Ballot secrecy with malicious bulletin boards

David Bernhard[1] and Ben Smyth[2]

[1]University of Bristol, UK
[2]Mathematical and Algorithmic Sciences Lab, France Research Center, Huawei Technologies Co. Ltd., France

April 13, 2015

Abstract

We propose a definition of ballot secrecy in the computational model of cryptography. The definition builds upon and strengthens earlier definitions by Bernhard *et al.* (ASIACRYPT'12, ESORICS'11 & ESORICS'13). The new definition is intended to ensure that ballot secrecy is preserved in the presence of malicious bulletin boards, whereas earlier definitions only consider trusted bulletin boards. It follows that the new definition prevents more attacks in comparison with earlier definitions.

1 Introduction

Ballot secrecy is a standard privacy requirement of voting systems.

- *Ballot secrecy.* A voter's vote is not revealed to anyone.

Many electronic voting systems — including systems that have been deployed in real-world, large-scale public elections — attempt to satisfy ballot secrecy by placing extensive trust in software and hardware. Unfortunately, many systems are not trustworthy and are vulnerable to attacks that could compromise ballot secrecy [GH07, Bow07, WWH$^+$10, WWIH12, SFD$^+$14]. Such vulnerabilities can be avoided by formulating ballot secrecy as a rigorous and precise security definition, and proving that systems satisfy the definition.

Bernhard *et al.* propose definitions of ballot secrecy [SB14, SB13a, BPW12a, BPW12b, BCP$^+$11a]. In their model, the participants are voters, an administrator, and a bulletin board. The definitions focus on detecting attacks by adversaries that control some voters. Attacks by adversaries that control the bulletin board are not detected, hence, the bulletin board is implicitly assumed to operate in accordance with the election scheme's rules. Unfortunately, this introduces a trust assumption and no privacy guarantees are provided if this trust assumption is violated.

Contribution. We examine definitions of ballot secrecy by Bernhard *et al.* and show that they do not prevent attacks by adversaries controlling the bulletin board. We propose a new definition of ballot secrecy that builds upon and strengthens these definitions and show that our definition prevents such attacks. In addition, we define a notion of extractability, which assert that election outcomes correspond to votes encapsulated inside ballots. Moreover, we show that extractability is implied by correctness.

2 Election schemes

We adopt the definition of election schemes from Smyth & Bernhard [SB14, SB13a], with one refinement: we define bulletin boards as sets, rather than multisets.

Definition 1 (Election scheme)**.** *An* election scheme *is a tuple of efficient algorithms* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{BB}, \mathsf{Tally})$ *such that:*

- Setup *takes a security parameter* 1^n *as input and outputs a bulletin board* $\mathfrak{bb}$*, vote space* $\mathfrak{m}$*, public key* pk*, and private key* sk*, where* $\mathfrak{bb}$ *is a set and* $\mathfrak{m}$ *is a set.*
- Vote *takes a public key* pk *and vote* $v \in \mathfrak{m}$ *as input, and outputs a ballot* b*.*
- BB *takes a bulletin board* $\mathfrak{bb}$ *and ballot* b *as input. It outputs* $\mathfrak{bb} \cup \{b\}$ *if successful (i.e.,* b *is added to* $\mathfrak{bb}$*) or* $\mathfrak{bb}$ *to denote failure (i.e.,* b *is not added). This algorithm must be deterministic*[1]*.*
- Tally *takes a private key* sk *and bulletin board* $\mathfrak{bb}$ *as input. It outputs a multiset* $\mathfrak{v}$ *representing the election outcome if successful or the empty multiset to denote failure. It also outputs auxiliary data* aux*.*

Moreover, the scheme must satisfy correctness, which we define in Section 2.1.

We refer the reader to Bernhard *et al.* for demonstrations of the definition's applicability. They propose a construction ($\mathsf{Enc2Vote}$) for election schemes from any non-malleable encryption scheme [SB14, SB13a, BPW12b, BCP+11a]. They also show that real voting systems, such as Helios, can be modelled as election schemes [BPW12b, BCP+11a].

Refinement: Bulletin boards as sets. Cortier & Smyth [CS13, CS11] demonstrate the following *malleability attacks* against election schemes that permit meaningfully related ballots on bulletin boards: an adversary observes a voter's ballot, casts a meaningfully related ballot, and exploits the relation to

[1] Bernhard *et al.* implicitly assume algorithm BB is deterministic and use this property in proofs, e.g., [BCP+11b, Appendix B], [BPW12a, Section 4], and [SB13b, Section 6]. Moreover, real schemes – such as Helios [Adi08] and Civitas [JCJ02] – define deterministic BB algorithms.

recover the voter's vote from the election outcome. For instance, in an election with voters Alice, Bob and Charlie, if Bob can cast a ballot that contains the same vote as Alice's ballot, then he can deduce Alice's vote by checking which candidate obtained at least two votes. A special case of malleability attacks are *replay attacks*, whereby an adversary casts an exact copy of a voter's ballot. We prevent replay attacks by assuming the bulletin board is a set. By comparison, Smyth & Bernhard [SB14, SB13a] assume the bulletin board is a multiset. It follows that our syntax for election schemes refines the definition by Smyth & Bernhard.

2.1 Correctness

Smyth & Bernhard [SB14, SB13a] formalise correctness.[2] Their definition is intended to ensure that a ballot can contribute a single vote to the tally and cannot influence the tally in any other way (e.g., by altering or removing votes). Furthermore, the contribution of a ballot for vote v is to add a vote for v to the tally. Unfortunately, the formalisation by Smyth & Bernhard implies that *every board tallies to the empty multiset*, which is clearly a mistake. We revise their correctness definition to eliminate this mistake.

