A foundation for secret, verifiable elections

Ben Smyth

Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg

February 23, 2018

Abstract

Many voting systems rely on art, rather than science, to ensure that votes are freely made, with equal influence. Such systems build upon creativity and skill, rather than scientific foundations. These systems are routinely broken in ways that compromise free-choice or permit undue influence. Breaks can be avoided by proving that voting systems satisfy formal notions of voters voting freely and of detecting undue influence. This manuscript provides a detailed technical introduction to a definition of ballot secrecy by Smyth that formalises the former notion and a definition of verifiability by Smyth, Frink & Clarkson that formalises the latter. The definitions are presented in the computational model of cryptography: Ballot secrecy is expressed as the inability to distinguish between an instance of the voting system in which voters cast some votes, from another instance in which the voters cast a permutation of those votes. Verifiability decomposes into individual verifiability, which is expressed as the inability to cause a collision between ballots, and universal verifiability, which is expressed as the inability to cause an incorrect election outcome to be accepted. The definitions are complimented with simple examples that demonstrate the essence of these properties and detailed proofs are constructed to show how secrecy and verifiability can be formally proved. Finally, the Helios and Helios Mixnet voting systems are presented as case studies to provide an understanding of state-of-the-art systems that are being used for binding elections.

Keywords. Elections, privacy, provable security, verifiability, voting.

1 Introduction

An election is a decision-making procedure to choose representatives [LG84, Saa95, Gum05, AH10]. Choices should be made by voters with equal influence, and this must be ensured by voting systems, as prescribed by the United Nations [UN48], the Organisation for Security & Cooperation in Europe [OSC90], and the Organization of American States [OAS69]. Historically, "Americans [voted] with their voices – *viva voce* – or with their hands or with their feet. Yea or nay. Raise your hand. All in favor of Jones, stand on this side of the town common; if you support Smith, line up over there" [Lep08]. Thus, ensuring that only voters voted and did so with equal influence was straightforward. Indeed, the election outcome could be determined by anyone present, simply by considering at most one vote per voter and disregarding non-voters. Yet, voting systems must also ensure choices are made freely, as prescribed by the aforementioned organisations [UN48, OSC90, OAS69]. Mill eloquently argues that choices cannot be expressed freely in public: "The unfortunate voter is in the power of some opulent man; the opulent man informs him how he must vote. Conscience, virtue, moral obligation, religion, all cry to him, that he ought to consult his own judgement, and faithfully follow its dictates. The consequences of pleasing, or offending the opulent man, stare him in the face...the moral obligation is disregarded, a faithless, a prostitute, a pernicious vote is given" [Mil30].

The need for free-choice started a movement towards voting as a private act, i.e., "when numerous social constraints in which citizens are routinely and universally enmeshed – community of religious allegiances, the patronage of big men, employers or notables, parties, 'political machines' – are kept at bay," and "this idea has become the current *doxa* of democracy-builders worldwide" [BBP07]. The most widely used embodiment of this idea is the Australian system, which demands that votes be marked on uniform ballots in polling booths and deposited into ballot boxes. Uniformity is intended to enable free-choice during distribution, collection and tallying of ballots, and the isolation of polling booths is intended to facilitate free-choice whilst marking.[1] Moreover, the Australian system can assure that only voters vote and do so with equal influence. Indeed, observers can check that ballots are only distributed to voters and at most one ballot is deposited by each voter. Furthermore, observers can check that spoiled ballots are discarded and that votes expressed in the remaining ballots correspond to the election outcome. Albeit, assurance is limited by an observer's ability to monitor [Bjo04, Kel12, Nor15] and the ability to transfer that assurance is limited to the observer's "good word or sworn testimony" [NA03].

