Protecting the Privacy of Voters: New Definitions of Ballot Secrecy for E-Voting

Ashley Fraser and Elizabeth A. Quaglia

Information Security Group, Royal Holloway, University of London, UK
{Ashley.Fraser.2016, Elizabeth.Quaglia}@rhul.ac.uk

Abstract. Protecting the privacy of voters is a basic requirement of any electronic voting scheme, and formal definitions can be used to prove that a scheme satisfies privacy. In this work, we provide new game-based definitions of ballot secrecy for electronic voting schemes. First, we propose an intuitive definition in the honest model, i.e., a model in which all election officials are honest. Then, we show that this definition can be easily extended to the malicious ballot box setting and a setting that allows for a distributed tallier. In fact, to the best of our knowledge, we provide the first game-based definition of ballot secrecy that models both a malicious ballot box *and* a malicious subset of talliers. We demonstrate that our definitions of ballot secrecy are satisfiable, defining electronic voting scheme constructions which we prove satisfy our definitions. Finally, we revisit existing definitions, exploring their limitations and contextualising our contributions to the field.

Keywords: E-voting, ballot secrecy, game-based definitions

1 Introduction

Voter privacy ensures that a voter can participate in democratic processes without risk of intimidation or blackmail, and is regarded as a basic requirement in many elections.[1] At a fundamental level, voter privacy is defined as *ballot secrecy*, which intuitively states that a voter's vote remains secret throughout the election, except when the result of the election reveals the vote, for example, in the event of a unanimous result. Stronger notions of privacy exist [23], including receipt-freeness and coercion-resistance but, in this work, we restrict discussion to the basic requirement of ballot secrecy.

Rigorous definitions of privacy have been proposed in the literature, the most common of which are game-based definitions. Early game-based definitions [4, 5] follow the well-established route of indistinguishability experiments (such as, for example, IND-CPA for public key encryption [27]). That is, an adversary must

[1] Though the focus of this paper is privacy, another basic property of e-voting schemes is *verifiability*, by which any interested party can check that the result of the election is computed correctly. For a full discussion of this notion, including formal definitions, the interested reader can consult [18].

distinguish two different election views, when provided with a result computed with respect to the viewed election. Informally, we refer to this approach as the Benaloh approach, recognising the fact that Benaloh established this approach in early works [4, 5].[2] The Benaloh approach is utilised in a number of ballot secrecy definitions [10, 11, 19, 30]. However, to address the fact that the Benaloh approach limits the class of voting result functions that can be considered (see §5), a separate line of research departed from this approach, focusing instead on definitions that provide an adversary with a view of a 'real' election or a 'fake' election [7–9, 16, 20]. Here, the adversary is always provided with a tally computed with respect to the 'real' election. In particular, this approach is favoured by BPRIV [7], a highly-regarded definition of ballot secrecy. More generally, the majority of game-based definitions of ballot secrecy position themselves in the so-called *honest* model. That is, they consider all election officials to be trusted. With respect to formal definitions, the consideration of the malicious setting is a young area of research and has focused on a malicious ballot box [11, 19, 20].

1.1 Our Contributions

New definitions of ballot secrecy. In this work, we revisit the approach taken in [5] and present new definitions of ballot secrecy. We choose this approach due to the well-established, intuitive nature of indistinguishability experiments. Moreover, though we recognise that the Benaloh approach limits the class of result functions considered, an issue that we explore in §3 and §5, the Benaloh approach, and our definitions specifically, provides a number of advantages over existing definitions, which we discuss below.

First, we define BS, a definition of ballot secrecy in the honest model (§3). BS builds upon [5], capturing several additional functionalities. First of all, BS models e-voting schemes with a registration phase. That is, eligible voters are provided with a credential that is required to cast a ballot. Voter registration is not modelled in [5] or subsequent, related, definitions [10, 11] and, hence, BS is the first definition that follows this approach to model voter registration. This reflects how advanced e-voting schemes are modelled, for example, Belenios [2, 17] and Civitas [14, 15]. Secondly, BS allows for *adaptive* corruption of voters. Previous game-based definitions that model registration of voters are limited to static corruption of voters only [16, 19] or allow for voters to cast only a single ballot [30]. Therefore, BS improves upon existing definitions in the honest model by modelling registration of voters and adaptive corruption of voters.

Our second definition, mbbBS, extends BS to the malicious ballot box setting (§4.1). mbbBS is similar to the definition presented in [11], which also adopts the approach of [5]. However, contrary to [11], and similarly to BS, mbbBS models registration of voters. Our model for mbbBS captures static corruption of voters only, a consequence of using the Benaloh approach, as we will explain in §4.1. We note, however, that *all* ballot secrecy definitions that model voter registration in the malicious ballot box setting only capture static voter corruption [19, 20].

[2] In particular, Benaloh is the sole author of [4] and is a co-author of [5].

Finally, we define dtBS, which, to the best of our knowledge, is the first ballot secrecy definition that models a malicious ballot box and a distributed tallier, in which the adversary corrupts a subset of talliers (§4.2). dtBS extends mbbBS to model an adversary that corrupts a subset of talliers. Like mbbBS, dtBS considers static corruption of voters. Given that the malicious setting has received significantly less attention than the honest model, and a malicious tallier has not been explored, we believe that definitions in the malicious model, such as dtBS, are valuable and desirable. In particular, Helios [28] distributes the role of the tallier, yet proofs of ballot secrecy for Helios model the tallier as a single entity [7, 17]. As such, we see dtBS as an important step towards formal proofs of security under realistic trust assumptions.

Feasibility of our definitions. For all of our definitions, we show feasibility, proving that our definitions can be satisfied. Specifically, we define an e-voting scheme Γ_{mini}, and prove it satisfies BS and mbbBS. We then extend Γ_{mini} to a setting with a distributed tallier, in a construction that we call Γ'_{mini}, and prove that it satisfies dtBS. Our scheme Γ_{mini} is a simple construction; specifically, it is not verifiable. However, Γ_{mini} can be used to prove the security of real, verifiable, e-voting schemes, similarly to how minivoting was used to prove ballot secrecy of Helios in [8].[3] Indeed, using the technique of [8], Γ_{mini} can also be extended to a generic, Helios-like, e-voting protocol which can be shown to satisfy our notions of ballot secrecy if Γ_{mini} satisfies ballot secrecy (and the verification process is secure). As such, though we simply demonstrate feasibility, our definitions can be applied to practical e-voting schemes.

Contextualising our definitions. Finally, we compare ballot secrecy definitions, with the goal of better understanding the limitations of definitions. In particular, we discuss related work and place our definitions in the context of the existing literature in §5. It emerges that our definitions provide an advantage with respect to extendibility, when compared to BPRIV. Moreover, our definitions improve upon existing definitions that follow the Benaloh approach, particularly with respect to the attacker model.

2 Preliminaries

Notation. We write $x \leftarrow X$ to denote assignment of X to x. We use standard set notation and write $\{\{x_1, \ldots, x_n\}\}$ to denote a multiset consisting of elements

[3] In fact, in [8], Bernhard *et al.* introduce minivoting, a simple e-voting scheme in which voters simply encrypt their vote and a tallier decrypts each ciphertext and computes the result from the resulting plaintext votes. Like Γ_{mini}, mini-voting is not verifiable. However, Bernhard *et al.* prove that mini-voting satisfies a notion of ballot secrecy and subsequently build a generic, verifiable, e-voting scheme with homomorphic tallying that also satisfies ballot secrecy. Helios is an instantiation of this generic e-voting scheme and, therefore, Helios can be shown to satisfy ballot secrecy if the underlying mini-voting construction is ballot secret.

$x_1, \ldots, x_n$, writing $\{\{\}\}$ to denote the empty multiset. Additionally, we write $L = (x_1, \ldots, x_n)$ to denote a list L with entries $x_1, \ldots, x_n$, and write $L \leftarrow L||y$ to denote appending y to the list L. We let $A(x_1, \ldots, x_n; c)$ denote the output of algorithm A on inputs $x_1, \ldots, x_n$ and coins c. Generally, we omit coins and simply write $A(x_1, \ldots, x_n)$. Moreover, we write $A^{\mathcal{O}}$ to denote algorithm A with oracle access to $\mathcal{O}$.

We provide syntax for a *single-pass* e-voting scheme, which requires a voter to post a single message in order to cast a vote. We consider that an e-voting scheme is defined relative to a result function[4] $f : (\mathcal{V} \times \mathcal{C})^* \to R$, where $\mathcal{V} = \{id_1, \ldots, id_{n_v}\}$ is the set of n_v eligible voters, $\mathcal{C} = \{1, \ldots, n_c\}$ is the set of n_c candidates, and R is the result space of the election. Informally, our syntax captures an e-voting scheme with the following structure. Firstly, an election administrator publishes public parameters of the scheme. Then, a registrar provides eligible voters with a public and private credential and adds public credentials to a set $\mathcal{L}$. Voters cast ballots that are linked to their public credential, and a ballot box manager processes and posts ballots to a ballot box $\mathcal{BB}$. We assume that the ballot box is private and the ballot box manager publishes a public view of the ballot box, known as the bulletin board, $\mathcal{PBB}$. Finally, the tallier computes and publishes the result of the election with a proof of correct tallying that can be verified by anyone. We formally define the syntax of an e-voting scheme, adapted from the syntax of [7, 12], in Definition 1.

Definition 1 (E-voting scheme). *An* e-voting scheme *Γ for a result function f is a tuple of probabilistic polynomial-time (PPT) algorithms* (Setup, Register, Vote, Valid, Append, Publish, Tally, Verify) *such that:*

Setup(1^λ) On input security parameter 1^λ, algorithm Setup initialises set $\mathcal{L}$ and ballot box $\mathcal{BB}$ as empty. Setup outputs an public/private election key pair (pk, sk). We assume that pk includes the sets $\mathcal{V}$ and $\mathcal{C}$.

Register$(id, \mathcal{L}, pk)$ On input voter identity id, set $\mathcal{L}$ and pk, algorithm Register outputs a public/private credential pair (pk_{id}, sk_{id}) and updates set $\mathcal{L}$ with pk_{id} such that $\mathcal{L} \leftarrow \mathcal{L} \cup \{pk_{id}\}$.

Vote$(v, pk_{id}, sk_{id}, pk)$ On input vote v, pk_{id}, sk_{id} and pk, algorithm Vote outputs a ballot b that includes the voter's public credential.

Valid$(b, \mathcal{BB}, \mathcal{L}, pk)$ On input ballot b, ballot box $\mathcal{BB}$, set $\mathcal{L}$ and pk, algorithm Valid outputs 1 if ballot b is accepted to ballot box $\mathcal{BB}$, and 0 otherwise.

Append$(b, \mathcal{BB}, \mathcal{L}, pk)$ On input b, $\mathcal{BB}$, set $\mathcal{L}$ and pk, algorithm Append outputs the updated ballot box to include ballot b.

Publish$(\mathcal{BB})$ On input $\mathcal{BB}$, algorithm Publish outputs bulletin board $\mathcal{PBB}$.

Tally$(\mathcal{BB}, \mathcal{L}, sk)$ On input $\mathcal{BB}$, set $\mathcal{L}$ and sk, algorithm Tally computes and outputs the election result $r \in R$ with a proof π that the result is correct.

Verify$(\mathcal{BB}, \mathcal{L}, r, \pi, pk)$ On input $\mathcal{BB}$, set $\mathcal{L}$, result r, proof π and pk, algorithm Verify outputs 1 if the election result verifies, and 0 otherwise.

[4] Our result function, similar to the result function in [7], determines how the result of the election is computed.

E-voting schemes must satisfy *correctness*, which requires that the result output by algorithm Tally is equivalent to result function f applied to all votes input to algorithm Vote.