Definition 2 (Correctness)**.** *A tuple of algorithms* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{BB}, \mathsf{Tally})$ *satisfy correctness, if for any* $(\mathfrak{bb}_0, \mathfrak{m}, pk, sk)$ *output by* $\mathsf{Setup}(1^n)$ *and any bulletin board* $\mathfrak{bb}$*, the following conditions are satisfied.*

1. *If computing* $\mathsf{Tally}_{sk}(\mathfrak{bb})$ *twice produces* $(\mathfrak{v}, aux)$ *and* $(\mathfrak{v}', aux')$*, then* $\mathfrak{v} = \mathfrak{v}'$*.*

Let algorithm τ *be defined as follows:* $\tau_{sk}(\mathfrak{bb})$ *computes* $(\mathfrak{v}, aux) \leftarrow \mathsf{Tally}_{sk}(\mathfrak{bb})$ *and outputs* $\mathfrak{v}$*. By Condition 1,* τ *is deterministic.*

2. *If* b *is output by* $\mathsf{Vote}_{pk}(v)$ *and* $b \notin \mathfrak{bb}$*, then* $\mathsf{BB}(\mathfrak{bb}, b) = \mathfrak{bb} \cup \{b\}$*.*
3. *If* $\mathfrak{bb} \neq \emptyset$ *and* $\tau_{sk}(\mathfrak{bb}) = \emptyset_M$ *(i.e.,* $\mathfrak{bb}$ *is invalid), then for all ballots* b *we have* $\tau_{sk}(\mathfrak{bb} \cup \{b\}) = \emptyset_M$ *too.*
4. *If* $\mathfrak{bb} = \emptyset$ *or* $\tau_{sk}(\mathfrak{bb}) \neq \emptyset_M$ *(i.e.,* $\mathfrak{bb}$ *is valid), then for any vote* $v \in \mathfrak{m}$ *and any ballot* b *output by* $\mathsf{Vote}_{pk}(v)$ *such that* $b \notin \mathfrak{bb}$*, we have* $\tau_{sk}(\mathfrak{bb} \cup \{b\}) = \tau_{sk}(\mathfrak{bb}) \cup_M \{\!|\, v \,|\!\}$*.*
5. *If* $\tau_{sk}(\mathfrak{bb}) \neq \emptyset_M$*, then* $|\tau_{sk}(\mathfrak{bb})| = |\mathfrak{bb}|$*.*

[2] Let $A(x_1, \ldots, x_n; r)$ denote the result of running probabilistic algorithm A on input $x_1, \ldots, x_n$ and coins r.

We write "for any x output by $A(x_1, \ldots, x_n)$" for the universal quantification over x such that x is a result of running probabilistic algorithm A on input $x_1, \ldots, x_n$, i.e., $x = A(x_1, \ldots, x_n; r)$ for some coins r.

We denote multisets as $\{\!|\, x_1, \ldots, x_n \,|\!\}$ and write $\emptyset_M$ for the empty multiset. The multiset union operator is denoted $\cup_M$ and the multiset intersection operator is denoted $\cap_M$. We write $|S|$ for the cardinality of multiset S.

Condition 1 asserts that the non-deterministic algorithm Tally always computes the same election outcome for a particular bulletin board. This allows us to speak of *the* result of tallying a particular board. Condition 2 asserts that ballots output by Vote are always accepted by algorithm BB, if they are not already present. Condition 3 asserts that if a non-emptyt board is invalid (i.e., produces the empty result), then adding more ballots to the board will never make it valid again. Condition 4 asserts that adding a ballot generated by Vote to a board increases the election outcome by exactly the vote in that ballot, except if the board is already invalid (in which case the previous condition says it stays invalid). Condition 5 asserts that on any valid board, the size of the result matches the number of ballots on the board. Note that this condition implies that the result of tallying an empty board is empty too.

Comparison with Smyth & Bernhard. The formulation of correctness by Smyth & Bernhard omitted the precondition $\mathfrak{bb} \neq \emptyset$ in Condition 3, which unfortunately implies that tallying always fails.

2.2 Honest-Ballot Extractability

Bernhard *et al.* [BCG+15] define *strong correctness*, which, among other things, asserts that there exists an extraction algorithm that inputs a private key and a ballot, and outputs a vote (or declares the ballot to be invalid). For ballots output by Vote, extraction returns the vote used to create the ballot. It follows that the extractor can be applied to bulletin boards to recover the election outcome. Moreover, each ballot contributes at most one vote to the election outcome. Our correctness property ensures a weaker result: ballots output by Vote contribute the vote used to create the ballot to the election outcome, and any remaining m ballots contribute at most m votes to the outcome (i.e., we do not ensure that each ballot contributes at most one vote).

Definition 3 (Honest-ballot extractability)**.** *An election scheme* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{BB}, \mathsf{Tally})$ *has honest-ballot extractability, if there exists a deterministic extraction algorithm* E*, which takes a private key and a ballot as input and outputs a vote, such that for any* $(\mathfrak{bb}_0, \mathfrak{m}, pk, sk)$ *output by* $\mathsf{Setup}(1^n)$*, the following condition holds.*

1. *For any* b *output by* $\mathsf{Vote}_{pk}(v)$*, we have* $E(sk, b) = v$*.*
2. *For any bulletin board* $\mathfrak{bb} = \mathfrak{bb}_1 \cup \mathfrak{bb}_2$ *with* $\mathfrak{bb}_1 \cap \mathfrak{bb}_2 = \emptyset$ *(i.e.,* $\mathfrak{bb}_1$ *and* $\mathfrak{bb}_2$ *are any partition of* $\mathfrak{bb}$*),* $\mathfrak{bb} \neq \emptyset$*,* $\tau_{sk}(\mathfrak{bb}) \neq \emptyset_M$ *(i.e.,* $\mathfrak{bb}$ *is valid), and all ballots in* $\mathfrak{bb}_1$ *are outputs of* Vote*, we have* $\tau_{sk}(\mathfrak{bb}_1) = \{\!|\, E(sk, b) \mid b \in \mathfrak{bb}_1 \,|\!\}$ *and* $\tau_{sk}(\mathfrak{bb}) = \tau_{sk}(\mathfrak{bb}_1) \cup_M \tau_{sk}(\mathfrak{bb}_2)$*.*

Proposition 1. *(Correct) Election schemes have honest-ballot extractability.*

The proof of Proposition 1 appears in Appendix A.