Many electronic voting systems – including systems that have been used in large-scale, binding elections – rely on art, rather than science, to ensure that votes are freely made, with equal influence. Such systems build upon creativity and skill, rather than scientific foundations. These systems are routinely broken in ways that violate free-choice, e.g., [KSRW04, GH07, Bow07, WWH+10, WWIH12, SFD+14], or permit undue influence, e.g., [KSRW04, UK07, Bow07, Ger09, JS12]. Breaks can be avoided by proving that systems satisfy formal notions of voters voting freely and of detecting undue influence. Smyth, Frink & Clarkson [SFC17] propose a definition of *verifiability* that formalises the latter notion, and Smyth [Smy18a] proposes a definition of *ballot secrecy* that for-

[1] Earlier systems merely required ballots to be marked in polling booths and deposited into ballot boxes, which permitted non-uniform ballots, including ballots of different colours and sizes, that could be easily identified as party tickets [Bre06].

malises the former. This manuscript provides a detailed technical introduction to those definitions. The definitions are presented in the computational model of cryptography using games, whereby a benign challenger, a malicious adversary and a voting system engage in a series of interactions which task the adversary to break security.

Verifiability. Verifiability requires voting systems to produce evidence of correct operation. Such evidence can be checked to determine whether the selection of representatives has been unduly influenced. Hence, breaks permitting undue influence can be eradicated, moreover, the aforementioned limitations of observers can be overcome. Indeed, universal verifiability formalises a notion of checking whether votes expressed in ballots correspond to the election outcome. Thus, undue influence can be detected, without monitoring.

- Universal verifiability. Anyone can check whether an outcome corresponds to votes expressed in collected ballots.

Smyth, Frink & Clarkson capture universal verifiability as a game that tasks the adversary to falsify evidence that causes checks to succeed when the outcome does not correspond to the votes expressed in collected ballots, or that cause checks to fail when the outcome does correspond to the votes expressed. Hence, winning signifies the existence of a scenario in which a spurious outcome will be accepted or a legitimate outcome rejected, i.e., a security breach. By comparison, when no winning adversary exists, anyone can determine whether the election outcome is correct.

Voting systems must ensure outcomes include only voters' votes, which can be achieved by collecting only voters' ballots. Moreover, only one ballot should be collected from each voter, to ensure equal influence. Alternatively, voting systems may satisfy a stronger notion universal verifiability (whereby, anyone can check whether an outcome corresponds to votes expressed in collected ballots that are authorised, except votes cast by the same voter) and a notion of *unforgeability*, i.e., only voters can construct authorised ballots.[2] Unforgeability seems to require expensive infrastructures for voter credentials and some systems – including Helios and Helios Mixnet – forgo unforgeability in favour of cheaper, non-verifiable ballot authentication mechanisms. We focus on voting systems with such non-verifiable mechanisms.

Merely casting a ballot is insufficient to ensure it is collected, because an adversary may discard or modify ballots. Hence, evidence produced by voting systems should include the set of collected ballots and voters should check that their ballot has not been omitted. Yet, this is insufficient, because two ballots may collide and an adversary may discard just one ballot. Thus, voters must uniquely identify *their* ballot. Individual verifiability formalises a notion of voters convincing themselves that their ballot is amongst those collected, assuming ballots are constructed in the prescribed manner.

[2] I have previously used 'eligibility verifiability' as a synonym for 'unforgeability' [SFC17], despite the absence of any checks in the corresponding security definition. In hindsight, unforgeability is a better name.

- Individual verifiability. A voter can check whether their ballot is collected.

Smyth, Frink & Clarkson capture individual verifiability as a game that tasks the adversary to cause a collision between ballots constructed in the manner prescribed by the voting system. The game proceeds as follows: First, the adversary provides any inputs necessary to construct a ballot. Secondly, the challenger constructs a ballot using those inputs, in the manner prescribed by the voting system. Finally, the adversary and the challenger repeat the process to construct a second ballot. The adversary wins if the two independently constructed ballots are equal. Hence, winning signifies the existence of a scenario in which voters cannot uniquely identify their ballot, thus voters cannot be convinced that their ballot is collected. By comparison, when no winning adversary exists, voters can determine whether their ballot is collected.

