0$ if the voter *id* does not cast any additional ballot and $\beta = 1$ if she casts an additional ballot that deniably updates her vote:

$\underline{\mathsf{Exp}^{\mathsf{rfnum},\beta}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t}:}$
$m \leftarrow\!\!\$\ \mathbb{P}_d$
$t_1, ..., t_m, t_{m+\beta} \leftarrow\!\!\$\ \mathbb{P}_t$
return $m + \beta, t_1, ..., t_m, t_{m+\beta}$

Let $\delta^{\mathsf{rfnum}}_{\mathbb{P}_d,\mathbb{P}_t}$ denote an advantage in this experiment, so that $\Pr\left[\mathsf{Exp}^{\mathsf{rfnum},0}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t} = 0\right] - \Pr\left[\mathsf{Exp}^{\mathsf{rfnum},1}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t} = 0\right] - \delta^{\mathsf{rfnum}}_{\mathbb{P}_d,\mathbb{P}_t}$ is negligible. We are now ready to provide an evaluation of δ-receipt-freeness for KTV-Helios.

Theorem 2. *KTV-Helios, instantiated with probability distributions $\mathbb{P}_d, \mathbb{P}_t$, achieves δ-receipt-freeness privacy given the algorithms* SimProof*,* DeniablyUpdate*,* Obfuscate*, with $\delta = \delta^{\mathsf{rfnum}}_{\mathbb{P}_d,\mathbb{P}_t}$. It further does not achieve δ'-participation privacy for any $\delta' < \delta$.*

We provide the proof of this theorem in Appendix E.

5 Ballot Privacy

In this section we prove ballot privacy (BPRIV) for the KTV-Helios scheme following the defintion in [11]. 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 [11] that both the tabulation teller and the bulletin board are honest, which corresponds to our informal definition in the introduction of this paper.

5.1 Purpose and Definition of BPRIV

We adjust the definition proposed by Bernhard et al. [11] – 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 [11] 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 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[12]. 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 2.2 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 $\mathsf{Valid}(\mathsf{BB}_\beta, b) = 1$,

[12] 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.

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[13] 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 $\mathsf{Append}(\mathsf{BB}_1, b_1)$.
- $O\mathsf{Tally}()$: 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 3. *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).

5.2 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.

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

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 3.

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 3 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 $f_R : V \cup \{\perp\} \to \{0, 1, \ldots, n\}$ where for example $f_R(v) = 3$ means that three voters voted for v and $f_R(\perp)$ is the number of voters who cast an invalid vote. We have $f_R(\perp) + \sum_{v \in V} f_R(v) = n$, 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 $O\mathsf{VoteLR}$ 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 3. *KTV-Helios satisfies the BPRIV security definition.*

5.3 Strong Correctness and Strong Consistency

Together with BPRIV, [11] 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 [11, Def. 9] to include these elements.

Definition 4. *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 2.2 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 4. □

The definition of strong correctness may seem tautological (and the proof trivial) but it prevents the following counter-example from [11, 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 5. *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* $\mathsf{ValidInd}$*: 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. □

6 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. We accept the following assumptions: the register of eligible voters and the PKI is trustworthy, and the honest voters' secret keys and the signatures on ballots are not leaked to the adversary. The latter assumption relies on the trustworhiness of the voting devices, which, however, might be realistically expected in some settings, e.g. in case a national eID infrastructure with tamper-resistant smartcards is in place.

For the actual proof, we rely on the 'verifiability against a malicious bulletin board' framework definition for Helios alike schemes of [3] which matches the verifiability definition in our introduction. We adjust the definition in [3] 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 2.2 on behalf of the election organizer, registration authority and eligible voters. 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:

$\underline{\mathcal{O}\mathsf{Vote}(id', id, v, t)}$:
if $id \notin I \cup \{\hat{id}\}$ or $id \in I^C$
  return $\perp$
endif
$b = \mathsf{Vote}((id', \mathsf{sk}_{id'}), id, v, t)$
Append b to BB

$\underline{\mathcal{O}\mathsf{Cast}(b)}$:
Append b to BB

$\underline{\mathcal{O}\mathsf{Corrupt}(id)}$:
if $id \in I^C$ then
  return $\perp$
endif
Add id to I^C
Remove all tuples $(id, *, *)$ from HVote
return sk_{id}

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 $\mathcal{O}\mathsf{Vote}(id, id, v, t)$, 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:

Case 1: 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.