Definition 2 (Correctness). *An e-voting scheme Γ defined with respect to a result function f is* correct *if, for any set of n_v voters $\mathcal{V} = \{id_1, \ldots, id_{n_v}\}$ and votes $v_1, \ldots, v_{n_v} \in \mathcal{C}$, there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{c} (pk,sk) \leftarrow \mathsf{Setup}(1^\lambda);\ \mathit{for}\ i{=}1,\ldots n_v: \\ \big\{ (pk_{id_i},sk_{id_i}) \leftarrow \mathsf{Register}(id_i,\mathcal{L},pk);\ b_i \leftarrow \mathsf{Vote}(v_i,pk_{id_i},sk_{id_i},pk); \\ \mathit{if}\ \mathsf{Valid}(b_i,\mathcal{BB},\mathcal{L},pk){=}1:\ \mathcal{BB} \leftarrow \mathsf{Append}(b_i,\mathcal{BB},\mathcal{L},pk) \big\}; \\ (r,\pi) \leftarrow \mathsf{Tally}(\mathcal{BB},\mathcal{L},sk):\ r{=}f((pk_{id_1},v_1),\ldots(pk_{id_{n_v}},v_{n_v})) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

3 Ballot Secrecy in the Honest Model

We introduce BS, a definition of ballot secrecy in which an adversary can *adaptively* corrupt voters, submitting ballots on their behalf, and can submit two votes (the left and right vote) on behalf of honest voters. Thus, BS describes an experiment in which an adversary is provided with access to a bulletin board, and the corresponding election result, that consists of ballots for the left or right vote submitted on behalf of honest voters, in addition to ballots submitted on behalf of corrupted voters. If the adversary cannot determine whether the bulletin board and result contains the left or right votes submitted by honest voters, a scheme is said to satisfy BS.

BS requires a *balancing condition* to ensure that the adversary cannot trivially distinguish views. For example, an adversary in the BS experiment could submit '0' as the left vote and '1' as the right vote on behalf of *all* honest voters. Then, the election result allows the adversary to trivially distinguish the two possible views. We define our balancing condition to prevent such trivial distinctions, and to model adaptive voter corruption. We first describe BS, followed by details of our balancing condition.

The BS experiment $\mathsf{Exp}^{\mathsf{BS}}_{\Gamma,\mathcal{A}}(\lambda)$ for adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, formally defined in Figure 1, proceeds as follows. A challenger initialises sets V_0 and V_1, required to model the balancing condition, as empty, generates the election key pair (pk, sk), and chooses a bit β. Adversary $\mathcal{A}_1$ is given public key pk and proceeds to query a number of oracles, formally defined in Figure 1 and described as follows. We write $\mathcal{O}\mathsf{x}_{(y_1,\ldots,y_n)}(z_1, \ldots, z_n)$ to denote oracle $\mathcal{O}\mathsf{x}$ parametrised by $y_1, \ldots, y_n$ that takes as input $z_1, \ldots, z_n$.

$\mathcal{O}\mathsf{reg}_{(pk,\mathcal{L}\mathcal{Q}\mathsf{reg})}(id)$ registers an eligible voter. If id is in the set of eligible voters $\mathcal{V}$, oracle $\mathcal{O}\mathsf{reg}$ runs algorithm $\mathsf{Register}$ on behalf of id and returns the voter's public credential pk_{id} to $\mathcal{A}_1$. Oracle $\mathcal{O}\mathsf{reg}$ additionally updates a list of queries $\mathcal{Q}\mathsf{reg}$ to include the tuple (id, pk_{id}, sk_{id}).

$\mathcal{O}\mathsf{corrupt}_{(\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt})}(id)$ corrupts a voter. If id is a registered voter, oracle $\mathcal{O}\mathsf{corrupt}$ returns the voter's private credential sk_{id} to $\mathcal{A}_1$, and updates a list of queries $\mathcal{Q}\mathsf{corrupt}$ to include the tuple (id, pk_{id}, sk_{id}).

$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathcal{BB},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},V_0,V_1)}(pk_{id}, v_0, v_1)$ produces and submits a ballot for vote v_β on behalf of an uncorrupted voter. If voter pk_{id} is registered but not corrupt, and v_0, v_1 are valid vote choices, oracle $\mathcal{O}\mathsf{vote}$ runs algorithms Vote and Append on behalf of voter pk_{id} and vote v_β. Oracle $\mathcal{O}\mathsf{vote}$ also updates sets V_0 and V_1 to include votes v_0 and v_1 respectively, and removes any previous entries for pk_{id}, modelling an e-voting scheme with a last-vote-counts revote policy.

$\mathcal{O}\mathsf{cast}_{(pk,\mathcal{L},\mathcal{BB},V_0,V_1)}(pk_{id}, b)$ submits a ballot on behalf of a voter. If ballot b is valid and created for voter pk_{id}, and ballot b does not exist in $\mathcal{BB}$, oracle $\mathcal{O}\mathsf{cast}$ appends ballot b to ballot box $\mathcal{BB}$. Oracle $\mathcal{O}\mathsf{cast}$ also removes any entries in sets V_0 and V_1 for pk_{id}.

$\mathcal{O}\mathsf{board}_{\mathcal{BB}}()$ returns bulletin board $\mathcal{PBB}$ to $\mathcal{A}_1$.

Adversary $\mathcal{A}_1$ outputs state information st, indicating that the experiment should transition to the tallying phase. Upon receiving the result r and proof of correct tallying π, $\mathcal{A}_2$ outputs a bit β'. The experiment returns 1 if $\beta' = \beta$ and the balancing condition is satisfied.

Definition 3 (BS). *An e-voting scheme Γ satisfies* BS *if, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{BS}}_{\Gamma,\mathcal{A}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{BS}}_{\Gamma,\mathcal{A}}(\lambda)$ is the experiment defined in Figure 1.

3.1 Our Balancing Condition

Following a query to oracle $\mathcal{O}\mathsf{vote}$, sets V_0 and V_1 are updated to contain tuples (pk_{id}, v_0) and (pk_{id}, v_1) respectively. After the result of the election is announced, multisets V_0' and V_1' are defined to contain the votes v_0 and v_1 from sets V_0 and V_1 respectively. Our balancing condition, $V_0' = V_1'$, ensures that, for every left-hand vote of an honest voter, there exists an honest voter for whom the same vote is submitted as their right-hand vote. Thus, we prevent trivial distinctions.

This notion of balance is inspired by Benaloh and Yung's early definition of ballot secrecy [5] and is used by Bernhard and Smyth in their ballot secrecy definition $\mathsf{IND-SEC}$ [10]. However, in comparison to BS, neither $\mathsf{IND-SEC}$ nor Benaloh and Yung's approach model registration of voters. It transpires that, to capture registration of voters and, in particular, to capture revoting and adaptive corruption of voters for e-voting schemes with voter registration, the balancing condition described above must include more complex features. We describe these subtleties and demonstrate their necessity through examples.

Eliminating entries from $\mathcal{O}$ vote. We require that, following a query to $\mathcal{O}\mathsf{vote}$ on behalf of a voter pk_{id}, previous entries containing pk_{id} are removed from sets V_0 and V_1. Else, it is possible that an adversary can submit oracle queries that

```
$\mathsf{Exp}^{\mathsf{BS}}_{\Gamma,\mathcal{A}}(\lambda)$

$V_0, V_1, \mathcal{Q}\mathsf{reg}, \mathcal{Q}\mathsf{corrupt} \leftarrow \emptyset$
$(pk, sk) \leftarrow \mathsf{Setup}(1^\lambda)$
$\beta \leftarrow \{0,1\}$
$st \leftarrow \mathcal{A}_1^{\mathcal{O}}(pk)$
$(r, \pi) \leftarrow \mathsf{Tally}(\mathcal{BB}, \mathcal{L}, sk)$
$\beta' \leftarrow \mathcal{A}_2(r, \pi, \mathcal{PBB}, st)$
for $i \in \{0,1\}$
    $V'_i \leftarrow \{\{v_i | (\cdot, v_i) \in V_i\}\}$
if $\beta' = \beta \wedge V'_0 = V'_1$
    return $1$
else
    return $0$
```

```
$\mathcal{O}\mathsf{reg}_{(pk,\mathcal{L}\mathcal{Q}\mathsf{reg})}(id)$

if $id \notin \mathcal{V} \vee (id, \cdot, \cdot) \in \mathcal{Q}\mathsf{reg}$   return $\perp$
$(pk_{id}, sk_{id}) \leftarrow \mathsf{Register}(id, \mathcal{L}, pk)$
$\mathcal{Q}\mathsf{reg} \leftarrow \mathcal{Q}\mathsf{reg} \cup \{(id, pk_{id}, sk_{id})\}$
return $pk_{id}$
```

```
$\mathcal{O}\mathsf{board}_{\mathcal{BB}}()$

return $\mathsf{Publish}(\mathcal{BB})$
```

```
$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathcal{BB},\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt},V_0,V_1)}(pk_{id}, v_0, v_1)$

if $(\cdot, pk_{id}, \cdot) \notin \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt} \vee v_0, v_1 \notin \mathcal{C}$   return $\perp$
for $sk_{id}.(\cdot, pk_{id}, sk_{id}) \in \mathcal{Q}\mathsf{reg}$
    $b \leftarrow \mathsf{Vote}(v_\beta, pk_{id}, sk_{id}, pk)$
$\mathcal{BB} \leftarrow \mathsf{Append}(b, \mathcal{BB}, \mathcal{L}, pk)$
for $i \in \{0,1\}$
    if $\exists (pk_{id}, \cdot) \in V_i$
        $V_i \leftarrow V_i \setminus \{(pk_{id}, \cdot)\}$
$V_0 \leftarrow V_0 \cup \{(pk_{id}, v_0)\}$
$V_1 \leftarrow V_1 \cup \{(pk_{id}, v_1)\}$
return $\top$
```

```
$\mathcal{O}\mathsf{cast}_{(pk,\mathcal{L},\mathcal{BB},V_0,V_1)}(pk_{id}, b)$

if $\mathsf{Valid}(b, \mathcal{BB}, \mathcal{L}, pk) = 0 \vee pk_{id} \notin b \vee b \in \mathcal{BB}$   return $\perp$
$\mathcal{BB} \leftarrow \mathsf{Append}(b, \mathcal{BB}, \mathcal{L}, pk)$
for $i \in \{0,1\}$
    if $\exists (pk_{id}, \cdot) \in V_i$
        $V_i \leftarrow V_i \setminus \{(pk_{id}, \cdot)\}$
return $\top$
```

```
$\mathcal{O}\mathsf{corrupt}_{(\mathcal{Q}\mathsf{reg},\mathcal{Q}\mathsf{corrupt})}(id)$

if $(id, \cdot, \cdot) \notin \mathcal{Q}\mathsf{reg}$   return $\perp$
$\mathcal{Q}\mathsf{corrupt} \leftarrow \mathcal{Q}\mathsf{corrupt} \cup \{(id, pk_{id}, sk_{id})\}$
return $sk_{id}$
```

Fig. 1: The ballot secrecy experiment $\mathsf{Exp}^{\mathsf{BS}}_{\Gamma,\mathcal{A}}(\lambda)$ where $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has access to oracles $\mathcal{O} = \{\mathcal{O}\mathsf{reg}, \mathcal{O}\mathsf{corrupt}, \mathcal{O}\mathsf{vote}, \mathcal{O}\mathsf{cast}, \mathcal{O}\mathsf{board}\}$.

allow trivial distinctions, even if a scheme is intuitively ballot secret. Consider an e-voting scheme that employs a last-vote-counts revote policy and allows voters to cast a ballot for '0' or '1'. Let the election result be a vector of size two, indicating the number of votes cast for each candidate. Assume that $\mathcal{O}\mathsf{vote}$ does *not* remove any entries from sets V_0 or V_1. Then an adversary in the BS experiment can query $\mathcal{O}\mathsf{vote}(pk_{id_1}, 0, 1)$, $\mathcal{O}\mathsf{vote}(pk_{id_2}, 1, 0)$, $\mathcal{O}\mathsf{vote}(pk_{id_1}, 1, 0)$ and $\mathcal{O}\mathsf{vote}(pk_{id_3}, 0, 1)$ for pk_{id_1}, pk_{id_2} and pk_{id_3} obtained via queries to $\mathcal{O}\mathsf{reg}$ such that $V_0 = \{(pk_{id_1}, 0), (pk_{id_2}, 1), (pk_{id_1}, 1), (pk_{id_3}, 0)\}$ and $V_1 = \{(pk_{id_1}, 1), (pk_{id_2}, 0), (pk_{id_1}, 0), (pk_{id_3}, 1)\}$. Subsequently, $V'_0 = V'_1$ and the balancing condition is satisfied. However, if $\beta = 0$, $r = (1, 2)$ and, if $\beta = 1$, $r = (2, 1)$. Then the adversary trivially distinguishes the two views and succeeds in the BS experiment. Therefore, it is essential that $\mathcal{O}\mathsf{vote}$ removes previous entries that contain pk_{id} from sets V_0 and V_1. Indeed, if the first entry of V_0 and V_1 is removed, the balancing condition is not satisfied and the adversary cannot succeed in the BS experiment.