Privacy. Privacy requires voting systems to ensure free-choice. Formulations of privacy differ depending on the adversary's capabilities and any operational assumptions. Ballot secrecy formalises a notion of free-choice assuming the adversary's capabilities are limited to controlling ballot collection and assuming voters' ballots are constructed and tallied in the prescribed manner.[3]

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

Smyth captures ballot secrecy as a game that proceeds as follows. First, the adversary picks a pair of votes v_0 and v_1. Secondly, the challenger constructs a ballot for vote v_β, in the manner prescribed by the voting system, where β is a bit chosen uniformly at random. That ballot is given to the adversary. The adversary and challenger repeat the process to construct further ballots, using the same bit β. Thirdly, the adversary constructs a set of ballots, which may include ballots constructed by the adversary and ballots constructed by the challenger. Thus, the game captures a setting where the adversary casts ballots on behalf of some voters and controls the distribution of votes cast by the remaining voters. Fourthly, the challenger tallies the set of ballots, in the manner prescribed by the voting system, to determine the election outcome, which is given to the adversary. Finally, the adversary is tasked with determining if $\beta = 0$ or $\beta = 1$. To avoid trivial distinctions, we require that the aforementioned distribution of votes cast (which the adversary controls) remains constant regardless of whether $\beta = 0$ or $\beta = 1$. If the adversary wins, then a voter's vote can be revealed, otherwise, it cannot, i.e., the voting system provides ballot secrecy.

Equipped with definitions of ballot secrecy and verifiability, we can analyse existing voting systems to determine whether they are secure and we can build new systems that can be proven secure.

We introduce two voting systems to demonstrate how secrecy and verifiability can be achieved. The first (Nonce) instructs each voter to pair their vote with

[3] *Ballot secrecy* and *privacy* occasionally appear as synonyms in the literature. We favour ballot secrecy to avoid confusion with other privacy notions, such as receipt-freeness and coercion resistance.

a nonce and instructs the tallier to publish the distribution of votes. The second (Enc2Vote) instructs each voter to encrypt their vote using an asymmetric encryption scheme and instructs the tallier to decrypt the encrypted votes and publish the distribution of votes. Verifiability is ensured by the former system, because voter's can use their nonce to check that their ballot is collected (individual verifiability) and anyone can recompute the election outcome to check that it corresponds to votes expressed in collected ballots (universal verifiability). But, ballot secrecy is not ensured, because voters' votes are revealed. By comparison, secrecy is ensured by the latter system, because asymmetric encryption can ensure that votes cannot be recovered from ballots and the tallying procedure ensures that individual votes are not revealed. But, verifiability is not ensured. Indeed, spurious election outcomes need not correspond to the encrypted votes, which violates universal verifiability, and public keys can be maliciously constructed such that ciphertexts collide, which violates individual verifiability. Thus, Enc2Vote ensures secrecy not verifiability, and Nonce achieves the reverse. More advanced voting systems must simultaneously satisfy both secrecy and verifiability, and we will consider the Helios voting system.

Helios is an open-source, web-based electronic voting system, which has been used in binding elections. In particular, the International Association of Cryptologic Research (IACR) has used Helios annually since 2010 to elect board members [BVQ10, HBH10], the Association for Computing Machinery (ACM) used Helios for their 2014 general election [Sta14], the Catholic University of Louvain used Helios to elect their university president in 2009 [AMPQ09], and Princeton University has used Helios since 2009 to elect student governments. Helios is intended to satisfy verifiability whilst maintaining ballot secrecy. For ballot secrecy, each voter is instructed to encrypt their vote using an asymmetric homomorphic encryption scheme. Encrypted votes are homomorphically combined and the homomorphic combination is decrypted to reveal the outcome [AMPQ09]. Alternatively, a mixnet is applied to the encrypted votes and the mixed encrypted votes are decrypted to reveal the outcome [Adi08, BGP11]. We refer to the former voting system as *Helios* and the latter as *Helios Mixnet*. For verifiability, the encryption step is accompanied by a non-interactive zero-knowledge proof demonstrating correct computation. This ensures homomorphic combinations of encrypted votes and mixed encrypted votes can be decrypted, hence, the outcome can be recovered. Helios additionally requires proof that ciphertexts encrypt votes. This prevents an adversarial voter crafting a ciphertext that could be combined with others to derive an election outcome in the voter's favour. (E.g., votes might be switched between candidates.) The decryption step is similarly accompanied by a non-interactive zero-knowledge proof to prevent spurious outcomes.