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

holds; while the list of tuples $(id_{E,i}, v_{E,i})$ were cast by honest voters (i.e. it holds, $(id_{E,i}, v_{E,i}, *) \in \mathsf{HVote}$ 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 $|I^C|$.

Case 2: 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 against a malicious bulletin board.

Definition 6. *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 4. *KTV-Helios provides verifiability against a malicious bulletin board.*

The proof is provided in Appendix F.

7 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 [11], and a framework that proposes a uniform treatment of the verifiability definitions from [3,28–31] is described in [23]. Other approaches for defining and evaluating the security of voting schemes include applied pi-calculus [32–34], process algebra [35] or k-resilience terms [36]. These approaches have been applied to evaluate various voting schemes [12, 34, 36, 37]. In particular, the formal security analysis of Helios has been the topic of [3, 11, 12].

Due to its versatility, various extensions and modifications of the original Helios scheme [1] have been proposed: A modification in [10] fixes the vulnerabilities in [1]. Zeus [9], Selene [8] or the work by Guasch et al. [7] propose alternatives to the cast-as-intended mechanism in Helios. Improvements of various security

properties were proposed in [6] (ensuring long-term ballot privacy), in [3] (ensuring verifiability against malicious bulletin board), in [4] (introducing threshold decryption for better robustness); and in [5] (ensuring receipt-freeness). Other research focused on improving the usability of Helios [13, 14].

8 Conclusion and Future Work

We have evaluated the security properties of the KTV-Helios extension proposed in [15]. Namely, we have proven that KTV-Helios satisfies ballot privacy and verifiability against a malicious bulletin board according to the definitions from [3, 11] adjusted to the context of KTV-Helios. Furthermore, we proposed a probabilistic abstract definition of (δ, k)-participation privacy, with δ representing the adversarial advantage in distinguishing whether a particular honest voter has cast up to k ballots in the election. We also proposed a probabilistic abstract definition of δ-receipt-freeness for voting schemes based on deniable vote updating. We proposed instantiations of both (δ, k)-participation privacy and δ-receipt-freeness defintions for KTV-Helios and determined the value of δ as the adversarial advantage for both of these properties.

We plan to extend the proofs in this paper for the case where the tabulation teller and the posting trustee are implemented in a distributed way. We further plan to adress the existing security and efficiency issues of KTV-Helios, such as the possibility of board flooding and the neccessity of trusting the device that holds the secret key for integrity, as well as its usability.

Acknowledgements

We would like to thank Vincent Cheval, Véronique Cortier, Marc Fischlin, Vanessa Teague, Tomasz Trudering and Reto Koenig 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. B. Adida, "Helios: Web-based open-audit voting," in *SS 2018: 17th Conference on Security Symposium*. USENIX, July 2008, pp. 335–348.
2. IACR, "IACR Election 2016," http://www.iacr.org/elections/2016, 2016, [Online; accessed 9-November-2016].
3. V. Cortier, D. Galindo, S. Glondu, and M. Izabachène, "Election verifiability for helios under weaker trust assumptions," in *ESORICS 2014: 19th European Symposium on Research in Computer Security, Part II*, ser. Lecture Notes in Computer Science. Springer, September 2014, pp. 327–344.