Eliminating entries from $\mathcal{O}\mathsf{cast}$. Similarly, following a query to $\mathcal{O}\mathsf{cast}$ on behalf of voter pk_{id}, it is essential that entries containing pk_{id} are removed from sets V_0 and V_1. In fact, rather than the second two queries to $\mathcal{O}\mathsf{vote}$ in the example above, the adversary can query $\mathcal{O}\mathsf{cast}(pk_{id_1}, b)$ where b encodes a vote for '1'. Then, if $\mathcal{O}\mathsf{cast}$ does not remove the previous entry for pk_{id_1}, the balancing condition is satisfied. Yet, the result $r = (0, 2)$ (if $\beta = 0$) or $r = (1, 1)$ (if $\beta = 1$). Under static corruption of voters, removal of entries is not necessary as an adversary cannot make a query to $\mathcal{O}\mathsf{vote}$ on behalf of a corrupted voter, i.e., voter pk_{id_1} in the example. Thus, removal of previous entries is key to ensure that our balancing condition allows for adaptive corruption of voters.

Voting policies. Our balancing condition models a last-vote-counts revote policy, a policy applied to implemented e-voting schemes, for example, the Estonian i-Voting scheme [25]. On the other hand, it is common to implement an e-voting scheme with a no revote policy. For instance, Helios [1, 28] allows for both a last-vote-counts and a no revote policy. For this reason, we briefly describe how our balancing condition can be modified in a straightforward way to account for a no revote policy as follows. $\mathcal{O}\mathsf{vote}$ does not remove previous entries containing pk_{id} from sets V_0 and V_1 following a query on behalf of voter pk_{id}. Additionally, $\mathcal{O}\mathsf{vote}$ adds new entries to sets V_0 and V_1 only if the sets do not contain an entry on behalf of pk_{id}. Finally, $\mathcal{O}\mathsf{cast}$ does not update sets V_0 and V_1.

3.2 Satisfiability of BS

We demonstrate satisfiability of BS by constructing an e-voting scheme $\mathit{\Gamma}_{\mathsf{mini}}$, defined formally in Figure 2. $\mathit{\Gamma}_{\mathsf{mini}}$ relies on a homomorphic public key encryption scheme $\mathit{\Pi}$ and a signature of knowledge SOK, both of which are formally defined in Appendix A. A voter with credential pair (pk_{id}, sk_{id})[5] casts a ballot for $v \in \{0, 1\}$[6] by producing an encryption, denoted c, of v under $\mathit{\Pi}$, and generating a signature of knowledge, denoted σ, that the resulting ciphertext encrypts $v \in \{0, 1\}$ using their private credential under SOK. The form of a ballot b is (pk_{id}, c, σ). If ciphertext c does not appear in a ballot on the ballot box, and signature of knowledge σ verifies, the ballot is appended to the ballot box. The ballot box and the bulletin board are identical in this scheme. To compute the result, ciphertext c is extracted from the final ballot cast by each voter. The extracted ciphertexts are homomorphically tallied and the homomorphic ciphertext is decrypted, giving the result of the election. As we do not focus on verifiability, we simply define algorithm Tally to output $\perp$ in place of a proof of correct tallying, and algorithm Verify is defined to always return 1.

To satisfy BS, we require that $\mathit{\Pi}$ satisfies non-malleable-CPA security [3], NM-CPA, and SOK satisfies extractability [13]. We define these security properties in Appendix A. We obtain the result in Theorem 1.

[5] We write that credential pairs are generated by a one-way function f.

[6] This can be extended to multi-candidate elections in the style of Helios [1, 28].

```
$\mathsf{Setup}(1^\lambda)$
$\mathcal{BB} \leftarrow ()$
$\mathcal{L} \leftarrow \emptyset$
$\mathcal{V} = \{id_1, \ldots, id_{n_v}\}$
$\mathcal{C} \leftarrow \{0,1\}$
$(pk_\Pi, sk_\Pi) \leftarrow \Pi.\mathsf{Gen}(1^\lambda)$
$pp \leftarrow \mathsf{SoK.Setup}(1^\lambda)$
$pk \leftarrow (pk_\Pi, pp, \mathcal{V}, \mathcal{C})$
$sk \leftarrow (sk_\Pi, pk)$
return $(pk, sk)$
```

```
$\mathsf{Register}(id, \mathcal{L}, pk)$
parse $pk$ as $(pk_\Pi, pp, \mathcal{V}, \mathcal{C})$
$(pk_{id}, sk_{id}) \leftarrow f(pp)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{pk_{id}\}$
return $(pk_{id}, sk_{id})$
```

```
$\mathsf{Vote}(v, sk_{id}, pk_{id}, pk)$
parse $pk$ as $(pk_\Pi, pp, \mathcal{V}, \mathcal{C})$
$c \leftarrow \Pi.\mathsf{Enc}(pk_\Pi, v; r)$
$\sigma \leftarrow \mathsf{SoK.Sign}(pp, (c, pk_\Pi, pk_{id}), (sk_{id}, r), c)$
$b \leftarrow (pk_{id}, c, \sigma)$
return $b$
```

```
$\mathsf{Valid}(b, \mathcal{BB}, pk, \mathcal{L})$
parse $b$ as $(c, \sigma, pk_{id})$
parse $pk$ as $(pk_\Pi, pp, \mathcal{V}, \mathcal{C})$
if $pk_{id} \notin \mathcal{L} \vee \exists b^* \in \mathcal{BB} : c = b^*[2] \vee \mathsf{SoK.Verify}(pp, (c, pk_\Pi, pk_{id}), \sigma, c) = 0$
  return 0
else
  return 1
```

```
$\mathsf{Append}(b, \mathcal{BB}, pk, \mathcal{L})$
if $\mathsf{Valid}(b, \mathcal{BB}, pk, \mathcal{L}) = 0$
  return $\mathcal{BB}$
else
  return $\mathcal{BB} \leftarrow \mathcal{BB} \parallel b$
```

```
$\mathsf{Publish}(\mathcal{BB})$
return $\mathcal{BB}$
```

```
$\mathsf{Verify}(\mathcal{BB}, r, \pi, pk)$
return 1
```

```
$\mathsf{Tally}(\mathcal{BB}, \mathcal{L}, sk)$
parse $sk$ as $(sk_\Pi, pk)$
$\mathcal{BB}' \leftarrow ()$
for $i = 1, \ldots |\mathcal{BB}|$
  parse $\mathcal{BB}[i]$ as $(pk_{id}, c, \sigma)$
  if $\mathsf{Valid}(\mathcal{BB}[i], \mathcal{BB}, pk, \mathcal{L}) = 1 \wedge \mathcal{BB}[j] \neq (pk_{id}, \cdot, \cdot) \forall j = i+1, \ldots, |\mathcal{BB}|$
    $\mathcal{BB}' \leftarrow \mathcal{BB}' \parallel c$
$\mathsf{cipher} = \sum_{i=1}^{|\mathcal{BB}'|} \mathcal{BB}'[i]$
$r = \Pi.\mathsf{Dec}(sk_\Pi, \mathsf{cipher})$
return $(r, \perp)$
```

Fig. 2: The e-voting scheme Γ_{mini} constructed from public key encryption scheme Π and signature of knowledge SOK.

Theorem 1. *Γ_{mini} (Figure 2) satisfies* BS *if public key encryption scheme Π satisfies* $\mathsf{NM\text{-}CPA}$ *and signature of knowledge* SOK *satisfies extractability.*

We formally prove Theorem 1 in Appendix B. Informally, assuming that an adversary queries $\mathcal{O}\mathsf{vote}$ and $\mathcal{O}\mathsf{cast}$ such that $V_0' = V_1'$, the result computed over $\mathcal{BB}$ in the BS experiment for $\beta = 0$ is indistinguishable from the result for $\beta = 1$. In fact, with respect to the ballots included in the election result, $\mathcal{BB}$ contains the same number of votes for each candidate, regardless of β. Moreover, an adversary cannot distinguish whether $\mathcal{BB}$ contains ballots for a left- or right-hand vote on behalf of honest voters if Π satisfies $\mathsf{NM\text{-}CPA}$. Finally, $\mathsf{NM\text{-}CPA}$ security of Π and extractability of SOK ensures that an adversary cannot submit ballots on behalf of a corrupt voter that are meaningfully related to the ballots of honest voters, thus skewing the result and allowing the adversary to distinguish views [21].

3.3 Limitation of BS

For transparency, we elaborate on an aspect of BS that limits the class of e-voting schemes that can be captured by BS. We believe that such discussion is essential to understand security definitions for e-voting and ensure that, to prove security of an e-voting scheme, the most relevant definition is chosen based on the nuances of the scheme. In Section 5, we elaborate on the complexities and limitations of ballot secrecy definitions in the literature, further casting light on the applicability of definitions.

Our balancing condition limits the class of result functions that can be captured. In particular, consider an e-voting scheme in which a voter can submit a score for a candidate, e.g., $\mathcal{C} = \{0, 1, 2\}$, and the result consists of the sum of all scores submitted. In [7], the authors show that Benaloh and Yung's ballot secrecy definition [5], upon which BS is based, does not capture this result function. Specifically, [5] and BS do not model an attacker that can distinguish different vote assignments that lead to the same result, for example, two voters voting 0 and 2 respectively, and both voters voting 1. Despite this, common result functions, such as plurality voting, are within the scope of our definition.

4 Extending BS to the Malicious Setting

Corrupt election officials can break ballot secrecy. In particular, a malicious *ballot box* can 'stuff' the ballot box with ballots, which can lead to attacks against ballot secrecy [21]. Furthermore, a malicious *tallier* can potentially reveal the votes of all honest voters, for example, if ballots consist of a vote encrypted under the tallier's public key, such as in our construction Γ_{mini}. To overcome this, many e-voting schemes distribute the role of the tallier [29, 31] and assume that a proportion of talliers are honest. We define mbbBS, a definition that extends BS to the malicious ballot box setting, and dtBS, an extension of mbbBS in which the adversary can further corrupt a subset of talliers where the role of the tallier is distributed. In doing so, we provide definitions that allow a scheme designer to prove ballot secrecy in the event of an attacker that can corrupt the ballot box *and* a proportion of talliers.

4.1 Malicious Ballot Box Manager

We extend BS to mbbBS, defining an adversary that arbitrarily constructs the ballot box, obtaining ballots for honest voters that correspond to either a left or right vote submitted to an oracle $\mathcal{O}\mathsf{vote}$. Hence, mbbBS models a corrupt ballot box, but considers all other election entities to be honest. We note that, in this setting, the adversary can only *statically* corrupt voters, a common restriction [16, 20]. This arises as a consequence of the balancing condition, which we elaborate on following the formal definition.

The mbbBS experiment $\mathsf{Exp}^{\mathsf{mbbBS}}_{\Gamma,\mathcal{A}}(\lambda)$ for adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, formally defined in Figure 3, registers all n_v eligible voters and provides the adversary $\mathcal{A}_1$ with the set of public credentials $\mathcal{L}$ and the election public key pk. $\mathcal{A}_1$ selects a subset of public credentials $\mathsf{corr}\mathcal{L}$ to corrupt and $\mathcal{A}_2$ receives a list of corresponding private credentials $\mathsf{c}\mathcal{L}$. Adversary $\mathcal{A}_2$ is provided with access to an oracle $\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{corr}\mathcal{L},V)}(pk_{id}, v_0, v_1)$ that returns ballots for v_β on behalf of honest voters, and constructs a ballot box $\mathcal{BB}$, which may include both honestly and maliciously generated ballots. Upon receiving the result r computed over

$\mathcal{BB}$ and a proof of correct tallying π, $\mathcal{A}_3$ outputs a bit β'. If $\beta' = \beta$ and the balancing condition is satisfied, the experiment returns 1. We observe that the adversary does not require access to oracles $\mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{corrupt}$, as defined for BS, because voters are statically corrupted. Moreover, the adversary does not require access to an oracle $\mathcal{O}\mathsf{cast}$ because $\mathcal{A}_2$ constructs the ballot box.