3 Ballot secrecy with a trusted board

Our informal definition of ballot secrecy (Section 1) could be formulated as an indistinguishability game similar to indistinguishability games for asymmetric encryption (e.g., IND-CPA and IND-CCA): we could challenge the adversary to determine whether a ballot is for one of two possible votes. This formalisation is too weak, because election schemes also output the election outcome and auxiliary data, which needs to be incorporated into the game. Unfortunately, it is insufficient to simply grant the adversary access to an oracle that provides an election outcome and auxiliary data corresponding to some ballots, because such a game is unsatisfiable, in particular, the adversary can use the oracle to reveal the vote encapsulated inside the challenge ballot. This reveals some limitations in our informal definition of ballot secrecy.

For simplicity, our informal definition of ballot secrecy deliberately omits some side-conditions, which are necessary for satisfiability, in particular, we did not stress that a voter's vote may be revealed in the following scenarios: unanimous election outcomes reveal how everyone voted and, more generally, election outcomes can be coupled with partial knowledge about the distribution of voters' votes to reveal voters' votes. For example, suppose Alice, Bob and Mallory vote in a referendum and the outcome is two "yes" votes and one "no" vote. Mallory can collude with Alice to reveal Bob's vote. Similarly, Mallory can collude with Bob to reveal Alice's vote. Moreover, Mallory can reveal that Alice and Bob both voted yes, if she voted no. Accordingly, ballot secrecy must concede that election outcomes reveal partial information about voters' votes[3], hence, we refine our informal definition of ballot secrecy as follows:

> A voter's vote is not revealed to anyone, except when the vote can be deduced from the election outcome and any partial knowledge on the distribution of votes.

This refinement ensures that the aforementioned examples are not violations of ballot secrecy. By comparison, if Mallory votes yes and can reveal the vote of either Alice or Bob without collusion, then she violates ballot secrecy.

Bernhard *et al.* use a bulletin board in their games and derive the election outcome and auxiliary data from the ballots on this board. The bulletin board is maintained in accordance with the election scheme's rules. The adversary can read the bulletin board, and can write ballots to the bulletin board on behalf of some voters, assuming such a write conforms to conditions defined by the scheme. In addition, the adversary has access to a left-right oracle [BDJR97, BR05] which can construct and write ballots to the bulletin board on the adversary's behalf. Ballots can be computed by the left-right oracle in two ways, corresponding to a randomly chosen bit β. If $\beta = 0$, then, given a pair of votes v_0, v_1, the oracle computes a ballot for v_0 and writes the ballot to

[3] We acknowledge that alternative formalisms of election schemes may permit different results. For instance, election schemes which only announce the winning candidate [BY86, HK02, HK04, DK05], rather than the breakdown of the votes for each candidate, could offer stronger notions of ballot secrecy.

the bulletin board. Otherwise ($\beta = 1$), the oracle writes a ballot for v_1 to the bulletin board. The left-right oracle essentially allows the adversary to control the distribution of votes cast by voters, but ballots cast by the oracle are always constructed using the prescribed Vote algorithm. This essentially corresponds to trusting the bulletin board.

At the end of an election, the adversary is given an election outcome and auxiliary data, and must determine whether $\beta = 0$ or $\beta = 1$. The computation of the election outcome and auxiliary data depends on whether the game is *consistent*: whether the inputs $(v_1, v'_1), \ldots, (v_n, v'_n)$ to the left-right oracle are equivalent, i.e., $\{\!|\, v_1, \ldots, v_n \,|\!\} = \{\!|\, v'_1, \ldots, v'_n \,|\!\}$. If the game is consistent, then the election outcome and auxiliary data are computed from the bulletin board. Otherwise (the game is inconsistent), the outcome is computed from the bulletin board that would have been produced if β had been 0, and no auxiliary data is returned.

The consistency condition prevents trivial distinctions. For example, suppose an adversary makes a single left-right oracle query with input $(0, 1)$, hence, the game is inconsistent. In this case, tallying the ballot resulting from the left-right oracle query would allow the adversary to trivially determine whether $\beta = 0$ or $\beta = 1$, yet this is not a privacy violation. Our consistency condition prevents the adversary from winning the game this way. By comparison, the consistency condition does not prevent distinctions due to the following two attacks that violate privacy.

1. Suppose the adversary inputs $(0, 1)$ and $(1, 0)$ to the left-right oracle, hence, the game is consistent. Further suppose that an adversary can recover the vote in the first ballot. This scheme cannot satisfy IND-SEC (defined below). (Cf. Benaloh's notion ballot secrecy [Ben96] which informally asserts that an adversary should not be able to detect if two voters swap their votes.)

2. Once again, suppose the adversary inputs $(0, 1)$ and $(1, 0)$ to the left-right oracle. Further suppose the adversary transforms the first ballot output by the left-right oracle into a new ballot for the same vote, without learning whether the first ballot is for 0 or 1. Moreover, suppose the adversary writes the new ballot to the bulletin board. The game is consistent: only the left-right oracle can affect consistency. The adversary can derive β from the tally by checking which candidate got two votes. This scheme cannot satisfy IND-SEC either. (Cf. malleability attacks *à la* Cortier & Smyth.)

It follows that the consistency condition does not prevent distinctions due to the above attacks.

3.1 Security definition

We recall[4] the security definition for ballot secrecy from Smyth & Bernhard [SB14].