4. V. Cortier, D. Galindo, S. Glondu, and M. Izabachène, "Distributed elgamal à la pedersen: Application to helios," in *WPES 2013: 12th ACM Workshop on Workshop on Privacy in the Electronic Society*. ACM, November 2013, pp. 131–142.
5. V. Cortier, G. Fuchsbauer, and D. Galindo, "Beleniosrf: A strongly receipt-free electronic voting scheme," Cryptology ePrint Archive, Report 2015/629, June 2015, http://eprint.iacr.org/.
6. D. Demirel, J. Van De Graaf, and R. S. dos Santos Araújo, "Improving helios with everlasting privacy towards the public," in *EVTW/WTE 2012: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX, August 2012.
7. S. Guasch and P. Morillo, "How to challenge and cast your e-vote," in *FC 2016: 20th international conference on Financial Cryptography and Data Security*. IFCA, February 2016.
8. P. Y. A. Ryan, P. B. Roenne, and V. Iovino, "Selene: Voting with transparent verifiability and coercion-mitigation," Cryptology ePrint Archive, Report 2015/1105, November 2015, http://eprint.iacr.org/.
9. G. Tsoukalas, K. Papadimitriou, P. Louridas, and P. Tsanakas, "From helios to zeus," in *EVTW/WTE 2013: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX, August 2013.
10. D. Bernhard, O. Pereira, and B. 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*, ser. Lecture Notes in Computer Science. Springer, December 2012, pp. 626–643.
11. D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi, "Sok: A comprehensive analysis of game-based ballot privacy definitions," in *IEEE S&P 2015: IEEE Symposium on Security and Privacy*. IEEE, May 2015, pp. 499–516.
12. S. Kremer, M. Ryan, and B. Smyth, "Election verifiability in electronic voting protocols," in *ESORICS 2010: 15th European Symposium on Research in Computer Security*, ser. Lecture Notes in Computer Science. Springer, September 2010, pp. 389–404.
13. S. Neumann, M. M. Olembo, K. Renaud, and M. Volkamer, "Helios verification: To alleviate, or to nominate: Is that the question, or shall we have both?" in *EGOVIS 2014: Electronic Government and the Information Systems Perspective: Third International Conference*, ser. Lecture Notes in Computer Science. Springer, September 2014, pp. 246–260.
14. F. Karayumak, M. Kauer, M. M. Olembo, T. Volk, and M. Volkamer, "User study of the improved helios voting system interfaces," in *STAST 2011: 1st Workshop on Socio-Technical Aspects in Security and Trust*. IEEE, September 2011, pp. 37–44.
15. O. Kulyk, V. Teague, and M. Volkamer, "Extending helios towards private eligibility verifiability," in *VoteID 2015: E-Voting and Identity, 5th International Conference*, ser. Lecture Notes in Computer Science. Springer, September 2015, pp. 57–73.
16. O. Spycher, R. Koenig, R. Haenni, and M. Schläpfer, "A new approach towards coercion-resistant remote e-voting in linear time," in *FC 2011: Financial Cryptography and Data Security, 15th International Conference, Revised Selected Papers*. Springer, Mar. 2011, pp. 182–189.
17. M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a Secure Voting System." in *IEEE S&P 2008: IEEE Symposium on Security and Privacy*. IEEE Computer Society, May 2008, pp. 354–368.

18. D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology — CRYPTO' 92: 12th Annual International Cryptology Conference*, ser. Lecture Notes in Computer Science. Springer, August 1993, pp. 89–105.
19. N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in *Advances in Cryptology — EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques*, ser. Lecture Notes in Computer Science. Springer, June 1998, pp. 591–606.
20. R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology — CRYPTO '94: 14th Annual International Cryptology Conference*, ser. Lecture Notes in Computer Science. Springer, August 1994, pp. 174–187.
21. B. Terelius and D. Wikström, "Proofs of restricted shuffles," in *Progress in Cryptology – AFRICACRYPT 2010: Third International Conference on Cryptology in Africa*, ser. Lecture Notes in Computer Science. Springer, May 2010, pp. 100–113.
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*. IEEE, July 2010, pp. 122–136.
23. V. Cortier, D. Galindo, R. Kuesters, J. Mueller, and T. Truderung, "Verifiability notions for e-voting protocols," Cryptology ePrint Archive, Report 2016/287, March 2016, http://eprint.iacr.org/.
24. P. Locher and R. Haenni, "Receipt-free remote electronic elections with everlasting privacy," *Annals of Telecommunications*, vol. 71, no. 7, pp. 323–336, 2016.
25. P. Locher, R. Haenni, and R. E. Koenig, "Coercion-resistant internet voting with everlasting privacy," in *FC 2016: International Workshops, BITCOIN, VOTING, and WAHC, Revised Selected Papers*. Springer, 2016.
26. D. Achenbach, C. Kempka, B. Löwe, and J. Müller-Quade, "Improved coercion-resistant electronic elections through deniable re-voting," *JETS 2015: USENIX Journal of Election Technology and Systems*, pp. 26–45, 2015.
27. A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *WPES 2005: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*. ACM, 2005, pp. 61–70.
28. R. Küsters, T. Truderung, and A. Vogt, "Accountability: Definition and relationship to verifiability," in *CCS 2010: 17th ACM Conference on Computer and Communications Security*. ACM, April 2010, pp. 526–535.
29. J. D. C. Benaloh, "Verifiable secret-ballot elections," *PhD thesis, Yale University, Department of Computer Science*, 1987.
30. A. Kiayias, T. Zacharias, and B. 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*, ser. Lecture Notes in Computer Science. Springer, April 2015, pp. 468–498.
31. B. Smyth, S. Frink, and M. 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/.
32. S. Kremer and M. Ryan, "Analysis of an electronic voting protocol in the applied pi calculus," in *ESOP 2005: Programming Languages and Systems, 14th European Symposium on Programming*, ser. Lecture Notes in Computer Science. Springer, April 2005, pp. 186–200.
33. 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*. IEEE, June 2008, pp. 195–209.