Definition 4 **(mbbBS).** *An e-voting scheme Γ satisfies* mbbBS *if, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{mbbBS}}_{\Gamma,\mathcal{A}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{mbbBS}}_{\Gamma,\mathcal{A}}(\lambda)$ is the experiment defined in Figure 3.

```
$\mathsf{Exp}^{\mathsf{mbbBS}}_{\Gamma,\mathcal{A}}(\lambda)$
$V \leftarrow \{\}$
$V_0, V_1 \leftarrow \{\{\}\}$
$\beta \leftarrow \{0,1\}$
$(pk, sk) \leftarrow \mathsf{Setup}(1^\lambda)$
for $i = 1, \ldots, n_v$
  $(pk_{id_i}, sk_{id_i}) \leftarrow \mathsf{Register}(id_i, \mathcal{L}, pk)$
$L \leftarrow \{(pk_{id_1}, sk_{id_1}), \ldots, (pk_{id_{n_v}}, sk_{id_{n_v}})\}$
$(\mathsf{corr}\mathcal{L}, st_1) \leftarrow \mathcal{A}_1(pk, \mathcal{L})$
$\mathsf{c}\mathcal{L} \leftarrow \{(pk_{id}, sk_{id}) | pk_{id} \in \mathsf{corr}\mathcal{L} \cap \mathcal{L}\}$
$(\mathcal{BB}, st_2) \leftarrow \mathcal{A}_2^{\mathcal{O}\mathsf{vote}}(\mathsf{c}\mathcal{L}, st_1)$
$(r, \pi) \leftarrow \mathsf{Tally}(\mathcal{BB}, \mathcal{L}, sk)$
$\beta' \leftarrow \mathcal{A}_3(r, \pi, \mathcal{BB}, st_2)$
for $i = 1, \ldots, |\mathcal{BB}|$
  if $\exists(pk_{id}, v_0, v_1, \mathcal{BB}[i]) \in V$
    for $j = i+1, \ldots, |\mathcal{BB}|$
      if $\nexists(pk_{id}, *, *, \mathcal{BB}[j]) \in V$
        $V_0 \leftarrow V_0 \cup \{\{v_0\}\}$
        $V_1 \leftarrow V_1 \cup \{\{v_1\}\}$
if $\beta' = \beta \wedge V_0 = V_1$
  return 1
else
  return 0
```

```
$\mathcal{O}\mathsf{vote}_{(pk,\mathcal{L},\mathsf{corr}\mathcal{L},L,V)}(pk_{id}, v_0, v_1)$
if $pk_{id} \notin \mathcal{L} \setminus \mathsf{corr}\mathcal{L} \vee v_0, v_1 \notin \mathcal{C}$   return $\perp$
for $sk_{id}.(pk_{id}, sk_{id}) \in L$
  $b \leftarrow \mathsf{Vote}(v_\beta, pk_{id}, sk_{id}, pk)$
$V \leftarrow V \cup \{(pk_{id}, v_0, v_1, b)\}$
return $b$
```

Fig. 3: The malicious ballot box ballot secrecy experiment $\mathsf{Exp}^{\mathsf{mbbBS}}_{\Gamma,\mathcal{A}}(\lambda)$ where $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ has access to oracle $\mathcal{O}\mathsf{vote}$.

Our Balancing Condition. mbbBS maintains a list of queries to $\mathcal{O}\mathsf{vote}$ in a set V such that each entry in V consists of a tuple (pk_{id}, v_0, v_1, b). Then, if b appears on ballot box $\mathcal{BB}$ and the tuple contains the final ballot that appears on $\mathcal{BB}$ with respect to voter pk_{id}, the experiment adds v_0 (resp., v_1) to a multiset V_0 (resp., V_1). In other words, multisets V_0 and V_1 contain only the final vote for every honest voter such that a corresponding ballot is appended to $\mathcal{BB}$.[7]

As a result of our balancing condition, mbbBS considers *static* corruption of voters. Indeed, if mbbBS allows *adaptive* corruption of voters, a trivial distinguishing attack is possible. To demonstrate this, we recall our construction $\mathit{\Gamma}_{\mathsf{mini}}$ (Figure 2) and assume that an adversary in the mbbBS experiment can adaptively corrupt voters. That is, the adversary has access to oracles $\mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{corrupt}$ as defined in Figure 1. Then, the adversary queries $b_1 \leftarrow \mathcal{O}\mathsf{vote}(pk_{id_1}, 0, 1)$ and $b_2 \leftarrow \mathcal{O}\mathsf{vote}(pk_{id_2}, 1, 0)$ for pk_{id_1} and pk_{id_2} obtained via queries to $\mathcal{O}\mathsf{reg}$. Voter pk_{id_1} is corrupted via a query to $\mathcal{O}\mathsf{corrupt}$ and the adversary appends b_1, b_2 and b_3, a ballot for '1' on behalf of pk_{id_1}, to $\mathcal{BB}$. The balancing condition is satisfied but, if $\beta = 0$ (resp., $\beta = 1$), $r = (0, 2)$ (resp., $r = (1, 1)$), and the adversary can trivially distinguish the two views. Consequently, we restrict mbbBS to allow only static corruption of voters. Then, if the adversary wishes to corrupt pk_{id_1}, they cannot make queries to $\mathcal{O}\mathsf{vote}$ on behalf of pk_{id_1} and, in the example above, the balancing condition is not satisfied. Therefore, the adversary does not succeed in the mbbBS experiment.

Satisfiability of mbbBS. We show that our e-voting construction $\mathit{\Gamma}_{\mathsf{mini}}$ (Figure 2) satisfies mbbBS under the same conditions that $\mathit{\Gamma}_{\mathsf{mini}}$ satisfies BS. Indeed, we design the ballots in $\mathit{\Gamma}_{\mathsf{mini}}$ to be *non-malleable*, which is required to satisfy mbbBS, a fact that we elaborate on following our formal result. In other words, we intentionally design $\mathit{\Gamma}_{\mathsf{mini}}$ to satisfy stronger notions of ballot secrecy than BS. We require a $\mathsf{NM\text{-}CPA}$ secure public key encryption scheme and a signature of knowledge that satisfies extractability. We obtain the result in Theorem 2.

Theorem 2. $\mathit{\Gamma}_{\mathsf{mini}}$ *(Figure 2) satisfies* mbbBS *if public key encryption scheme* $\mathit{\Pi}$ *satisfies* $\mathsf{NM\text{-}CPA}$ *and signature of knowledge* SOK *satisfies extractability.*

We formally prove Theorem 2 in Appendix B. The proof of Theorem 2 is very similar to the proof that $\mathit{\Gamma}_{\mathsf{mini}}$ satisfies BS (Theorem 1). In particular, an adversary cannot distinguish whether $\mathcal{O}\mathsf{vote}$ returns a ballot corresponding to v_0 or v_1 as a result of $\mathsf{NM\text{-}CPA}$ security of $\mathit{\Pi}$. Moreover, the ballots returned by $\mathcal{O}\mathsf{vote}$ cannot be modified by the adversary in a meaningful way due to non-malleability of the ballot, provided by extractability of SOK and $\mathsf{NM\text{-}CPA}$ security of $\mathit{\Pi}$. Thus, any ballot in $\mathcal{BB}$ is either a ballot of a corrupt voter that is independent of any other ballot, or an output of oracle $\mathcal{O}\mathsf{vote}$. Then, if the balancing condition is satisfied, the result computed over $\mathcal{BB}$ constructed by the adversary is indistinguishable for $\beta = 0$ or 1.

[7] As with BS, our balancing condition can be modified to model a no-revote policy.

As noted, ballots must be non-malleable to satisfy mbbBS. Intuitively, if an attacker with control of the ballot box can transform ballots, then they can append ballots to the ballot box that are meaningfully related to the vote of an honest voter such that the result reveals the honest voter's vote. We demonstrate this by describing an attack against our e-voting scheme Γ_{mini} where we replace the signature of knowledge with a standard digital signature scheme. An adversary in the mbbBS experiment can query $b_1 \leftarrow \mathcal{O}\mathsf{vote}(pk_{id_1}, 0, 1)$, $b_2 \leftarrow \mathcal{O}\mathsf{vote}(pk_{id_2}, 1, 0)$ and $b_3 \leftarrow \mathcal{O}\mathsf{vote}(pk_{id_3}, 0, 1)$, appending ballots b_1 and b_2 to $\mathcal{BB}$. Let $b_3 = (pk_{id_3}, c_3, \sigma_3)$. The adversary produces a signature σ_4 for ciphertext c_3 for a corrupt voter pk_{id_4} and appends the modified ballot $b^* = (pk_{id_4}, c_3, \sigma_4)$ to $\mathcal{BB}$. The balancing condition is satisfied, yet, if $\beta = 0$, $r = (2, 1)$ and, if $\beta = 1$, $r = (1, 2)$, which allows $\mathcal{A}$ to distinguish the two views. This attack is possible because the ballot is malleable. In contrast, if σ is a signature of knowledge, the adversary requires knowledge of the plaintext encrypted by c_3 in order to construct a signature of knowledge for pk_{id_4}. Therefore, we conclude that e-voting schemes that allow malleable ballots cannot satisfy mbbBS.

We recognise that there exists e-voting schemes for which the ballots are malleable, for example, e-voting schemes that include a timestamp in the ballot. An adversary in the mbbBS experiment can modify the timestamp in a ballot output by $\mathcal{O}\mathsf{vote}$ and append the modified ballot to $\mathcal{BB}$. Then, the result trivially reveals β and the balancing condition holds. However, there may not be a ballot secrecy issue with the scheme in practice. Therefore, if a scheme produces ballots that contain a malleable part, the scheme does not satisfy mbbBS, despite the fact that it is intuitively ballot secret. Despite this, we believe that non-malleable ballots are desirable. For example, if a ballot includes a malleable timestamp, an attacker with control of the ballot box can modify the timestamp, potentially ensuring that a ballot is not included in the result of the election. Furthermore, ballots with malleable elements can be modified to ensure non-malleability. For instance, a ballot can include a signature of knowledge or proof of knowledge that ties the signature or proof to the malleable element, ensuring that the malleable element cannot be modified without detection (which, in turn, ensures that a modified ballot is not valid).

4.2 Distributed and Malicious Tallier

We now consider an extension of mbbBS for e-voting schemes with a distributed tallier, that is, we write tallier $\mathcal{T}$ as $\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_n)$. In this case, we consider an election private key that is distributed amongst n talliers such that $sk = (sk_{\mathcal{T}_1}, \ldots, sk_{\mathcal{T}_n})$ and at least t shares are required to reconstruct sk where $t \leq n$.[8] We extend mbbBS to a definition dtBS that models a corrupt ballot box *and* a subset of corrupt talliers. In particular, we model an attacker that obtains the private key share of up to $t - 1$ talliers. As with mbbBS, we consider other election entities to be honest and only allow the static corruption of voters.

[8] We note that, specifically, $t = n$ is possible. That is, all n shares are required to reconstruct the private key.

The corruption strategy captured by dtBS does not model an attacker that generates key shares for corrupt talliers. In fact, we consider that all key shares are generated honestly. In other words, the attacker corrupts talliers *after* key generation and, therefore, cannot influence the generation of key shares. As dtBS is a preliminary exploration of ballot secrecy with a malicious ballot box and tallier, we consider the attack strategy captured by dtBS to be appropriate and leave the case of stronger attacker models as an open problem.

We define the dtBS experiment $\mathsf{Exp}^{\mathsf{dtBS}}_{\Gamma,\mathcal{A}}(\lambda, t, n)$, parametrised by the number of talliers n and the number of shares required to reconstruct the election private key t, as the mbbBS experiment, defined in Figure 3, but with the following modifications. In addition to statically corrupting a subset of voters, adversary $\mathcal{A}$ corrupts $t-1$ talliers. That is, $\mathcal{A}$ submits $t-1$ unique indices $\{i_1, \ldots, i_{t-1}\}$ and obtains the set of private key shares $\{sk_{\mathcal{T}_{i_1}}, \ldots, sk_{\mathcal{T}_{i_{t-1}}}\}$. Additionally, algorithm Tally takes as input t private key shares that include the $t-1$ shares returned to $\mathcal{A}$. In all other respects, the dtBS experiment is identical to the mbbBS experiment.