Definition 4 (Ballot secrecy with a trusted board)**.** *Given an election scheme* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{BB}, \mathsf{Tally})$*, a security parameter* n *and an adversary* $\mathcal{A} = (A_1, A_2)$*, let* $\mathsf{IND\text{-}SEC}_{\mathcal{A},\Gamma}(n)$ *be the following quantity*[5]*:*

$$2 \cdot \Pr\left[\begin{array}{l} M_0 \leftarrow \emptyset_M; M_1 \leftarrow \emptyset_M; (\mathfrak{bb}_0, \mathfrak{m}, pk, sk) \leftarrow \mathsf{Setup}(1^n); \\ \mathfrak{bb}_1 \leftarrow \mathfrak{bb}_0; \beta \leftarrow_R \{0,1\}; s \leftarrow A_1^{\mathcal{O}}(\mathfrak{m}, pk); \\ \text{if } M_0 = M_1 \text{ then } \{(\mathfrak{v}, aux) \leftarrow \mathsf{Tally}_{sk}(\mathfrak{bb}_\beta)\} \\ \text{else } \{aux \leftarrow \perp; (\mathfrak{v}, aux') \leftarrow \mathsf{Tally}_{sk}(\mathfrak{bb}_0)\} \\ \quad : A_2(\mathfrak{v}, aux, s) = \beta \end{array}\right] - 1$$

Oracle $\mathcal{O}$ *is defined as follows:*

$\mathcal{O}()$**:** *output* $\mathfrak{bb}_\beta$*.*

$\mathcal{O}(b)$**:** $\mathfrak{bb}'_\beta \leftarrow \mathfrak{bb}_\beta$*;* $\mathfrak{bb}_\beta \leftarrow \mathsf{BB}(\mathfrak{bb}_\beta, b)$*;* if $\mathfrak{bb}_\beta \neq \mathfrak{bb}'_\beta$ then $\mathfrak{bb}_{1-\beta} \leftarrow \mathsf{BB}(\mathfrak{bb}_{1-\beta}, b)$*.*

$\mathcal{O}(v_0, v_1)$**:** $M_0 \leftarrow M_0 \cup_M \{\!|\, v_0 \,|\!\}$*;* $M_1 \leftarrow M_1 \cup_M \{\!|\, v_1 \,|\!\}$*;* $b_0 \leftarrow \mathsf{Vote}_{pk}(v_0)$*;* $b_1 \leftarrow \mathsf{Vote}_{pk}(v_1)$*;* $\mathfrak{bb}_0 \leftarrow \mathsf{BB}(\mathfrak{bb}_0, b_0)$*;* $\mathfrak{bb}_1 \leftarrow \mathsf{BB}(\mathfrak{bb}_1, b_1)$*. We assume* $v_0, v_1 \in \mathfrak{m}$*.*

We say Γ *satisfies* ballot secrecy with a trusted board *(*$\mathsf{IND\text{-}SEC}$*) if for all probabilistic polynomial time adversaries* $\mathcal{A}$ *we have* $\mathsf{IND\text{-}SEC}_{\mathcal{A},\Gamma}(n)$ *is negligible in* n*.*

The game captures a setting where an administrator generates a key pair using the scheme's Setup algorithm, publishes the public key, and only uses the private key to compute the election outcome at the end of an election[6]. Moreover, the administrator generates a bulletin board using algorithm Setup and uses algorithm BB to ensure that any writes to the bulletin board conform to conditions defined by the scheme, for instance, $\mathsf{BB}(\mathfrak{bb}, b)$ might only write to bulletin board $\mathfrak{bb}$ when ballot b is not meaningfully related to any other ballot on the bulletin board, thereby preventing the class of malleability attacks highlighted by Cortier & Smyth [CS13, CS11].

Adversarial read and write capabilities are captured by the oracle:

- Oracle $\mathcal{O}()$ allows the adversary to read the bulletin board.

[4] Our presentation revises notation to explicitly distinguish sets and multisets, Smyth & Bernhard do not. And we present the entire experiment as code, whereas Smyth & Bernhard mix code with descriptions in natural language.

[5] We write $A(x_1, \ldots, x_n)$ for $A(x_1, \ldots, x_n; r)$, where r is chosen uniformly at random. Assignment of α to x is written $x \leftarrow \alpha$. The assignment of a random element from set S to x is written $x \leftarrow_R S$.

[6] The administrator is assumed to be trusted, in particular, the administrator is assumed not to compute the election outcome for individual ballots. Generalising the definition to multiple administrators is a possible direction for future work.

- Oracle $\mathcal{O}(b)$ allows the adversary to write b to the bulletin board, assuming it conforms to conditions defined by the scheme, i.e., algorithm BB succeeds.

- Left-right oracle $\mathcal{O}(v_0, v_1)$ allows the adversary to write a ballot b to the bulletin board such that: in case $\beta = 0$ ballot b is for v_0 whereas in case $\beta = 1$ ballot b is for v_1.

In essence, the oracles allow the adversary to cast ballots on behalf of some voters and control the distribution of votes cast by the remaining voters.

The adversary is given the election outcome and auxiliary data, and challenged to determine the bit β. We stress that a unanimous election outcome will always reveal all voters' votes and we tolerate this factor in our game by challenging the adversary to determine the bit β, rather than the distribution of votes. Intuitively, if the adversary loses the game, then the adversary is unable to distinguish between the bulletin boards $\mathfrak{bb}_0$ and $\mathfrak{bb}_1$, hence, the adversary cannot distinguish between a ballot $b_0 \in \mathfrak{bb}_0$ and a ballot $b_1 \in \mathfrak{bb}_1$, therefore, voters' votes cannot be revealed. On the other hand, if the adversary wins the game, then there exists a strategy to distinguish ballots.

3.2 Limitations of trusted boards

Bernhard *et al.* assume the bulletin board is maintained in accordance with the election scheme's rules, in particular, ballots written to the bulletin board must conform to conditions defined by the scheme. This can be assured by insisting that all ballots written to the bulletin board are written using algorithm BB. The security game (Definition 4) enforces conformance by restricting the adversary's write capabilities to oracle calls which only write to the bulletin board using algorithm BB. It follows that ballot secrecy with a trusted board only offers privacy guarantees when the adversary's write capability is restricted in this manner. Unfortunately, an unnecessary trust assumption is introduced: voters must trust the system to only add ballots to the bulletin board using algorithm BB. If this trust assumption is violated, then an election scheme satisfying ballot secrecy with a trusted board may fail to provide privacy. We give an example of this using a variant of Bernhard *et al.*'s $\mathsf{Enc2Vote}$ construction [SB14, SB13a, BPW12b, BCP+11a].