34. S. Delaune, S. Kremer, and M. Ryan, "Verifying privacy-type properties of electronic voting protocols," *Journal of Computer Security*, vol. 17, no. 4, pp. 435–487, 2009.
35. M. Moran, J. Heather, and S. Schneider, "Verifying anonymity in voting systems using csp," *Formal Aspects of Computing*, vol. 26, no. 1, pp. 63–98, December 2012.
36. G. Schryen, M. Volkamer, S. Ries, and S. M. Habib, "A formal approach towards measuring trust in distributed systems," in *SAC 2011: 2011 ACM Symposium on Applied Computing*. ACM, March 2011, pp. 1739–1745.
37. M. Arnaud, V. Cortier, and C. Wiedling, "Analysis of an electronic boardroom voting system," in *Vote-ID 2013: E-Voting and Identify, 4th International Conference*, ser. Lecture Notes in Computer Science. Springer, July 2013, pp. 109–126.

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 $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},k}$ defined Section 3, the query $O\mathsf{VoteAbstain}$ corresponds to a remaining honest posting trustee casting dummy ballots on behalf of each voter. Hence, the definition and proofs from Section 3 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 [11], 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 2.1 has the partial counting property. Let $I = \{id_1, ..., id_N\}$ be a set of voter ids, $\hat{id} \notin I$ 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 $id \in I \cup \{\hat{id}\}$ and $v \in \mathbb{G}_q$.

Let $I_1, .., I_k$ be partitions of $I \cup \{\hat{id}\}$, so that $\bigcup_{i=1}^{k} I_i = I \cup \{\hat{id}\}$ and $I_i \cap I_j = \emptyset$ for all $i \neq j$. We further define the lists $\mathbb{V}^{(i)}_{cast} \subset \mathbb{V}_{cast}$ 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 $\mathbb{V}^{(i)}_{cast}$, $i = 1, ..., k$. Namely, it should hold

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

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 2.1, with $\mathbb{V}$ as a set of tuples $(id, v) \in I \cup \{\hat{id}\} \times \mathbb{G}_1$, 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 Ballot Privacy, Strong Correctness and Strong Consistency

We base the proofs described in this section on the same ideas as [11].

C.1 Ballot Privacy

The core of a BPRIV proof is a simulator 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 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 3. *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 $\mathcal{O}\mathsf{VoteLR}$ and the $\mathcal{O}\mathsf{Cast}$ 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 $\mathcal{O}\mathsf{VoteLR}$. Therefore we conclude that either two ballots created by $\mathcal{O}\mathsf{VoteLR}$ have collided, the probability of which is certainly negligible, or the adversary has submitted in a $\mathcal{O}\mathsf{Cast}$ query a copy of a ciphertext that $\mathcal{O}\mathsf{VoteLR}$ 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 3.

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 3 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 $f_R : V \cup \{\perp\} \to \{0, 1, \ldots, n\}$ where for example $f_R(v) = 3$ means that three voters voted for v and $f_R(\perp)$ is the number of voters who cast an invalid vote. We have $f_R(\perp) + \sum_{v \in V} f_R(v) = n$, 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 5. *KTV-Helios satisfies the BPRIV security definition.*

C.2 Strong Correctness and Strong Consistency

Together with BPRIV, [11] 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 [11, Def. 9] to include these elements.