Definition 5 (dtBS). *An e-voting scheme Γ for n talliers and threshold t, where election private key $sk = (sk_{\mathcal{T}_1}, \ldots, sk_{\mathcal{T}_n})$, satisfies* dtBS *if, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{dtBS}}_{\Gamma,\mathcal{A}}(\lambda, t, n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{dtBS}}_{\Gamma,\mathcal{A}}(\lambda, t, n)$ is the mbbBS *experiment defined in Figure 3 for $\mathcal{A}$ provided with $t-1$ private key shares of their choice and where algorithm* Tally *takes as input the $t-1$ private keys provided to $\mathcal{A}$.*

Satisfiability of dtBS. To illustrate satisfiability of dtBS, we consider a modification to Γ_{mini} (Figure 2) that uses a (t, n)-threshold public key encryption scheme [24], Φ, which we define in Appendix A. We call our modified construction Γ'_{mini}. Formally, Γ'_{mini} is identical to Γ_{mini} with the exceptions of the following modifications to algorithms Setup, Vote and Tally. We write that Setup takes additional input t and n and, rather than running algorithm $\Pi.\mathsf{Gen}$, Setup runs algorithm $\Phi.\mathsf{Gen}$ to generate a public key pk_ϕ and n private keys $sk_{\phi_1}, \ldots, sk_{\phi_n}$. Each tallier is provided with a private key. Algorithm Vote encrypts vote v by running $\Phi.\mathsf{Enc}$, rather than $\Pi.\mathsf{Enc}$. Finally, algorithm Tally requires interaction between t talliers. In detail, any tallier can produce the homomorphic ciphertext cipher, and, then, t talliers each produce a partial decryption of cipher by running algorithm $\Phi.\mathsf{Dec}$. The final result r is computed by running algorithm $\Phi.\mathsf{Combine}$ on input of the t partial decryptions.

As for our previous results, to satisfy dtBS, we require that Φ satisfies t-$\mathsf{NM\text{-}CPA}$ security and SOK satisfies extractability. We define t-$\mathsf{NM\text{-}CPA}$ security for a (t, n)-threshold encryption scheme in Appendix A. Briefly, security of a (t, n)-threshold encryption scheme is defined as a natural extension of security for a standard encryption scheme with the exception that the adversary can statically corrupt $t-1$ decryption servers, see, for example [26, 32]. Therefore,

t-NM-CPA security for Φ is equivalent to NM-CPA security for a standard public key encryption scheme but the adversary obtains the private keys of $t-1$ decryption servers of their choice. We obtain the result in Theorem 3.

Theorem 3. *Γ'_{mini} satisfies* dtBS *if (t,n)-threshold public key encryption scheme Φ satisfies t*-NM-CPA *and signature of knowledge* SOK *satisfies extractability.*

We formally prove Theorem 3 in Appendix B. The proof of this result follows largely from the fact that Γ_{mini} satisfies mbbBS and, as such, the result and output of $\mathcal{O}$vote is indistinguishable for $\beta = 0$ or 1. Moreover, by t-NM-CPA security of the threshold encryption scheme, access to $t-1$ private keys does not provide the adversary with any more information about the votes of honest voters.

5 A Comparison of Ballot Secrecy Definitions

In this section, we provide a comparison of existing game-based definitions, grouping them according to their underlying intuition. In particular, we identify two types of definitions: those that tally the 'real' election, and those that rely on a balancing condition and tally the viewed election. We place our definitions (Definitions 3-5) in context, highlighting our contribution to the area.

5.1 Tally the 'Real' Election

Recall from the introduction that definitions in this category provide the adversary with a view of a real or fake election, depending on the value of a coin flip, and always compute the tally for the real election. This approach to defining ballot secrecy was introduced in [8], and refined in [9]. In [7], Cortier *et al.* reviewed ballot secrecy definitions from the literature and defined BPRIV as an alteration of [8, 9], avoiding weaknesses found in both previous, related, definitions.[9] Arguably, BPRIV has since become the most well-known and widely used definition of ballot secrecy in the literature. In fact, in [6], it was used to prove the security of Helios and has been extended to capture receipt-freeness [12]. Moreover, it has been extended to model e-voting schemes with registration of voters [16] and to capture a malicious ballot box [20]. As BPRIV is the state-of-the-art definition in this category, we focus on BPRIV and its extensions from [16] and [20]. BPRIV avoids the limitations of existing definitions, including those definitions that follow the 'tally the viewed election' approach (see §5.2). Moreover, BPRIV is shown to imply a simulation-based notion of ballot secrecy [7], reinforcing the correctness and strength of the BPRIV approach. Despite this, we highlight two drawbacks of this approach.

[9] For full details of the review and weakness found in [8, 9], and other ballot secrecy definitions, consult [7].

Need for Additional Properties. BPRIV is strong and well-established, yet, as a stand-alone definition, it is subject to attacks, as highlighted by Cortier *et al.*, the authors of BPRIV [7]. Specifically, BPRIV does not capture an attacker that can cause the rejection of honestly created ballots, which can violate ballot secrecy. Cortier *et al.* define an e-voting scheme such that ballots are appended with a bit, 0 or 1, where algorithm Vote always appends a 0. Then, if there exists a ballot in $\mathcal{BB}$ that is appended with 1, all subsequent ballots are rejected. As BPRIV always computes the result of the 'real' election, such a scheme satisfies BPRIV. Yet, an attacker can ensure that a majority of honestly created ballots are rejected, potentially revealing the votes of a small subset of honest voters. Therefore, Cortier *et al.* define strong correctness, an additional property required to prevent such attacks. Our definition BS, and other definitions in the same category [4, 5, 10, 11, 19, 30], on the other hand, capture this attack. In fact, the balancing condition of BS ensures that votes of honest voters are added to multisets V_0' and V_1', even if the ballot is rejected by algorithm Valid, yet the votes contained in these multisets will not necessarily be included in the result. As a consequence, the adversary can output a ballot box such that the balancing condition is satisfied, and determine β from the result computed over $\mathcal{BB}$.

Furthermore, Cortier *et al.* highlight that BPRIV is subject to another attack in which the ballots of honest voters are *not* included in the result, potentially revealing the vote of an honest voter, and describe an e-voting scheme for a referendum that rejects the ballot of the first voter if the ballot is for a specified candidate [7]. Then, depending on whether the ballot is included in the result, it is possible to determine how the first voter voted. This scheme satisfies BPRIV despite the fact that, intuitively, it is not ballot secret. Therefore, BPRIV must be accompanied by a second additional property, strong consistency, that prevents this attack. By contrast, BS and [4, 5, 10, 11, 19, 30] capture this attack. By defining an adversary that submits a query to $\mathcal{O}$vote such that the left-hand vote is for the specified candidate and the right-hand vote is for a second candidate, the balancing condition can be satisfied and the result returned to the adversary in the BS experiment reveals whether the first ballot was removed. We additionally note that definitions derived from BPRIV [12, 16, 20] also require strong consistency and strong correctness to capture the two attacks outlined above.

Extendibility. For schemes with a registration phase, BPRIV has been extended to *static* voter corruption only [16]. In fact, extending BPRIV to an e-voting scheme with a registration phase is non-trivial and attempting to model adaptive corruption of voters in a logical fashion (e.g., by providing access to a corrupt oracle as in our definition BS) results in a definition that is too strong [16]. By contrast, BS captures adaptive corruption.

BPRIV is also difficult to extend to the malicious ballot box setting, though a recent attempt was made in [20]. There, $\mathcal{BB}_\beta$ is the ballot box created by the adversary in the BPRIV experiment for $\beta \in \{0, 1\}$. Briefly, if $\beta = 0$, the result and tallying proof are returned for $\mathcal{BB}_0$. If $\beta = 1$, the experiment returns the tally computed over $\mathcal{BB}_0$ such that $\mathcal{BB}_0$ is transformed according to the ways in which the adversary tampers with the ballots on $\mathcal{BB}_1$. This ensures that the result returned to the adversary corresponds to the actions taken by the adversary when constructing the ballot box. To achieve this, the extension defines an algorithm that detects the ways in which an adversary can tamper with ballots in the ballot box (e.g., the algorithm can be defined to include one or more of the following actions: delete, modify, re-order ballots). In doing so, the definition is flexible, capable of capturing different potential attack scenarios. However, this means that, before applying the definition, it is necessary to first define an algorithm, presenting an opportunity for flawed security proofs if the algorithm is not defined correctly. On the other hand, BS can be easily extended to mbbBS and does not require additional algorithms, providing a simple-to-apply definition in the malicious setting. Further, BPRIV cannot be easily extended to a setting in which the tallier is distributed and a subset of talliers can be corrupted. Indeed, in [22], del Pino *et al.* state that it is difficult to adapt BPRIV to this setting because corrupted talliers must participate in the tallying stage of the election for *both* ballot boxes, yet BPRIV only ever returns the result for the ballot box corresponding to the 'real' election. To overcome this, like our definition dtBS, del Pino *et al.* also rely on a balancing condition, effectively departing from BPRIV's approach.

5.2 Tallying the Viewed Election

Recall that the second type of approach, introduced in [4, 5], tallies the ballot box corresponding to the bulletin board viewed by the adversary. The definitions in [4] and [5] have been adopted in subsequent definitions and extended to the malicious ballot box setting. Namely, [4] (respectively, [5]) has been adopted by [30] (respectively, [10]) and extended to the malicious ballot box setting in [19] (respectively, [11]). The definitions that follow this approach require a balancing condition. The approaches defined in [4] and [5] differ with respect to the balancing condition. We choose to follow [5] to avoid the requirement of a *partial tally assumption*, which we describe in this section, and which is required by definitions that follow [4]. As a result, the definitions presented in this paper are close in spirit to [5, 10, 11]. However, our definitions are distinct. In fact, our definitions extend to e-voting schemes with voter registration, and avoid an incompatibility issue found with [10] and outlined below. We now discuss some of the benefits and limitations of this approach and, in particular, we elaborate on the features that set BS apart from other definitions that follow this approach.

Restricted Result Functions. The balancing condition required in this style of definition restricts the class of result functions that can be captured, a criticism that does not apply to BPRIV. Some definitions [4, 19, 30] define the balancing condition such that the output of the result function applied to each of the two multisets is equal, i.e., $f(V_0) = f(V_1)$. It is well-known that this approach requires a *partial tally assumption* [7, 19]. That is to say, where a set of votes can be written as $V = V' \cup V''$, the partial tally assumption states that $f(V) = f(V') * f(V'')$.

We avoid the partial tally assumption by following the approach of [5, 10, 11], and require that two sets, when viewed as multisets, are equal (we additionally extend this to e-voting schemes with a registration phase). However, this approach does still restrict the class of result functions. In fact, as we discuss in Section 3.3, our definitions, and those in [5, 10, 11], do not capture result functions that allow different vote assignments that lead to the same result. Despite this, we note that common result functions, such as plurality voting, are within the scope of our definitions.

Extendibility. Unlike BPRIV, it has been shown that definitions in this category can be easily extended to the malicious setting. In particular, Bernhard and Smyth [11] and Cortier and Lallemand [19] extend this approach to the malicious ballot box setting in an intuitive way. Our definitions mbbBS and dtBS also demonstrate how this approach can be extended to the malicious ballot box and distributed tallier settings respectively.

On the other hand, extending to model e-voting schemes with a registration phase is not as well understood. Definitions in this category that model voter registration either allow static corruption of voters only [19], or allow adaptive corruption but only allow the adversary to submit a single left- and right-hand vote on behalf of each honest voter [30]. In this paper, we show that it is possible to model adaptive corruption of voters *and* allow the adversary to submit an arbitrary number of votes on behalf of each voter, capturing e-voting schemes with revoting policies. Thus, BS models an attack strategy that has not yet captured been captured by any previous definition.

Compatibility with Verifiability. Though in this paper we focus on *privacy* for e-voting, it is desirable that a ballot secrecy definition is compatible with *verifiability* [18]. In [7] it was discovered that $\mathsf{IND}-\mathsf{SEC}$ [10], a ballot secrecy definition that relies on a balancing condition that is very similar to ours, is *not* compatible with verifiability. $\mathsf{IND}-\mathsf{SEC}$, like BS, requires that the multisets of left- and right-hand votes submitted on behalf of honest voters are equal. However, if these two multisets are not equal, $\mathsf{IND}-\mathsf{SEC}$ returns the result and accompanying tallying proof computed over the ballot box corresponding to the $\mathsf{IND}-\mathsf{SEC}$ experiment where $\beta = 0$. In [7], Cortier *et al.* prove that an e-voting scheme cannot simultaneously satisfy $\mathsf{IND}-\mathsf{SEC}$ *and* tally uniqueness, a minimal property required to ensure verifiability of an e-voting scheme, as a result of the actions performed when the multisets are not equal. We refer the reader to [7] for full details of this result. A fix to $\mathsf{IND}-\mathsf{SEC}$ was put forth in [33] to overcome this weakness, proposing that, if the multisets are not equal, the $\mathsf{IND}-\mathsf{SEC}$ experiment still returns the result computed over $\mathcal{BB}$ for $\beta = 0$, but not the tallying proof. However, a definition that does not return a tallying proof does not capture verifiable voting schemes. Instead, our definitions avoid the verifiability compatibility issue of both versions of $\mathsf{IND}-\mathsf{SEC}$ [10, 33] by restricting the adversary and *requiring* that the two multisets are equal, i.e., the experiment returns 0 otherwise.