Definition 5 ($\mathsf{Backdoor\text{-}Enc2Vote}$)**.** *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, the election scheme* $\mathsf{Backdoor\text{-}Enc2Vote}(\Pi)$ *is defined as follows.*

- Setup *takes a security parameter* 1^n *as input and outputs* $(\emptyset, \mathfrak{m}, pk, sk)$*, where* $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ *and* $\mathfrak{m}$ *is the encryption scheme's message space.*

- Vote *takes a public key* pk *and vote* $v \in \mathfrak{m}$ *as input, computes* $b \leftarrow \mathsf{Enc}_{pk}(v)$*, and outputs* b*.*

- BB *takes a bulletin board* $\mathfrak{bb}$ *and ballot* b *as input. If* $b \in \mathfrak{bb} \cup \{\epsilon\}$*, then the algorithm outputs* $\mathfrak{bb}$ *(denoting failure), otherwise, the algorithm outputs* $\mathfrak{bb} \cup \{b\}$*.*
- Tally *takes as input a private key* sk *and a bulletin board* $\mathfrak{bb}$*. If* $\epsilon \in \mathfrak{bb}$*, then* $aux \leftarrow \{(b, \mathsf{Dec}_{sk}(b)) \mid b \in \mathfrak{bb}\}$*, otherwise,* $aux \leftarrow \perp$*. It outputs the multiset* $\{\!| \mathsf{Dec}_{sk}(b) \mid b \in \mathfrak{bb} |\!\}$ *and auxiliary data* aux*.*

Informally, given an asymmetric encryption scheme Π satisfying NM-CPA, the encryption scheme enables election scheme Backdoor-Enc2Vote(Π) to ensure ballot secrecy until tallying. Moreover, if the bulletin board does not contain ϵ, then algorithm Tally maintains ballot secrecy by returning the number of votes for each candidate as a multiset of votes. Since algorithm BB prevents ϵ from appearing on the bulletin board, election scheme Backdoor-Enc2Vote(Π) preserves ballot secrecy with a trusted board.

Proposition 2. *Given an encryption scheme* Π *satisfying NM-CPA, the election scheme* Backdoor-Enc2Vote(Π) *satisfies ballot secrecy with a trusted board.*

A proof that Backdoor-Enc2Vote(Π) satisfies ballot secrecy with a trusted board can be constructed similarly to the proof of [BPW12b, Theorem 4.2]. Nonetheless, privacy can be violated if the bulletin board contains ϵ, since this causes algorithm Tally to output auxiliary data which maps ballots to votes. This may occur in practice if the bulletin board is not trustworthy. We overcome this limitation in a new definition of ballot secrecy.

4 Ballot secrecy with malicious boards

The definition of ballot secrecy by Bernhard *et al.* assumes the bulletin board is trusted. We remove this trust assumption by assuming that the adversary controls the bulletin board, i.e., we remove restrictions on the adversary's write capabilities. This essentially corresponds to the bulletin board being malicious. We additionally reformulate the left-right oracle to output ballots to the adversary, rather than writing them to the bulletin board.

The adversary is once again supplied with the election outcome and auxiliary data, and challenged to guess the randomly chosen bit β which controls the left-right oracle's behaviour. The computation of the election outcome and auxiliary data uses a refined notion of consistency that is satisfied if: inputs to the left-right oracle are equivalent when the corresponding left-right oracle's outputs appear on the bulletin board constructed by the adversary. For example, suppose the inputs to the left-right oracle are $(v_{1,0}, v_{1,1}), \ldots, (v_{n,0}, v_{n,1})$ and the corresponding outputs are $b_1, \ldots, b_n$, further suppose that the bulletin board $\mathfrak{bb} = \{b_1, \ldots, b_\ell\}$ and $\ell \leq n$, the game is consistent if $\{\!| v_{1,0}, \ldots, v_{\ell,0} |\!\} = \{\!| v_{1,1}, \ldots, v_{\ell,1} |\!\}$.

4.1 Security definition

We formulate a new definition of ballot secrecy based upon our informal discussion above.

Definition 6 (Ballot secrecy)**.** *Given an election scheme* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{BB}, \mathsf{Tally})$*, a security parameter* n *and a two-stage adversary* $\mathcal{A} = (A_1, A_2)$*, let* $\mathit{IND\text{-}SEC}^{\#}_{\mathcal{A},\Gamma}(n)$ *be the following quantity:*

$$2 \cdot Pr\left[\begin{array}{l}(\mathfrak{bb}, \mathfrak{m}, pk, sk) \leftarrow \mathsf{Setup}(1^n); \beta \leftarrow_R \{0,1\}; S \leftarrow \emptyset; \\ (\mathfrak{bb}', s) \leftarrow A_1^{\mathcal{O}}(\mathfrak{bb}, \mathfrak{m}, pk); (\mathfrak{v}, aux) \leftarrow \mathsf{Tally}_{sk}(\mathfrak{bb}') \\ \quad : \ A_2(\mathfrak{v}, aux, s) = \beta \ \wedge \forall v \in \mathfrak{m} \ . \\ \qquad |\{b \mid b \in \mathfrak{bb}' \wedge \exists v_1 \ . \ (b, v, v_1) \in S\}| = \\ \qquad |\{b \mid b \in \mathfrak{bb}' \wedge \exists v_0 \ . \ (b, v_0, v) \in S\}|\end{array}\right] - 1$$

Oracle $\mathcal{O}$ *is defined as follows:*

- $\mathcal{O}(v_0, v_1)$ *computes* $b \leftarrow \mathsf{Vote}_{pk}(v_\beta); S \leftarrow S \cup \{(b, v_0, v_1)\}$ *and outputs* b*, where* $v_0, v_1 \in \mathfrak{m}$*.*

We say Γ *satisfies* ballot secrecy *(*$\mathit{IND\text{-}SEC}^{\#}$*) if for all probabilistic polynomial time adversaries* $\mathcal{A}$ *we have* $\mathit{IND\text{-}SEC}^{\#}_{\mathcal{A},\Gamma}(n)$ *is negligible in* n*.*

Informally, an adversary who cannot win this game, cannot distinguish a ballot for vote v_0 from a ballot for vote v_1. Therefore, such an adversary cannot discover voters' votes from looking at their ballots.