Definition 7. *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 4. *The voting scheme described in Section 2.2 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 4. □

The definition of strong correctness may seem tautological (and the proof trivial) but it prevents the following counter-example from [11, 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 8. *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. □

D Proof of participation privacy for the KTV-Helios scheme

We base our proof on the idea, that the aforementioned sources of information (i.e. the number of ballots next to id_0 and id_1) is the only ones that give advan-

tage to the adversary. The rest of the public election data, as in case of ballot privacy (as shown in Section 5), does not provide any advantage to the adversary.

Our proof strategy is hence as follows. We consider a sequence of games, starting from $\mathsf{Exp}^{\mathsf{ppriv},0}_{\mathcal{A},\mathcal{S}k}$ and ending with $\mathsf{Exp}^{\mathsf{ppriv},1}_{\mathcal{A},\mathcal{S},k}$ 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 $\delta := \max_{k' \leq k} \delta^{\mathsf{num}}_{k',\mathbb{P}_d,\mathbb{P}_t}$. 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 $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},k}$ with $\beta = 0$, and $v_1, ..., v_{k'} \neq 0$ (hence, it is equivalent to the election where the voter id_0 abstains, and the voter id_1 casts $k' \leq k$ ballots with the votes $v_1, ..., v_{k'}$). 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 $\mathcal{O}\mathsf{Tally}$ at the end of $\mathsf{Exp}^{\mathsf{ppriv},0}_{\mathcal{A},\mathcal{S},k}$.

- G_2**.** The second game G_2 is equivalent to the election, where the voter id_0 abstains, and the voter id_1 casts $k' \leq k$ ballots with a null-vote. The contents of the bulletin board $\mathsf{BB}_{0,2}$ is equivalent to the content of the bulletin board BB_1 at the end of $\mathsf{Exp}^{\mathsf{ppriv},1}_{\mathcal{A},\mathcal{S},k}$ for the adversary using the query $\mathcal{O}\mathsf{VoteAbstain}(v_1, ..., v_{k'})$ with $v = 0$. The tally result R, however, is calculated on the contents of the bulletin board $\mathsf{BB}_{0,1}$ in the game G_1, and the auxiliary output Π_2 is simulated as $\Pi_2 = \mathsf{SimProof}(R_1, \mathsf{BB}_{0,2})$.

We prove, that the adversarial advantage in distinguishing between the output of G_1 and G_2 is at most the adversarial advantage in the ballot privacy experiment (Section 5). Consider an adversary $\mathcal{B}$ in the ballot privacy experiment $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A},\mathcal{S}}$, who simulates the games G_1 and G_2 for the adversary $\mathcal{A}$. The adversary $\mathcal{B}$ returns the output of $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A},\mathcal{S}}$ for the queries $\mathcal{O}\mathsf{Cast}$ and $\mathcal{O}\mathsf{Tally}$. For simulating the output of $\mathcal{O}\mathsf{VoteAbstain}(v_1, ..., v_{k'})$, $\mathcal{B}$ proceeds as follows: First, she simulates the dummy ballots for each voter id_i, $i \in \{0, 1\}$ by choosing a random values $m_i \leftarrow\!\!\$\, \mathbb{P}_d$, and a set of random timestamps $t_1, ..., t_{m_i} \leftarrow\!\!\$\, \mathbb{P}_t$. The dummy ballots $b_{i,1}, ..., b_{i,m_i}$ are computed as $b_{i,j} = \mathsf{Vote}((\hat{id}, 0), id_i, 0, t_j)$, $j = 1, ..., m_i$. Aftwewards, she simulates casting the votes $v_1, ..., v_{k'}$: For each of the votes v_l, $l = 1, ..., k'$, she uses the query $\mathcal{O}\mathsf{VoteLR}(id_1, id_1, 0, v_l, t)$ for a random $t_l \in \mathbb{P}_t$ in $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A},\mathcal{S}}$. The output of the queries $\mathcal{O}\mathsf{VoteLR}$ and the dummy ballots $b_{i,1}, ..., b_{i,m_i}$ is returned to $\mathcal{A}$. At the end, $\mathcal{B}$ returns the value β output by $\mathcal{A}$ as the guess in $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A},\mathcal{S}}$. Thus, it follows that the adversarial advantage in distinguishing G_1 from G_2 is at most equal to the adversarial advantage in $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A},\mathcal{S}}$, denoted as δ_{BPRIV}.