5.3 Summarising Our Contributions

Game-based definitions of ballot secrecy in the honest model are well-studied. BPRIV, in particular, has received a lot of attention and is often regarded as the *de facto* definition of ballot secrecy. In contrast, the ballot secrecy definitions introduced in this paper follow the approach of [5]. Consequently, our definitions are not affected by the limitations of BPRIV, namely, the need for additional properties in order to prove security of an e-voting scheme, and the difficulties of extendibility. In fact, our definitions inherit the benefits of the Benaloh approach, that is, our definitions are intuitive, based on long-established techniques for indistinguishability experiments, and are well-suited to extensions into the malicious setting. Moreover, our definitions differ from existing definitions that also follow this approach. In particular, our definitions model e-voting schemes with a registration phase and, in comparison to existing definitions, BS captures *adaptive* voter corruption in an e-voting scheme with a registration phase, whilst also modelling revoting policies. Moreover, though we do restrict the set of result functions that can be considered, we do not require a partial tally assumption. We believe that, to model realistic attack scenarios, the way forward is ballot secrecy definitions that model corrupted election officials. Our definitions mbbBS and dtBS model such attack scenarios and provide a spring-board for future research in this direction. Finally, we comment that, in light of current restrictions on movement caused by global pandemics, and the potential that democratic processes could move on-line as a result, we believe it is an apt time to revisit existing approaches and explore new definitions of ballot secrecy.

References

1. Ben Adida. Helios: Web-based open-audit voting. In *USENIX'08*, pages 335–348, 2008.
2. Belenios voting system. `https://www.belenios.org/index.html`.
3. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO'98*, pages 26–45, 1998.
4. Josh Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, 1987.
5. Josh Benaloh and Moti Yung. Distributing the power of a government to enhance the privacy of votes. In *PODC'86*, pages 52–62, 1986.
6. David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. A comprehensive analysis of game-based ballot privacy definitions. ePrint Report 2015/255, 2015.
7. 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*, pages 499–516, 2015.
8. David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting helios for provable ballot privacy. In *ESORICS'11*, pages 335–354, 2011,
9. 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*, pages 626–643, 2012.

10. David Bernhard and Ben Smyth. Ballot privacy and ballot independence coincide. In *ESORICS'13*, pages 463–480, 2013.
11. David Bernhard and Ben Smyth. Ballot secrecy with malicious bulletin boards. ePrint Report 2014/822, 2014.
12. Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. Beleniosrf: A non-interactive receipt-free electronic voting scheme. In *CCS'16*, pages 1614–1625, 2016.
13. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *CRYPTO'06*, pages 78–96, 2006.
14. Civitas voting system. `www.cs.cornell.edu/projects/civitas/`.
15. Michael R Clarkson, Stephen Chong, and Andrew C Myers. Civitas: toward a secure voting system. In *S&P'08*, pages 354–368. IEEE, 2008.
16. Véronique Cortier, Constantin Catalin Dragan, Francois Dupressoir, and Bogdan Warinschi. Machine-checked proofs for electronic voting: Privacy and verifiability for belenios. In *CSF'18*, pages 298–312, 2018.
17. Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election verifiability for helios under weaker trust assumptions. In *ESORICS'14*, pages 327–344, 2014.
18. Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. Sok: Verifiability notions for e-voting protocols. In *S&P'16*, pages 779–798, 2016.
19. Véronique Cortier and Joseph Lallemand. Voting: You cant have privacy without individual verifiability. In *CCS'18*, pages 53–66, 2018.
20. Veronique Cortier, Joseph Lallemand, and Bogdan Warinschi. Fifty shades of ballot privacy: Privacy against a malicious board. ePrint Report 2020/127, 2020.
21. Véronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. In *CSF'11*, pages 297–311, 2011.
22. Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical quantum-safe voting from lattices. In *CCS'17*, pages 1565–1581, 2017.
23. Stéphanie Delaune, Steve Kremer, and Mark Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *CSFW'06*, pages 12–42, 2006.
24. Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *CRYPTO' 89*, pages 307–315, 1990.
25. i-voting. `e-estonia.com/solutions/e-governance/i-voting/`.
26. Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography*, pages 90–104, 2001.
27. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
28. Helios voting system. `heliosvoting.org/`.
29. Ari Juels, Dario Catalano, and Markus Jakobsson. *Coercion-Resistant Electronic Elections*, pages 37–63. 2010.
30. Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *EUROCRYPT'15*, pages 468–498, 2015.
31. Olivier Pereira. Internet voting with helios. *Real-World Electronic Voting*, pages 277 – 308, 2016.
32. Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *EUROCRYPT'98*, pages 1–16, 1998.
33. Ben Smyth and David Bernhard. Ballot secrecy and ballot independence: Definitions and relations. ePrint Report 2013/235, 2013.

A Building Blocks for our Constructions

A.1 Public-Key Encryption

Definition 6. *(PKE scheme) A* public key encryption scheme *Π is a tuple of PPT algorithms* $(\Pi.\mathsf{Gen}, \Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ *such that*

$\Pi.\mathsf{Gen}(1^\lambda)$ On input security parameter 1^λ, algorithm $\Pi.\mathsf{Gen}$ outputs a key pair (pk_Π, sk_Π).

$\Pi.\mathsf{Enc}(pk_\Pi, m)$ On input public key pk_Π and message m, algorithm $\Pi.\mathsf{Enc}$ outputs a ciphertext c.

$\Pi.\mathsf{Dec}(sk_\Pi, c)$ On input private key sk_Π and ciphertext c, algorithm $\Pi.\mathsf{Dec}$ outputs a message m.

Definition 7 (NM-CPA). *A public key encryption scheme Π satisfies* NM-CPA *if, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{NM\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{NM\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda)$ *is the experiment defined in Figure 4.*

A.2 Threshold Public Key Encryption

Definition 8. *(Threshold PKE scheme) A* (t, n)-threshold public key encryption *scheme Φ is a tuple of PPT algorithms* $(\Phi.\mathsf{Gen}, \Phi.\mathsf{Enc}, \Phi.\mathsf{Dec}, \Phi.\mathsf{Combine})$ *such that*

$\Phi.\mathsf{Gen}(1^\lambda, t, n)$ On input security parameter 1^λ, threshold t and n, algorithm $\Phi.\mathsf{Gen}$ outputs a public key pk_Φ and n private keys, $sk_{\Phi_1}, \ldots, sk_{\Phi_n}$.

$\Phi.\mathsf{Enc}(pk_\Phi, m)$ On input public key pk_Φ and message m, algorithm $\Phi.\mathsf{Enc}$ outputs a ciphertext c.

$\Phi.\mathsf{Dec}(pk_\Phi, i, sk_{\Phi_i}, c)$ On input public key pk_Φ, an index $1 \leq i \leq n$, private key sk_{Φ_i} and ciphertext c, algorithm $\Phi.\mathsf{Dec}$ outputs a decryption share c_i.

$\Phi.\mathsf{Combine}(pk_\Phi, c, c_1, \ldots, c_t)$ On input public key pk_Φ, ciphertext c and t decryption shares $c_1, \ldots, c_t$, algorithm $\Phi.\mathsf{Combine}$ outputs a message m.

Definition 9 (t-NM-CPA). *A (t, n)-threshold public key encryption scheme Φ satisfies t-*NM-CPA *if, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{t\text{-}\mathsf{NM\text{-}CPA}}_{\Phi,\mathcal{A}}(\lambda, t, n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{t\text{-}\mathsf{NM\text{-}CPA}}_{\Phi,\mathcal{A}}(\lambda, t, n)$ *is the experiment defined in Figure 4.*

```
$\mathsf{Exp}^{\mathsf{NM\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda)$
$(pk_\Pi, sk_\Pi) \leftarrow \Pi.\mathsf{Gen}(1^\lambda)$
$\beta \leftarrow \{0,1\}$
$\mathcal{Q}\mathsf{Encrypt} \leftarrow \emptyset$
$(\boldsymbol{c}, st) \leftarrow \mathcal{A}_1^{\mathcal{O}\mathsf{Encrypt}}(pk_\Pi)$
for $i = 1, \dots |\boldsymbol{c}|$
  $m_i \leftarrow \Pi.\mathsf{Dec}(sk_\Pi, \boldsymbol{c}[i])$
$\boldsymbol{m} \leftarrow (m_1, \dots, m_{|\boldsymbol{c}|})$
$\beta' \leftarrow \mathcal{A}_2(\boldsymbol{m}, st_2)$
if $\beta' = \beta \wedge \forall c \in \boldsymbol{c}.c \notin \mathcal{Q}\mathsf{Encrypt}$
  return 1
else
  return 0

$\mathcal{O}\mathsf{Encrypt}_{pk_\Pi}(m_0, m_1)$
$c \leftarrow \Pi.\mathsf{Enc}(pk_\Pi, m_\beta)$
$\mathcal{Q}\mathsf{Encrypt} \leftarrow \mathcal{Q}\mathsf{Encrypt} \cup \{c\}$
return $c$
```

```
$\mathsf{Exp}^{\mathsf{NM\text{-}CPA}}_{\Phi,\mathcal{A}}(\lambda, t, n)$
$(k_1, \dots, k_{t-1}, st_1) \leftarrow \mathcal{A}_1()$
$(pk_\Phi, sk_{\Phi_1}, \dots, sk_{\Phi_n}) \leftarrow \Phi.\mathsf{Gen}(1^\lambda, t, n)$
$\beta \leftarrow \{0,1\}$
$\mathcal{Q}\mathsf{Encrypt} \leftarrow \emptyset$
$(\boldsymbol{c}, l_1, \dots, l_t, st_2) \leftarrow \mathcal{A}_2^{\mathcal{O}\mathsf{Encrypt}}(pk_\Phi, sk_{\Phi_{k_1}}, \dots, sk_{\Phi_{k_{t-1}}}, st_1)$
for $i = 1, \dots |\boldsymbol{c}|$
  for $j = 1, \dots, t$
    $c_{i,j} \leftarrow \Phi.\mathsf{Dec}(pk_\Phi, l_j, sk_{\Phi_{l_j}}, \boldsymbol{c}[i])$
  $m_i \leftarrow \Phi.\mathsf{Combine}(pk_\Phi, \boldsymbol{c}[i], c_{i,1}, \dots, c_{i.t})$
$\boldsymbol{m} \leftarrow (m_1, \dots, m_{|\boldsymbol{c}|})$
$\beta' \leftarrow \mathcal{A}_3(\boldsymbol{m}, st_2)$
if $\beta' = \beta \wedge \forall c \in \boldsymbol{c}.c \notin \mathcal{Q}\mathsf{Encrypt}$
  return 1
else
  return 0

$\mathcal{O}\mathsf{Encrypt}_{pk_\Phi}(m_0, m_1)$
$c \leftarrow \Phi.\mathsf{Enc}(pk_\Phi, m_\beta)$
$\mathcal{Q}\mathsf{Encrypt} \leftarrow \mathcal{Q}\mathsf{Encrypt} \cup \{c\}$
return $c$
```

Fig. 4: The NM-CPA experiment $\mathsf{Exp}^{\mathsf{NM\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda)$ for public key encryption scheme Π and the t-NM-CPA experiment $\mathsf{Exp}^{t\text{-}\mathsf{NM\text{-}CPA}}_{\Phi,\mathcal{A}}(\lambda)$ for (t, n)-threshold public key encryption scheme Φ.