4.2 Overcoming limitations of trusted boards

Ballot secrecy ($\mathsf{IND\text{-}SEC}^{\#}$) is strictly stronger than ballot secrecy with a trusted bulletin board ($\mathsf{IND\text{-}SEC}$). We prove this result as follows. First, we show that any election scheme satisfying $\mathsf{IND\text{-}SEC}^{\#}$ also satisfies $\mathsf{IND\text{-}SEC}$ (Theorem 1). Secondly, we have seen that $\mathsf{Backdoor\text{-}Enc2Vote}$ can be used to construct an election scheme $\mathsf{Backdoor\text{-}Enc2Vote}(\Pi)$ satisfying $\mathsf{IND\text{-}SEC}$ (Proposition 2) and we show that $\mathsf{Backdoor\text{-}Enc2Vote}(\Pi)$ does not satisfy $\mathsf{IND\text{-}SEC}^{\#}$ (Proposition 3). It follows that $\mathsf{IND\text{-}SEC}^{\#}$ is strictly stronger than $\mathsf{IND\text{-}SEC}$.

Theorem 1 ($\mathsf{IND\text{-}SEC}^{\#}$ is stronger than $\mathsf{IND\text{-}SEC}$)**.** *If an election scheme satisfies ballot secrecy, then the election scheme satisfies ballot secrecy with a trusted board.*

The proof of Theorem 1 appears in Appendix B.

Proposition 3. *Given an encryption scheme* Π *satisfying NM-CPA, the election scheme* $\mathsf{Backdoor\text{-}Enc2Vote}(\Pi)$ *does not satisfy ballot secrecy.*

A proof that $\mathsf{Backdoor\text{-}Enc2Vote}(\Pi)$ does not satisfy ballot secrecy can be constructed by formalising an adversary that adds ϵ to the bulletin board.

Our definition of ballot secrecy improves upon existing definitions by Bernhard *et al.* by detecting attacks that arise when the bulletin board is controlled by the adversary, in particular, we can detect attacks against our $\mathsf{Backdoor\text{-}Enc2Vote}$ construction.

4.3 Implementation notes

Definitions of ballot secrecy by Bernhard *et al.* have used three different data structures to model bulletin boards:

- *List* [BCP+11a, BPW12a, BPW12b]: bulletin board entries are ordered and may contain duplicates.
- *Multiset* [SB13a, SB14]: bulletin board entries are unordered and may contain duplicates.
- *Set* (this work): bulletin board entries are unordered and do not contain duplicates.

As discussed in Section 2, the shift to data structures which do not contain duplicates prevents the class of replay attacks identified by Cortier & Smyth [CS13, CS11] (variants of their attack that exploit malleable ballots are not eradicated). Hence, the data structure helps ensure ballot secrecy. It follows that implementors should ensure that the bulletin board is a set. Alternatively, the bulletin board should be converted to a set before input to algorithm Tally.

5 Conclusion

This paper shows that malicious bulletin boards can violate privacy in a manner that cannot be detected by Bernhard *et al.*'s definition of ballot secrecy. We have proposed a new definition of ballot secrecy to overcome this problem. Our definition builds upon the games by Bernhard *et al.* as follows. First, we refine their syntax for election schemes: we model the bulletin board as a set, rather than a multiset. Secondly, we remove restrictions on writing to the bulletin board: we assume the bulletin board is controlled by the adversary, rather than the administrator. Thirdly, we reformulate the left-right oracle: the oracle outputs ballots to the adversary, rather than writing them to the bulletin board. The resulting definition strengthens definitions by Bernhard *et al.* to ensure that ballot secrecy is preserved in the presence of malicious bulletin boards.

Acknowledgements. We are particularly grateful to Elizabeth Quaglia and Susan Thomson for discussion that helped simplify our new definition of ballot secrecy. We are also grateful to the anonymous reviewers for constructive criticism. This work has been partly supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC project *CRYSP* (259639) and by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO. This work was performed in part at INRIA.

A Proof of Proposition 1

We define the extractor $E(sk, b)$ to run $(\mathfrak{v}, aux) \leftarrow \mathsf{Tally}_{sk}(\{b\})$; if $\mathfrak{v}$ is a multiset $\{\!|\, v \,|\!\}$ of cardinality 1, then we let E return v, otherwise, it returns $\perp$. Condition

1 of correctness guarantees that this is well-defined: Tally always returns the same election result for the same board. Correctness condition 4 shows that the extractor works as desired for correctly generated ballots (i.e. generated using Vote).

For a non-empty and valid $\mathfrak{bb}$, take any partition into $\mathfrak{bb}_1$ and $\mathfrak{bb}_2$ such that all ballots in $\mathfrak{bb}_1$ are hoenstly generated (i.e. such b was produced by $\mathsf{Vote}_{pk}(v)$ for some $v \in \mathfrak{m}$). Let $\mathfrak{v}_2$ be the result of tallying $\mathfrak{bb}_2$. Correctness' condition 1 guarantees that multiple runs of Tally return the same result on any board, so "the" result $\mathfrak{v}_2$ is well-defined. We add the ballots of $\mathfrak{bb}_1$ to $\mathfrak{bb}_2$ one by one. Condition 4 of correctness says that this will add exactly the vote v from which each of these ballots was created to the result each time, since all ballots in $\mathfrak{bb}_1$ are outputs of Vote. We have established above that this is exactly the same vote as the extractor E returns on such ballots.