- G_3**.** The third game G_3 is equivalent to the election, where the voter id_0 casts $k' \leq k$ ballots with null-vote, and the voter id_1 abstains from the election. 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 $\mathsf{Exp}^{\mathsf{ppriv},1}_{\mathcal{A},\mathcal{S},k}$ for the adversary using the query $\mathcal{O}\mathsf{VoteAbstain}(v_1, ..., v_{k'})$ with $v_l = 0\ \forall l = 1, ..., k'$, $k' \leq k$. The tally outputs the result R_1 computed on $\mathsf{BB}_{0,1}$ and simulated auxilary data $\Pi_3 = \mathsf{SimProof}(R2, \mathsf{BB}_{0,3})$.

We prove, that the adversary has an advantage of $\max_{k' \leq k} \delta^{\mathsf{num}}_{k',\mathbb{P}_d,\mathbb{P}_t}$ of distinguishing between the output of G_2 and G_3. The tally result does not change, hence the tally output (R_1, Π_2) is equivalent to the tally output (R_1, Π_3). The only difference between the contents of $\mathsf{BB}_{0,1}$ and $\mathsf{BB}_{0,2}$ is the presence of k' additional ballots with the encryption of 0 on $\mathsf{BB}_{0,3}$. Therefore, we conclude that the challenge in distinguishing between the outputs of G_2 and G_3 is equivalent to the challenge in distinguishing between the output of $\mathsf{Exp}^{\mathsf{num},0}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t,k'}$ and $\mathsf{Exp}^{\mathsf{num},1}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t,k'}$ for every $k' \leq k$ chosen by the adversary, and therefore the adversarial advantage of distinguishing between the output of G_1 and G_2 is at most $\max_{k' \leq k} \delta^{\mathsf{num}}_{k',\mathbb{P}_d,\mathbb{P}_t}$.

- G_4. The fourth game G_4 is equivalent to the election where the voter id_0 casts k' ballots with the votes $v_1, ..., v_{k'}$, and the voter id_1 abstains. The tally is computed on $\mathsf{BB}_{0,1}$, and the auxiliary output is simulated as $\Pi_4 = \mathsf{SimProof}(R_1, \mathsf{BB}_{0,4})$. Applying the same argument as for the indistinguishability of G_1 and G_2, it holds that adversary distinghuishes between the outputs of two games with the same advantage as in the ballot privacy experiment, namely δ_{BPRIV}.

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 δ_{BPRIV} (for the games G_1 and G_2, and for the games G_3 and G_4) or $\delta^{\mathsf{num}}_{k',\mathbb{P}_d,\mathbb{P}_t}$ for $k' \leq k$ (for the games G_1 and G_2). Since δ_{BPRIV} is negligible, as proven in Section 5, it holds that the adversary distinguishes between the output in $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},k}$ with the advantage only negligibly larger than $\delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t}$ for each $k' < k$ that she chooses in the experiment. Thus, given that an adversary chooses k' so that $\delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t} \geq \delta_{num,k''}\ \forall k'' \neq k', k'' \leq k$, the adversarial advantage in $\mathsf{Exp}^{\mathsf{ppriv},\beta}_{\mathcal{A},\mathcal{S},k}$ is negligibly larger than $\delta_k := \max_{k' \leq k} \delta^{\mathsf{num}}_{k',\mathbb{P}_d,\mathbb{P}_t}$. □

E Proof of receipt-freeness for the KTV-Helios scheme

As in the case with δ-participation-privacy, we base our proof on the idea, that the number of ballots next to the voter is the only source of information that givens advantage to the adversary.

We consider a sequence of games, starting from $\mathsf{Exp}^{\mathsf{rfree},0}_{\mathcal{A}}$ and ending with $\mathsf{Exp}^{\mathsf{rfree},1}_{\mathcal{A}}$ and show, that the adversary $\mathcal{A}$ distinguishes the transition through all those games with the advantage of at most $\delta^{\mathsf{rfnum}}_{\mathbb{P}_d,\mathbb{P}_t}$. 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 $\mathsf{Exp}^{\mathsf{rfree},\beta}_{\mathcal{A}}$ with $\beta = 0$ (hence, it is equivalent to the election where the voter id does not try to deniably update her vote). 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 $\mathcal{O}\mathsf{Tally}$ at the end of $\mathsf{Exp}^{\mathsf{rfree},0}_{\mathcal{A}}$.
- G_2. The second game G_2 is equivalent to the election, where the voter id casts an additional 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 $\mathsf{Exp}^{\mathsf{rfree},1}_{\mathcal{A}}$ for the adversary using the query $\mathcal{O}\mathsf{Receipt}(id, v_0, v_1, t)$ with $v_0 = v_1$.