Structure. Figure 1 introduces notation and games-based security definitions. Section 2 presents the syntax we will use to model voting systems and to define their properties. Section 3 introduces definitions of universal (§3.1) and individual (§3.2) verifiability by Smyth, Frink & Clarkson, models the Nonce voting system and proves that verifiability definitions are satisfied (§3.3), contextualises our definitions of verifiability (§3.4), and provides insights into further aspects of verifiability (§3.5). Section 4 introduces the definition of ballot secrecy by Smyth (§4.1), models the Enc2Vote voting system (§4.2), introduces sufficient conditions for ballot secrecy that simplify proofs and proves that Enc2Vote satisfies secrecy (§4.3), contextualises our definition of ballot secrecy (§4.4), and provides insights into further aspects of secrecy (§4.5). Section 5 introduces Helios; shows how our definitions of secrecy and universal verifiability detect known vulnerabilities against the initial Helios release (§5.1), the current release (§5.2), and the next release (§5.3), and discusses fixes; and proves that a variant of Helios satisfies our secrecy and verifiability definitions (§5.4). Section 6 introduces Helios Mixnet, detects a universal verifiability vulnerability using our definition, discusses a fix, and proves secrecy and verifiability are satisfied when the fix is applied. Finally, Section 7 presents a brief reflection. (The manuscript is intended to be read sequentially, but Sections 3 & 4 can be studied in any order, Figure 1 can be skipped by readers familiar with games, Sections 3.4, 3.5 & 4.3–4.5 can be skipped by readers uninterested in the broader literature, and Sections 5 & 6 can be read in any order too.)

2 Election scheme syntax

We recall *election scheme* syntax (Definition 1), which captures voting systems that consist of the following four steps. First, a tallier generates a key pair. Secondly, each voter constructs and casts a ballot for their vote. These ballots are collected and recorded on a bulletin board. Thirdly, the tallier tallies the collected ballots and announces an outcome, i.e., a distribution of votes. The chosen representative is derived from this distribution, e.g., as the candidate with the most votes.[4] Finally, voters and other interested parties check that the outcome corresponds to votes expressed in collected ballots.

Definition 1 (Election scheme [SFC17])**.** *An* election scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *such that:*[5]

Setup*, denoted* $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$*, is run by the tallier. The algorithm takes a security parameter* κ *as input and outputs a key pair* pk, sk*, a maximum number of ballots* mb*, and a maximum number of candidates* mc*.*

Vote*, denoted* $b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa)$*, is run by voters. The algorithm takes as input a public key* pk*, a voter's vote* v*, some number of candidates* nc*, and*

[4] Beyond first-past-the-post voting systems, Smyth shows the syntax can model ranked-choice voting systems too [Smy17].

[5] The syntax bounds the number of ballots mb, respectively candidates mc, to broaden the correctness definition's scope (indeed, Helios requires mb and mc to be less than or equal to the size of the underlying encryption scheme's message space); represents votes as integers, rather than alphanumeric strings, for brevity; and omits algorithm Verify, because we focus on ballot secrecy, not verifiability.

Figure 1 Preliminaries: Games and notation

A game formulates a series of interactions between a benign challenger, a malicious adversary, and a cryptographic scheme. The adversary wins by completing a task that captures an execution of the scheme in which security is broken, i.e., winning captures what should be unachievable. Tasks can generally be expressed as *indistinguishability* or *reachability* requirements. For example, universal verifiability can be expressed as the inability to reach a state that causes a voting system's checks to succeed for invalid election outcomes, or fail for valid outcomes. Moreover, ballot secrecy can be expressed as the inability to distinguish between an instance of a voting system in which voters cast some votes, from another instance in which the voters cast a permutation of those votes.