We have shown $\mathfrak{v} = \mathfrak{v}_2 \cup_M \{\!| E(sk, b) \mid b \in \mathfrak{bb}_1 |\!\}$. So we define $\mathfrak{v}_1 = \mathfrak{v} \setminus_M \mathfrak{v}_2$; since all ballots in $\mathfrak{bb}_1$ are are outputs of Vote, it follows that $\mathfrak{v}_1$ is also the result of tallying $\mathfrak{bb}_1$. □

B Proof of Theorem 1

In brief, the proof is a reduction from $\mathsf{IND\text{-}SEC}$ to $\mathsf{IND\text{-}SEC}^{\#}$. If an adversary creates a consistent game, the reduction is trivial. If an adversary creates an inconsistent game however then we need to be more careful: an inconsistent $\mathsf{IND\text{-}SEC}$ will just return the left result with no auxiliary data but an inconsistent $\mathsf{IND\text{-}SEC}^{\#}$ will not let the adversary win. If the game is inconsistent when the tally should be computed, the reduction passes only the dishonest ballots (from $\mathcal{O}(b)$ queries) to the $\mathsf{IND\text{-}SEC}^{\#}$ challenger, restoring consistency. The reduction then adds the "left" honest votes from $\mathcal{O}(v_0, v_1)$ queries back into the returned result itself.

Suppose $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{BB}, \mathsf{Tally})$ is an election scheme that does not satisfy ballot secrecy with a trusted board. By Definition 4, there exists a probabilistic polynomial-time adversary $\mathcal{A} = (A_1, A_2)$ such that for every negligible function negl, we have $\mathsf{IND\text{-}SEC}_{\mathcal{A},\Gamma}(n) > \mathsf{negl}(n)$ for infinitely many n. An adversary $\mathcal{B} = (B_1, B_2)$ against $\mathsf{IND\text{-}SEC}^{\#}$ is constructed below. Let $\mathcal{O}_{\mathcal{A}}$ denote $\mathcal{A}$'s oracle and $\mathcal{O}_{\mathcal{B}}$ denote $\mathcal{B}$'s oracle.

Algorithm B_1. On input $\mathfrak{bb}$, $\mathfrak{m}$ and pk, the algorithm proceeds as follows. Initialise set $L \leftarrow \emptyset$ and compute $s \leftarrow A_1^{\mathcal{O}_{\mathcal{A}}}(\mathfrak{m}, pk)$, handling any oracle calls from A_1 as follows:

- $\mathcal{O}_{\mathcal{A}}(v_0, v_1)$: compute $b \leftarrow \mathcal{O}_{\mathcal{B}}(v_0, v_1); L \leftarrow L \cup \{(b, v_0, v_1)\}; \mathfrak{bb} \leftarrow \mathsf{BB}(\mathfrak{bb}, b)$.
- $\mathcal{O}_{\mathcal{A}}(b)$: compute $\mathfrak{bb} \leftarrow \mathsf{BB}(\mathfrak{bb}, b)$.
- $\mathcal{O}_{\mathcal{A}}()$: output $\mathfrak{bb}$.

Let L_0 be the multiset in which each vote v appears with multiplicity $|\{b \in \mathfrak{bb} \mid \exists v'.(b, v, v') \in L\}|$ and similarly let L_1 be the multiset in which

each v appears with multiplicity $|\{b \in \mathfrak{bb} \mid \exists v'.(b, v', v) \in L\}|$. These multisets have the same role as the ones used to evaluate the consistency condition in IND-SEC$^{\#}$.

If $L_0 = L_1$, then output $(\mathfrak{bb}, (s, L_0, L_1))$. Otherwise, compute $\mathfrak{bb}' \leftarrow \mathfrak{bb} \setminus \{b \mid b \in \mathfrak{bb} \wedge \exists v_0, v_1.(b, v_0, v_1) \in L\}$ and output $(\mathfrak{bb}', (s, L_0, L_1))$.

We show by induction that the embedded adversary A_1 sees the same distibution of all elements as in the IND-SEC game.

When A_1 makes an $\mathcal{O}()$ call, the board $\mathfrak{bb}$ is returned, so we have to show that this is consistent with what A_1 expects. At the start of the game, $\mathfrak{bb}$ is empty, which is what A_1 would see at the start of the IND-SEC game if it asked for the board before adding any ballots. In an $\mathcal{O}(b)$ query, b is appended to the board if and only if it passes $\mathsf{BB}(\mathfrak{bb}, b)$ validation, which is the same as in the IND-SEC game since BB is a pure function[7]. In an $\mathcal{O}(v_0, v_1)$ query, a ballot b is added to $\mathfrak{bb}$ (again with validation), and this ballot comes from the IND-SEC$^{\#}$ oracle which produces ballots identical to the IND-SEC two-parameter oracle. So the board $\mathfrak{bb}$ is kept consistent for all calls.

Algorithm B_2. Given input $\mathfrak{v}$, aux and (s, L_0, L_1), the algorithm computes g as follows:

$$g \leftarrow \begin{cases} A_2(\mathfrak{v}, aux, s) & \text{if } L_0 = L_1 \\ A_2(\emptyset_M, \perp, s) & \text{else if } \mathfrak{v} = \emptyset_M, \text{ denoting failure} \\ A_2(\mathfrak{v} \cup_M L_0, \perp, s) & \text{otherwise} \end{cases}$$

Output g.

It is sufficient to show that the adversary $\mathcal{B}$ chooses g correctly with the same advantage as $\mathcal{A}$ in the following two cases. Case I: $L_0 = L_1$. By definition of B_1, the bulletin board $\mathfrak{bb}$ contains exactly the ballots added by $\mathcal{O}_{\mathcal{A}}(\cdot)$ and $\mathcal{O}_{\mathcal{A}}(\cdot, \cdot)$ queries. Further, the game is consistent (from the challenger's point of view). It follows that the embedded adversary A_2 sees the same distibution of all elements as in IND-SEC, hence, adversary $\mathcal{B}$ chooses g correctly with the same advantage as $\mathcal{A}$.