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 $\mathsf{Exp}^{\mathsf{rfnum},0}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t}$ and $\mathsf{Exp}^{\mathsf{rfnum},1}_{\mathcal{A},\mathbb{P}_d,\mathbb{P}_t}$, and therefore the adversarial advantage of distinguishing between the output of G_1 and G_2 is $\delta^{\mathsf{rfnum}}_{\mathbb{P}_d,\mathbb{P}_t}$.

- G_3**.** The third game G_3 is equivalent to the election, where the voter cast a vote for a non-null voting option $v \neq 0$, and the tally result R is calculated on the bulletin board $\mathsf{BB}_{0,2}$ with simulated tally proof $\Pi = \mathsf{SimProof}(\mathsf{BB}_{0,3}, R)$.

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 5) $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A},\mathcal{S}}$, who simulates the games G_2 and G_3 for the adversary $\mathcal{A}$. The adversary $\mathcal{B}$ returns the output of $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A}}$ for the queries $\mathcal{O}\mathsf{VoteLR}$, $\mathcal{O}\mathsf{Tally}$. For simulating the output of $\mathcal{O}\mathsf{Receipt}(id, v_0, v_1, t)$, $\mathcal{B}$ proceeds as follows: first, she computes a ballot $b_v = \mathsf{Vote}((id, \mathsf{sk}_{id}), id, v_0, t)$. She then chooses a random value $m \leftarrow_{\$} \mathbb{P}_d$, 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 $\mathcal{O}\mathsf{VoteLR}(id, id, 0, v_1/v_0, t')$ for a random $t' \in \mathbb{P}_t$ in $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A}}$ and returns its output together with the ballots $b_v, b_1, ..., b_m$ to $\mathcal{A}$. At the end, $\mathcal{B}$ returns the value β output by $\mathcal{A}$ as the guess in $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A},\mathcal{S}}$. Thus, it follows that the adversarial advantage in distinguishing G_2 from G_3 is at most equal to the adversarial advantage in $\mathsf{Exp}^{\mathsf{bpriv},\beta}_{\mathcal{A}}$, denoted as δ_{BPRIV}.

It follows, that the in transition through the game sequence $G_1 \rightarrow G_2 \rightarrow G_3$, the outputs of each game are distinguished from the outputs of a previous game with the advantage either $\delta^{\mathsf{rfnum}}_{\mathbb{P}_d,\mathbb{P}_t}$ (for games G_1 and G_2) or δ_{BPRIV} (for games G_2 and G_3). Hence, the adversary distinguishes between the output in $\mathsf{Exp}^{\mathsf{rfree},\beta}_{\mathcal{A}}$ with the advantage at most $\delta = \delta^{\mathsf{rfnum}}_{\mathbb{P}_d,\mathbb{P}_t} + \delta_{BPRIV}$, with δ_{BPRIV} negligible as proven in Section 5. □

F Proof of verifiability for the KTV-Helios scheme

The proof is based on similar ideas as in [3]. We proceed 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 2.2. 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:

- $\mathbb{V}^{HC}_{cast} := ((id_{E,1}, v_{E,1}), ..., (id_{E,n_E}, v_{E,n_E}))$ the list of all tuples of honest voters and non-zero votes (i.e. $((id_{E,i}, v_{E,i}), *) \in$ HVote) who verified that their vote is properly stored on the bulletin board (i.e. $id_{E,i} \in$ Checked).
- $\mathbb{V}^{HU}_{cast} := ((id_{A,1}, v_{A,1}), ..., (id_{A,n_A}, v_{A,n_A}))$, the list of all tuples of honest voters and non-zero votes (i.e. $((id_{A,i}, v_{A,i}), *) \in$ HVote) who did not verify that their vote is properly stored on the bulletin board (i.e. $id_{E,i} \notin$ Checked).
- $\mathbb{V}^{C}_{cast} := ((id_{B,1}, v_{B,1}), ..., (id_{B,n_B}, v_{B,n_B}))$, the list of all tuples of corrupted voters with non-zero votes (i.e. $id_{B,i} \in I^C$) and their votes.
- $\mathbb{V}^{D}_{cast} := \{(*, 0)\}^{n_D}$: 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 $\mathbb{V}^{D}_{cast}$. 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 $(id, v) \in \mathbb{V}^{C}_{cast}$ 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 $id \in$ 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 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 $\mathbb{V}^{HC}_{cast}$. The adversary's ballots are only the ones in $\mathbb{V}^{C}_{cast}$ whose identities are in I^C so the number of these ballots is at most $|I^C|$. All the remaining ballots are in $\mathbb{V}^{HU}_{cast}$ and so must have been cast by non-checking honest voters. Since ρ supports partial counting as expained in Section 2.1 we conclude, for $\mathbb{V}_{cast}$ the list of all votes in ballots on the board:

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

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

□

G Example for (δ, k)-participation privacy in KTV-Helios

We now provide an example of how to quantify (δ, k)-participation privacy given a particular distribution for the number of dummy votes $\mathbb{P}_d$.

Let $\mathbb{P}_d$ be a geometric distribution with the parameter $p \in (0, 1]$, so that the probability $\Pr[X = m] = (1-p)^m p$ for $m \geq 0$ and $\Pr[X = m] = 0$ for $m < 0$. Since the 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.

Let $k > 0$, $M_c \subset \mathbb{N}_0^2$ be a set of all pairs (m_0, m_1) output in $\mathsf{Exp}^{\mathsf{num},\beta}_{\mathcal{A},k}$, for which an adversary guesses $\beta = 0$ (i.e. that $m_0 = m + k$ with $m \leftarrow\!\!\$\, \mathbb{P}_d$, $m_1 \leftarrow\!\!\$\, \mathbb{P}_d$. It holds for $\delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t}$ as defined in Section 3:

$$\begin{aligned}
\delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t} &:= \Pr\left[\mathsf{Exp}^{\mathsf{num},0}_{\mathcal{A},k} = 0\right] - \Pr\left[\mathsf{Exp}^{\mathsf{num},1}_{\mathcal{A},k} = 0\right] \\
&= \sum_{(m_0,m_1)\in M_c} \Pr[X = m_0 - k] \cdot \Pr[X = m_1] \\
&\quad - \Pr[X = m_0] \cdot \Pr[X = m_1 - k]
\end{aligned}$$

Let $M_+ := \{(m_0, m_1) \in \mathbb{N}_0^2 : P(X = m_0 - k) \cdot P(X = m_1) - P(X = m_0) \cdot P(X = m_1 - k) > 0\}$. It further holds,

$$\delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t} \geq \sum_{(m_0,m_1)\,\in M_+} P(X = m_0 - k) \cdot \Pr[X = m_1]$$
$$- \Pr[X = m_0] \cdot \Pr[X = m_1 - k] = \sum_{m_1=0}^{k-1} \sum_{m_0=k}^{\infty} \Pr[X = m_0 - k] \cdot \Pr[X = m_1]$$
$$= \sum_{m_1=0}^{k-1} (1-p)^{m_1} p \sum_{m_0=0}^{\infty} (1-p)^{m_0} p$$
$$= 1 - (1-p)^k$$

It further follows, that an adversary who is instructed to always output $\beta = 0$ if for the output pair (m_0, m_1) if it holds that $\Pr[X = m_0 - k] \cdot \Pr[X = m_1] - \Pr[X = m_0] \cdot \Pr[X = m_1 - k] > 0$, guesses β correctly with an advantage of $1-(1-p)^k$. Hence, it holds for the adversarial advantage $\delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t} = 1-(1-p)^k$. It further holds, that $\max_{k' \leq k} \delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t} = \delta^{\mathsf{num}}_{k,\mathbb{P}_d,\mathbb{P}_t}$. Thus, the KTV-Helios scheme with $\mathbb{P}_d$ as a geometric distribution with parameter p achieves (δ, k)-participation privacy with $\delta = 1-(1-p)^k$.