Formally, games are probabilistic algorithms that output booleans. We let $A(x_1, \ldots, x_n; r)$ denote the output of probabilistic algorithm A on inputs $x_1, \ldots, x_n$ and random coins r, and we let $A(x_1, \ldots, x_n)$ denote $A(x_1, \ldots, x_n; r)$, where coins r are chosen uniformly at random. Moreover, we let $x \leftarrow T$ denote assignment of T to x, and $x \leftarrow_R S$ denote assignment to x of an element chosen uniformly at random from set S. Using our notation, we can formulate the following game $\mathsf{Exp}(H, S, \mathcal{A})$ that tasks an adversary $\mathcal{A}$ to distinguish between a function H and a simulator S: $m \leftarrow \mathcal{A}(); \beta \leftarrow_R \{0,1\}$; **if** $\beta = 0$ **then** $x \leftarrow H(m)$; **else** $x \leftarrow S(m)$; $g \leftarrow \mathcal{A}(x);$ **return** $g = \beta$. Adversaries are *stateful*, i.e., information persists across invocations of an adversary in a game. In particular, adversaries can access earlier assignments. For instance, the adversary's second instantiation in game Exp has access to any assignments made during its first instantiation. An adversary *wins* a game by causing it to output true ($\top$) and the adversary's *success* in a game $\mathsf{Exp}(\cdot)$, denoted $\mathsf{Succ}(\mathsf{Exp}(\cdot))$, is the probability that the adversary wins, that is, $\mathsf{Succ}(\mathsf{Exp}(\cdot)) = \Pr[x \leftarrow \mathsf{Exp}(\cdot) : x = \top]$. We focus on computational security, rather than information-theoretic security, and tolerate breaks by adversaries in non-polynomial time and breaks with negligible success, since such breaks are infeasible in practice.

Game Exp captures a single interaction between the challenger and the adversary. We can extend games with oracles to capture arbitrarily many interactions. For instance, we can formulate a strengthening of Exp as follows: $\beta \leftarrow_R \{0,1\}; g \leftarrow \mathcal{A}^{\mathcal{O}}(x);$ **return** $g = \beta$, where $\mathcal{A}^{\mathcal{O}}$ denotes $\mathcal{A}$'s access to oracle $\mathcal{O}$ and $\mathcal{O}(m)$ computes **if** $\beta = 0$ **then** $x \leftarrow H(m)$; **else** $x \leftarrow S(m)$; **return** x. Oracles may access game parameters such as bit β.

Beyond the above notation, we let $x[i]$ denote component i of vector x and let $|x|$ denote the length of vector x. Moreover, we write $(x_1, \ldots, x_{|T|}) \leftarrow T$ for $x \leftarrow T; x_1 \leftarrow x[1]; \ldots; x_{|T|} \leftarrow x[|T|]$, when T is a vector, and $x, x' \leftarrow_R S$ for $x \leftarrow_R S; x' \leftarrow_R S$.

a security parameter κ. Vote v should be selected from a sequence $1, \ldots, nc$ of candidates. The algorithm outputs a ballot b or error symbol $\perp$.

Tally, *denoted $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$, is run by the tallier. The algorithm takes as input a private key sk, a bulletin board $\mathfrak{bb}$, some number of candidates nc, and a security parameter κ, where $\mathfrak{bb}$ is a set. And outputs an election outcome $\mathfrak{v}$ and a non-interactive tallying proof pf. The election outcome must be a vector of length nc and each index v of that vector should indicate the number of votes for candidate v. Moreover, the tallying proof should demonstrate that the outcome corresponds to votes expressed in ballots on the bulletin board.*

Verify, *denoted $s \leftarrow \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa)$, is run to audit an election. The algorithm takes as input a public key pk, a bulletin board $\mathfrak{bb}$, some number of candidates nc, an election outcome $\mathfrak{v}$, a tallying proof pf, and a security parameter κ. And outputs a bit s, which is 1 if the outcome should be accepted and 0 otherwise. We require the algorithm to be deterministic.*