Case II: $L_0 \neq L_1$. By definition of B_1, the bulletin board $\mathfrak{bb}'$ returned by B_1 contains exactly the ballots added by $\mathcal{O}_{\mathcal{A}}(\cdot)$ queries. Since $\mathfrak{bb}'$ does not contain any ballots added by $\mathcal{O}_{\mathcal{A}}(\cdot, \cdot)$ queries, no ballots in $\mathfrak{bb}'$ appear in elements of L. The key point here is that by passing only $\mathfrak{bb}'$ back to the challenger, the game is consistent again from the challenger's point of view.

We partition the board $\mathfrak{bb}$ into $\mathfrak{bb}_1$ consisting of all ballots from $\mathcal{O}(v_0, v_1)$ queries and $\mathfrak{bb}_2$ consisting of the ballots from $\mathcal{O}(b)$ queries. By construction, all ballots in $\mathfrak{bb}_1$ are outputs of Vote and $\mathfrak{bb}_2 = \mathfrak{bb}'$.

In the IND-SEC game, we have $\tau(\mathfrak{bb}) = \tau(\mathfrak{bb}_1) \cup_M \tau(\mathfrak{bb}_2)$ by honest-ballot extractability. A quick observation shows that L_0 in the reduction is identical

[7]This is why we are explicit about BB being pure. The IND-SEC game runs BB twice on $\mathcal{O}(b)$ ballots (once on each board) and our reduction runs BB a third time, which could cause problems if BB were stateful or randomised. Earlier proofs seem to take this for granted.

to $M_0 = \tau(\mathfrak{bb}_1)$ in the IND-SEC game for any execution: both these multisets collect v_0 from each $\mathcal{O}(v_0, v_1)$ query. The result $L_0 \cup_M \tau(\mathfrak{bb}')$ that the reduction computes is threefore the same value as the adversary would see in the IND-SEC game, showing that the distribution of the tallies is the same in both cases (the auxiliary data is always $\perp$ in the inconsistent case). □

References

[Adi08] Ben Adida. Helios: Web-based Open-Audit Voting. In *USENIX Security'08: 17th USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[BCG+15] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. (SoK) A comprehensive analysis of game-based ballot privacy definitions. In *S&P'15: 36th Security and Privacy Symposium*. IEEE Computer Society, 2015.

[BCP+11a] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot privacy. In *ESORICS'11: 16th European Symposium on Research in Computer Security*, volume 6879 of *LNCS*, pages 335–354. Springer, 2011.

[BCP+11b] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot privacy. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.388.1624`, 2011.

[BDJR97] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *FOCS'97: 38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society, 1997.

[Ben96] Josh Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Department of Computer Science, Yale University, 1996.

[Bow07] Debra Bowen. Secretary of State Debra Bowen Moves to Strengthen Voter Confidence in Election Security Following Top-to-Bottom Review of Voting Systems. California Secretary of State, press release DB07:042 `http://www.sos.ca.gov/elections/voting_systems/ttbr/db07_042_ttbr_system_decisions_release.pdf`, August 2007.

[BPW12a] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *ASIACRYPT'12: 18th International Conference on the Theory and Application of Cryptology and Information Security*, volume 7658 of *LNCS*, pages 626–643. Springer, 2012.

[BPW12b] David Bernhard, Olivier Pereira, and Bogdan Warinschi. On Necessary and Sufficient Conditions for Private Ballot Submission. Cryptology ePrint Archive, Report 2012/236 (version 20120430:154117b), 2012.

[BR05] Mihir Bellare and Phillip Rogaway. Symmetric Encryption. In *Introduction to Modern Cryptography*, chapter 4. 2005. `http://cseweb.ucsd.edu/~mihir/cse207/classnotes.html`.

[BY86] Josh Benaloh and Moti Yung. Distributing the Power of a Government to Enhance the Privacy of Voters. In *PODC'86: 5th Principles of Distributed Computing Symposium*, pages 52–62. ACM Press, 1986.

[CS11] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. In *CSF'11: 24th Computer Security Foundations Symposium*, pages 297–311. IEEE Computer Society, 2011.

[CS13] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1):89–148, 2013.

[DK05] Yvo Desmedt and Kaoru Kurosawa. Electronic Voting: Starting Over? In *ISC'05: International Conference on Information Security*, volume 3650 of *LNCS*, pages 329–343. Springer, 2005.

[GH07] Rop Gonggrijp and Willem-Jan Hengeveld. Studying the Nedap/Groenendaal ES3B Voting Computer: A Computer Security Perspective. In *EVT'07: Electronic Voting Technology Workshop*, 2007.

[HK02] Alejandro Hevia and Marcos A. Kiwi. Electronic Jury Voting Protocols. In *LATIN'02: Theoretical Informatics*, volume 2286 of *LNCS*, pages 415–429. Springer, 2002.

[HK04] Alejandro Hevia and Marcos A. Kiwi. Electronic jury voting protocols. *Theoretical Computer Science*, 321(1):73–94, 2004.

[JCJ02] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165, 2002.

[SB13a] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. In *ESORICS'13: 18th European Symposium on Research in Computer Security*, volume 8134 of *LNCS*, pages 463–480. Springer, 2013.

[SB13b] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. Cryptology ePrint Archive, Report 2013/235, 2013.

[SB14] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence: definitions and relations. Cryptology ePrint Archive, Report 2013/235 (version 20141010:082554), 2014.

[SFD+14] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. Security Analysis of the Estonian Internet Voting System. In *CCS'14: 21st ACM Conference on Computer and Communications Security*, pages 703–715. ACM Press, 2014.

[WWH+10] Scott Wolchok, Eric Wustrow, J. Alex Halderman, Hari K. Prasad, Arun Kankipati, Sai Krishna Sakhamuri, Vasavya Yagati, and Rop Gonggrijp. Security Analysis of India's Electronic Voting Machines. In *CCS'10: 17th ACM Conference on Computer and Communications Security*, pages 1–14. ACM Press, 2010.

[WWIH12] Scott Wolchok, Eric Wustrow, Dawn Isabel, and J. Alex Halderman. Attacking the Washington, D.C. Internet Voting System. In *FC'12: 16th International Conference on Financial Cryptography and Data Security*, volume 7397 of *LNCS*, pages 114–128. Springer, 2012.