A.3 Signature of Knowledge

Definition 10 (Signature of knowledge). *A* signature of knowledge SOK *is a tuple of algorithms* ($\mathsf{SoK.Setup}, \mathsf{SimSetup}, \mathsf{SoK.Sign}, \mathsf{SimSign}, \mathsf{SoK.Verify}$) *relative to a relation* $\mathcal{R}$ *such that*

- $\mathsf{SoK.Setup}(1^\lambda)$: on input security parameter 1^λ, algorithm $\mathsf{SoK.Setup}$ outputs public parameters pp.
- $\mathsf{SimSetup}(1^\lambda)$: on input security parameter 1^λ, algorithm $\mathsf{SimSetup}$ outputs public parameters pp and trapdoor τ.
- $\mathsf{SoK.Sign}(pp, s, w, m)$: on input pp, statement s, witness w and message m, algorithm $\mathsf{SoK.Sign}$ outputs a signature σ if $(s, w) \in \mathcal{R}$.
- $\mathsf{SimSign}(pp, s, \tau, m)$: on input pp, s, τ and m, algorithm $\mathsf{SimSign}$ outputs a signature σ.
- $\mathsf{SoK.Verify}(pp, s, m, \sigma)$: on input pp, s, m and σ, algorithm $\mathsf{SoK.Verify}$ outputs 1, if the signature verifies and 0 otherwise.

Definition 11 (Extractability). *Let* $\mathsf{Extract}$ *be a PPT algorithm such that*

– $\mathsf{Extract}(pp, \tau, s, m, \sigma)$*: on input public parameters pp, trapdoor* τ*, statement s, message m and signature* σ*, algorithm* Extract *outputs a witness w.*

Then signature of knowledge SOK *satisfies* extractability *if, for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{Ext}}_{\mathsf{SOK},\mathcal{A},\mathsf{Extract}}(\lambda)\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{Ext}}_{\mathsf{SOK},\mathcal{A},\mathsf{Extract}}(\lambda)$ *is the experiment defined in Figure 5.*

```
$\mathsf{Exp}^{\mathsf{Ext}}_{\mathsf{SOK},\mathcal{A},\mathsf{Extract}}(\lambda)$
$\mathcal{Q} \leftarrow \emptyset$
$(pp, \tau) \leftarrow \mathsf{SimSetup}(1^\lambda)$
$(s, m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(pp)$
$w \leftarrow \mathsf{Extract}(pp, s, m, \sigma)$
**if** $(s, w) \notin \mathcal{R} \wedge (s, m, \sigma) \notin \mathcal{Q} \wedge \mathsf{SoK.Verify}(pp, s, m, \sigma) = 1$
  **return** $1$
**else**
  **return** $0$

$\mathcal{O}_{pp,\tau}(s, w, m)$
$\sigma \leftarrow \mathsf{SimSign}(pp, s, \tau, m)$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup (s, m, \sigma)$
**return** $\sigma$
```

Fig. 5: The extractability experiment $\mathsf{Exp}^{\mathsf{Ext}}_{\mathsf{SOK},\mathcal{A},\mathsf{Extract}}(\lambda)$ for signature of knowledge SOK.

B Ballot Secrecy of our Constructions

B.1 Proof of Theorem 1

Let $\mathcal{A}$ be an adversary in the experiment $\mathsf{Exp}^{\mathsf{BS}}_{\Gamma_{\mathsf{mini}},\mathcal{A}}(\lambda)$ where Γ_{mini} is the construction in Figure 2. Assume that $\mathcal{A}$ can output a bit β' such that $\beta' = \beta$ and $\mathcal{A}$ queries $\mathcal{O}\mathsf{vote}$ and $\mathcal{O}\mathsf{cast}$ such that $V_0' = V_1'$. Then $\mathcal{A}$ succeeds in experiment $\mathsf{Exp}^{\mathsf{BS}}_{\Gamma_{\mathsf{mini}},\mathcal{A}}(\lambda)$ with probability non-negligibly greater than $\frac{1}{2}$. We note that, throughout the experiment, $\mathcal{A}$ can only gain information about β through access to the bulletin board $\mathcal{BB}$ and the election result r.

Let Forge denote the event that $\mathcal{A}$ submits a valid ballot $b = (pk_{id}, c, \sigma)$ to $\mathcal{O}\mathsf{cast}$ where $(\cdot, pk_{id}, \cdot) \in \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$. We write that b is valid if $\mathsf{Valid}(b, \mathcal{BB}, pk, \mathcal{L}) = 1$ which requires, in particular, that $\mathsf{SoK.Verify}(pp, (c, pk_{\Pi}, pk_{id}), c, \sigma) = 1$. Moreover, let $\mathsf{Success}$ denote the event that experiment $\mathsf{Exp}^{\mathsf{BS}}_{\Gamma_{\mathsf{mini}},\mathcal{A}}(\lambda)$ returns 1. Note that,

$$\Pr\left[\mathsf{Exp}^{\mathsf{BS}}_{\Gamma_{\mathsf{mini}},\mathcal{A}}(\lambda) = 1\right] \leq \Pr[\mathsf{Success} \wedge \mathsf{Forge}] + \Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right]$$
$$\leq \Pr[\mathsf{Forge}] + \Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right].$$

We show that $\Pr[\mathsf{Forge}] \leq \mathsf{negl}(\lambda)$ and $\Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. The result of the Theorem then follows.

First, we show that $\Pr[\mathsf{Forge}] \leq \mathsf{negl}(\lambda)$. In fact, we show that, if $\mathcal{A}$ can submit a valid ballot to $\mathcal{O}\mathsf{corrupt}$, then $\mathcal{A}$ can be used to construct an adversary $\mathcal{B}$ in the extractability experiment $\mathsf{Exp}^{\mathsf{Ext}}_{\mathsf{SOK},\mathcal{B},\mathsf{Extract}}(\lambda)$, where $\mathcal{B}$ plays the role of the challenger in the BS experiment and $\mathcal{C}$ is the challenger in the extractability experiment. In detail, we construct adversary $\mathcal{B}$ as follows.

1. $\mathcal{B}$ obtains public parameters pp for signature of knowledge SOK from $\mathcal{C}$ and runs $(pk_{\Pi}, sk_{\Pi}) \leftarrow \Pi.\mathsf{Gen}(1^{\lambda})$, and performs the setup of Γ_{mini}, providing $\mathcal{A}$ with pk. $\mathcal{B}$ additionally performs the initialisation steps of the BS experiment and selects a bit $\beta \leftarrow \{0, 1\}$.
2. $\mathcal{B}$ answers queries to $\mathcal{O}\mathsf{reg}$, $\mathcal{O}\mathsf{corrupt}$ and $\mathcal{O}\mathsf{board}$ as described in the BS experiment. Moreover, $\mathcal{B}$ computes the election result as described in algorithm Tally.
3. For queries to $\mathcal{O}\mathsf{vote}(pk_{id}, v_0, v_1)$, $\mathcal{B}$ computes $c \leftarrow \Pi.\mathsf{Enc}(pk_{\Pi}, m_{\beta}; r)$ and queries $\mathcal{O}_{pp,\tau}((c, pk_{\Pi}, pk_{id}), (sk_{id}, r), c)$ in the extractability experiment, receiving a signature of knowledge σ. $\mathcal{B}$ constructs ballot $b \leftarrow (pk_{id}, c, \sigma)$ and appends the ballot to $\mathcal{BB}$.
4. $\mathcal{B}$ answers queries to $\mathcal{O}\mathsf{cast}$ as described in the BS experiment. By assumption that event Forge occurs, $\mathsf{SoK.Verify}$ returns 1 for at least one tuple (pk_{id}, b) queried to $\mathcal{O}\mathsf{cast}$ such that $pk_{id} \in \mathcal{Q}\mathsf{reg} \setminus \mathcal{Q}\mathsf{corrupt}$. We denote this ballot as $(pk^{*}_{id}, c^{*}, \sigma^{*})$. Then, $\mathcal{B}$ output $((c^{*}, pk_{\Pi}, pk^{*}_{id}), c^{*}, \sigma^{*})$ in the extractability experiment.

$\mathcal{B}$ perfectly simulates the role of the challenger in the BS experiment to $\mathcal{A}$. In fact, $\mathcal{B}$ trivially simulates oracles $\mathcal{O}\mathsf{reg}$, $\mathcal{O}\mathsf{corrupt}$ and $\mathcal{O}\mathsf{board}$, and trivially computes the result of the election and returns r to $\mathcal{A}$. Moreover, when $\mathcal{A}$ queries $\mathcal{O}\mathsf{vote}$, $\mathcal{B}$ returns a ciphertext consisting of an encryption c that is identical to the encryption viewed by $\mathcal{A}$ in the BS experiment and obtains a signature of knowledge from $\mathcal{O}$ in the extractability experiment that is identical to the signature viewed by $\mathcal{A}$ in the BS experiment. Therefore, $\mathcal{B}$ perfectly simulates $\mathcal{O}\mathsf{vote}$ to $\mathcal{A}$. Furthermore, $\mathcal{B}$ perfectly simulates $\mathcal{O}\mathsf{cast}$ to $\mathcal{A}$ and, if event Forge occurs, $\mathcal{A}$ successfully creates a valid signature of knowledge without witness (sk_{id}, r), and, therefore, $\mathcal{B}$ can output this signature in the extractability experiment and succeeds. By assumption, the signature of knowledge SOK satisfies extractability, and hence, we conclude that $\Pr[\mathsf{Forge}] \leq \mathsf{negl}(\lambda)$.

We now show that $\Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. That is, we show that, if $\mathcal{A}$ succeeds in the BS experiment without event Forge occurring, we can use $\mathcal{A}$ to construct an adversary $\mathcal{B}'$ in the $\mathsf{NM\text{-}CPA}$ experiment $\mathsf{Exp}^{\mathsf{NM\text{-}CPA}}_{\Pi,\mathcal{B}'}(\lambda)$, where $\mathcal{B}'$ plays the role of the challenger in the BS experiment and $\mathcal{C}$ is the challenger in the $\mathsf{NM\text{-}CPA}$ experiment against scheme Π. In detail, we construct adversary $\mathcal{B}'$ as follows.

1. $\mathcal{B}'$ obtains a public key pk_Π for public key encryption scheme Π from $\mathcal{C}$, runs $(pp, \tau) \leftarrow \mathsf{SimSetup}(1^\lambda)$, and performs the setup of Γ_{mini}, providing $\mathcal{A}$ with pk. $\mathcal{B}'$ additionally performs the initialisation steps of the BS experiment.
2. $\mathcal{B}'$ answers queries to oracles $\mathcal{O}\mathsf{reg}$, $\mathcal{O}\mathsf{corrupt}$, $\mathcal{O}\mathsf{cast}$ and $\mathcal{O}\mathsf{board}$ as described in the BS experiment.
3. For queries $\mathcal{O}\mathsf{vote}(pk_{id}, v_0, v_1)$, $\mathcal{B}'$ queries $(m_0 = v_0, m_1 = v_1)$ to oracle $\mathcal{O}\mathsf{Encrypt}$ in the $\mathsf{NM\text{-}CPA}$ experiment and receives a ciphertext c of m_β. $\mathcal{B}'$ then computes $\sigma \leftarrow \mathsf{SimSign}(pp, (c, pk_\Pi, pk_{id}), \tau, c)$ and appends ballot $b = (pk_{id}, c, \sigma)$ to $\mathcal{BB}$.
4. $\mathcal{B}'$ computes the election result. Throughout, $\mathcal{B}'$ keeps track of tuples (pk_{id}, b) queried by $\mathcal{A}$ to $\mathcal{O}\mathsf{cast}$, such that the query results in a ballot that will be included in the result. We denote by B the set of all tuples of the form (pk_{id}, b). $\mathcal{B}'$ constructs a vector $\boldsymbol{c}$ that consists of the ciphertext element of each ballot in B and submits $\boldsymbol{c}$ to $\mathcal{C}$. $\mathcal{C}$ returns a vector of plaintexts $\boldsymbol{m}$ (i.e., the plaintext votes encoded in ballots submitted to $\mathcal{O}\mathsf{cast}$) to $\mathcal{B}'$. For each tuple $(pk_{id}, b) \in B$, $\mathcal{B}'$ replaces ballot b with the corresponding plaintext included in vector $\boldsymbol{m}$. $\mathcal{B}'$ then computes the result r by computing the result function $f(V_0 \cup B)$.
5. $\mathcal{B}'$ returns the bit β' output by $\mathcal{A}$.

We show that $\mathcal{B}'$ perfectly simulates the role of the challenger in the BS experiment to $\mathcal{A}$. Trivially, $\mathcal{B}'$ simulates oracles $\mathcal{O}\mathsf{reg}$, $\mathcal{O}\mathsf{corrupt}$, $\mathcal{O}\mathsf{cast}$ and $\mathcal{O}\mathsf{board}$ to $\mathcal{A}$. Additionally, $\mathcal{B}'$ answers queries to $\mathcal{O}\mathsf{vote}$ by obtaining a ciphertext from $\mathcal{O}\mathsf{Encrypt}$ in the $\mathsf{NM\text{-}CPA}$ experiment that is identical to the encryption viewed by $\mathcal{A}$ in the BS experiment and constructs a signature of knowledge that is identical to that viewed by $\mathcal{A}$. Consequently, the ballot obtained by $\mathcal{B}'$, and subsequently appended to the ballot box, is identical to the ballot computed by $\mathcal{O}\mathsf{vote}$. Therefore, $\mathcal{B}'$ perfectly simulates $\mathcal{O}\mathsf{vote}$. Finally, $\mathcal{B}'$ computes the result function for set V_0 (and B). By the assumption that event Forge does not occur, B cannot contain ballots meaningfully related to the votes of honest voters. Moreover, by assumption that $\mathcal{A}$ succeeds in the BS experiment, $V_0 = V_1$ and, as such, $\mathcal{B}'$ perfectly simulates algorithm Tally to $\mathcal{A}$. Moreover, we have that β' output by $\mathcal{B}'$ is equal to the bit β chosen by $\mathcal{C}$ in the $\mathsf{NM\text{-}CPA}$ experiment. In particular, if $\mathcal{A}$ correctly guesses β' in the BS experiment, $\mathcal{A}$ correctly determines whether $\mathcal{BB}$ contains ballots corresponding to left- or right-hand votes submitted via $\mathcal{O}\mathsf{vote}$. As these ballots are created by calling $\mathcal{O}\mathsf{Encrypt}$ in the $\mathsf{NM\text{-}CPA}$ experiment where the bit β is chosen by the $\mathsf{NM\text{-}CPA}$ challenger, $\mathcal{B}'$ succeeds in the $\mathsf{NM\text{-}CPA}$ experiment. However, by assumption, Π satisfies $\mathsf{NM\text{-}CPA}$ security and we conclude that $\Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. □

B.2 Proof of Theorem 2

The details of this result follow largely from the proof of Theorem 1. We let $\mathcal{A}$ be an adversary in the experiment $\mathsf{Exp}^{\mathsf{mbbBS}}_{\Gamma_{\mathsf{mini}},\mathcal{A}}(\lambda)$ where Γ_{mini} is the construction in Figure 2. We assume that $\mathcal{A}$ succeeds in experiment $\mathsf{Exp}^{\mathsf{mbbBS}}_{\Gamma_{\mathsf{mini}},\mathcal{A}}(\lambda)$ with probability non-negligibly greater than $\frac{1}{2}$, that is, $\mathcal{A}$ outputs a bit β' such that $\beta' = \beta$ and constructs ballot box $\mathcal{BB}$ such that $V_0 = V_1$. Throughout the experiment, $\mathcal{A}$ obtains information about β through ballots output by $\mathcal{O}\mathsf{vote}$ and the election result r.

Let Forge denote the event that $\mathcal{A}$ posts a valid ballot $b = (pk_{id}, c, \sigma)$ to $\mathcal{BB}$ where $pk_{id} \in \mathcal{L} \setminus \mathsf{corr}\mathcal{L}$ and b is not the output of $\mathcal{O}\mathsf{vote}$. We write that b is valid if $\mathsf{SoK.Verify}(pp, (c, pk_{\Pi}, pk_{id}), \sigma, c) = 1$. Moreover, let $\mathsf{Success}$ denote the event that experiment $\mathsf{Exp}^{\mathsf{mbbBS}}_{\Gamma_{\mathsf{mini}},\mathcal{A}}(\lambda)$ returns 1. As before,

$$\Pr\left[\mathsf{Exp}^{\mathsf{mbbBS}}_{\Gamma_{\mathsf{mini}},\mathcal{A}}(\lambda) = 1\right] \leq \Pr\left[\mathsf{Forge}\right] + \Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right].$$

We show that $\Pr[\mathsf{Forge}] \leq \mathsf{negl}(\lambda)$ and $\Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. The result of the Theorem then follows.

First, we show that $\Pr[\mathsf{Forge}] \leq \mathsf{negl}(\lambda)$. In fact, we show that, if $\mathcal{A}$ can post a valid ballot to $\mathcal{BB}$ for an honest voter (without calling $\mathcal{O}\mathsf{vote}$), then $\mathcal{A}$ can be used to construct an adversary $\mathcal{B}$ in the extractability experiment $\mathsf{Exp}^{\mathsf{Ext}}_{\mathsf{SOK},\mathcal{B},\mathsf{Extract}}(\lambda)$, where $\mathcal{B}$ plays the role of the challenger in the mbbBS experiment and $\mathcal{C}$ is the challenger in the extractability experiment. The detailed construction of $\mathcal{B}$ is very similar to the adversary $\mathcal{B}$ described in the proof of Theorem 1, and we refer the reader to this proof for full details. We describe the following changes to the adversary $\mathcal{B}$:

1. In step 2, $\mathcal{B}$ does not answer queries to oracles $\mathcal{O}\mathsf{reg}$, $\mathcal{O}\mathsf{corrupt}$ or $\mathcal{O}\mathsf{board}$ as $\mathcal{A}$ does not have access to these oracles in the mbbBS experiment.
2. In step 4, by assumption that event Forge occurs, $\mathsf{SoK.Verify}$ returns 1 for at least one ballot, which we denote $b^* = (pk^*_{id}, c^*, \sigma^*)$, that appears on $\mathcal{BB}$ such that $pk_{id} \in \mathcal{L} \setminus \mathsf{corr}\mathcal{L}$ and b is not the output of $\mathcal{O}\mathsf{vote}$. Then, $\mathcal{B}$ output $((c^*, pk_{\Pi}, pk^*_{id}), c^*, \sigma^*)$ in the extractability experiment.

$\mathcal{B}$ perfectly simulates the role of the challenger in the mbbBS experiment to $\mathcal{A}$. As in the proof of Theorem 1, $\mathcal{B}$ perfectly simulates $\mathcal{O}\mathsf{vote}$ to $\mathcal{A}$. Furthermore, if event Forge occurs, $\mathcal{A}$ successfully creates a valid signature without witness (sk_{id}, r), and, therefore, $\mathcal{B}$ can output this signature in the extractability experiment and succeeds. By assumption, the signature of knowledge SOK satisfies extractability, and hence, we conclude that $\Pr[\mathsf{Forge}] \leq \mathsf{negl}(\lambda)$.

We now show that $\Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. That is, we show that, if $\mathcal{A}$ succeeds in the $\mathsf{NM\text{-}CPA}$ experiment without event Forge occurring, we can use $\mathcal{A}$ to construct an adversary $\mathcal{B}'$ in the $\mathsf{NM\text{-}CPA}$ experiment $\mathsf{Exp}^{\mathsf{NM\text{-}CPA}}_{\Pi,\mathcal{B}'}(\lambda)$, where $\mathcal{B}'$ plays the role of the challenger in the mbbBS experiment and $\mathcal{C}$ is the challenger in the $\mathsf{NM\text{-}CPA}$ experiment. Again, the detailed construction of $\mathcal{B}'$ is very similar to the adversary $\mathcal{B}'$ described in the proof of Theorem 1, and we describe the following changes to $\mathcal{B}'$:

1. Step 2 is no longer required as $\mathcal{A}$ does not have access to oracles $\mathcal{O}\mathsf{reg}$, $\mathcal{O}\mathsf{corrupt}$ or $\mathcal{O}\mathsf{board}$ in the mbbBS experiment.
2. For queries to $\mathcal{O}\mathsf{vote}(pk_{id}, v_0, v_1)$, rather than appending ballot b to $\mathcal{BB}$, $\mathcal{B}'$ outputs b to $\mathcal{A}$.
3. To compute the election result, $\mathcal{B}'$ creates a set B that consists of tuples (pk_{id}, b) such that ballot b, submitted on behalf of voter credential pk_{id}, appears on $\mathcal{BB}$, is not an output of $\mathcal{O}\mathsf{vote}$, and will be included in the result (i.e., is the last ballot cast for voter pk_{id}). $\mathcal{B}'$ then constructs vector $\boldsymbol{c}$ from set B and proceeds to compute the election result as described in Step 4 in the proof of Theorem 1.

$\mathcal{B}'$ perfectly simulates the role of the challenger in the mbbBS experiment to $\mathcal{A}$. In particular, the ballot output by $\mathcal{B}'$ following a query to $\mathcal{O}\mathsf{vote}$ is identical to the ballot output by $\mathcal{O}\mathsf{vote}$ because $\mathcal{B}'$ obtains the ballot by querying $\mathcal{O}\mathsf{Encrypt}$ in the $\mathsf{NM\text{-}CPA}$ experiment. Moreover, as in the proof of Theorem 1, $\mathcal{B}'$ perfectly simulates algorithm Tally to $\mathcal{A}$ as we assume that $V_0 = V_1$ and, as such, the results computed for $\beta = 0$ and $\beta = 1$ are identical. $\mathcal{B}'$ outputs $\beta' = \beta$ in the $\mathsf{NM\text{-}CPA}$ experiment. That is, if $\mathcal{A}$ outputs $\beta' = \beta$ in the mbbBS experiment, $\mathcal{A}$ correctly determines whether $\mathcal{O}\mathsf{vote}$ returns a ballot corresponding to the left- or right-hand vote. As the ballot is constructed by calling $\mathcal{O}\mathsf{Encrypt}$, where β is chosen by the challenger $\mathcal{C}$, $\mathcal{B}'$ succeeds in the $\mathsf{NM\text{-}CPA}$ experiment. By assumption, Π satisfies $\mathsf{NM\text{-}CPA}$ security and we conclude that $\Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. $\square$

B.3 Proof of Theorem 3

The details of this result follow largely from the proof of Theorem 2. We focus on the changes to the proof of Theorem 2. We let $\mathcal{A}$ be an adversary in the experiment $\mathsf{Exp}^{\mathsf{dtBS}}_{\Gamma'_{\mathsf{mini}},\mathcal{A}}(\lambda, t, n)$ where Γ'_{mini} is the construction described in Section 4.2. We assume that $\mathcal{A}$ succeeds in experiment $\mathsf{Exp}^{\mathsf{dtBS}}_{\Gamma'_{\mathsf{mini}},\mathcal{A}}(\lambda, t, n)$ with probability non-negligibly greater than $\frac{1}{2}$. We define events Forge and $\mathsf{Success}$ as in the proof of Theorem 2 and we show that $\Pr[\mathsf{Forge}] \leq \mathsf{negl}(\lambda)$ and $\Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. The result of the Theorem then follows.

First, we show that $\Pr[\mathsf{Forge}] \leq \mathsf{negl}(\lambda)$. This part of the proof is identical to the proof of Theorem 2, with the following exceptions. In step 1 of the description of adversary $\mathcal{B}$, $\mathcal{B}$ runs $(pk_{\Phi}, sk_{\Phi_1}, \ldots, sk_{\Phi_n}) \leftarrow \Phi.\mathsf{Gen}(1^{\lambda}, t, n)$ and provides $\mathcal{A}$ with $t-1$ private keys of $\mathcal{A}$'s choice. Additionally, $\mathcal{B}$ computes the result using the $t-1$ private keys given to $\mathcal{A}$ plus one other private key. As in the proof of Theorem 2, $\mathcal{B}$ perfectly simulates the role of the challenger in the dtBS experiment to $\mathcal{A}$. In particular, $\mathcal{B}$ generates the keys for Φ and provides $\mathcal{A}$ with $t-1$ private keys as expected. This concludes the first part of the proof.

We now show that $\Pr\left[\mathsf{Success} \wedge \overline{\mathsf{Forge}}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. We describe the following change to adversary $\mathcal{B}'$. In step 1, $\mathcal{B}'$ requests the $t-1$ private keys from $\mathcal{C}$ that $\mathcal{A}$ requests, and $\mathcal{B}'$ returns the private keys provided by $\mathcal{C}$ to $\mathcal{A}$. Specific to the proof of this result, $\mathcal{B}'$ returns private keys to $\mathcal{A}$ that are identical to the private keys output to $\mathcal{A}$ in the dtBS experiment. Therefore, the second part of the proof holds. $\square$