Election schemes must satisfy correctness*: there exists a negligible function* negl*, such that for all security parameters κ, integers nb and nc, and votes $v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$, it holds that, given a zero-filled vector $\mathfrak{v}$ of length nc, we have:*

$$
\begin{aligned}
&\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa);\\
&\quad \textbf{for } 1 \le i \le nb \textbf{ do}\\
&\quad\quad b_i \leftarrow \mathsf{Vote}(pk, v_i, nc, \kappa);\\
&\quad\quad \mathfrak{v}[v_i] \leftarrow \mathfrak{v}[v_i] + 1;\\
&\quad (\mathfrak{v}', pf) \leftarrow \mathsf{Tally}(sk, \{b_1, \ldots, b_{nb}\}, nc, \kappa) :\\
&\quad nb \le mb \wedge nc \le mc \Rightarrow \mathfrak{v} = \mathfrak{v}'] > 1 - \mathsf{negl}(\kappa).
\end{aligned}
$$

The syntax provides a language to model voting systems and the correctness condition ensures that such systems function. That is, election outcomes correspond to votes expressed in ballots, when ballots are constructed and tallied in the prescribed manner. We will use our syntax to express verifiability and secrecy properties of election schemes, moreover, we will model and analyse voting systems, including Helios and Helios Mixnet.

3 Verifiability

The Australian system is reliant on monitoring to ensure election outcomes correspond to votes expressed in collected ballots, moreover, depositing ballots into ballot boxes suffices to ensure they collected. By comparison, election schemes compute outcomes in a manner that should not be monitored. Indeed, such monitoring would reveal the tallier's private key, which would compromise ballot secrecy. Furthermore, casting a ballot is insufficient to ensure it is collected, because an adversary may discard or modify ballots. Nevertheless, election schemes produce tallying proofs to provide evidence that outcomes are correctly computed and voters can check whether their ballot is collected. The former notion is formalised by universal verifiability and the latter by individual verifiability.

3.1 Universal verifiability

Universal verifiability asserts that anyone must be able to check whether an election outcome corresponds to votes expressed in collected ballots. Since checks can be performed by algorithm Verify, it suffices that Verify accept if and only if the outcome corresponds to votes expressed in collected ballots. The *only if* requirement is captured by soundness, which requires algorithm Verify to only accept correct outcomes, and the *if* requirement is captured by completeness, which requires election outcomes produced by algorithm Tally to be accepted by algorithm Verify.

Soundness. Correct outcomes are formalised using function *correct-outcome*. That function uses a predicate $(\exists^{=\ell} x : P(x))$ that holds exactly when there are ℓ distinct values of x for which $P(x)$ is satisfied [Sch05]. (Variable x is bound by the predicate and integer ℓ is free.) Using the predicate, function *correct-outcome* is defined such that

$$\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)[v] = \ell \text{ iff } \\ \exists^{=\ell} b \in \mathfrak{bb} \setminus \{\bot\} : \exists r : b = \mathsf{Vote}(pk, v, nc, \kappa; r),$$

where $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$ is a vector of length nc and $1 \leq v \leq nc$. Hence, component v of vector $\mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$ equals ℓ iff there exist ℓ ballots for vote v on the bulletin board. The function requires that ballots be interpreted for only one candidate, which can be ensured by injectivity.

Definition 2 (Injectivity [SFC17, Smy18b])**.** *An election scheme* (Setup, Vote, Tally, Verify) *satisfies* Injectivity*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, security parameters κ and computations $(pk, nc, v, v') \leftarrow \mathcal{A}(\kappa); b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa); b' \leftarrow \mathsf{Vote}(pk, v', nc, \kappa)$ such that $v \neq v' \wedge b \neq \bot \wedge b' \neq \bot$, we have $b \neq b'$.*

Our definition of injectivity ensures that a ballot for vote v can never be interpreted for a distinct vote v', hence, votes expressed in ballots correspond to unique outcomes.

Equipped with a notion of correct outcomes, we formalise soundness (Definition 3) as a game that tasks the adversary to compute inputs to algorithm Verify (Line 1), including an election outcome and some ballots, that cause the algorithm to accept when the outcome does not correspond to the votes expressed in those ballots (Line 2).

Definition 3 (Soundness [SFC17])**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme, $\mathcal{A}$ be an adversary, κ be a security parameter, and* $\mathsf{Soundness}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{Soundness}(\Gamma, \mathcal{A}, \kappa) =$

$(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf) \leftarrow \mathcal{A}(\kappa)$;
return $\mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1 \wedge \mathfrak{v} \neq \mathit{correct\text{-}outcome}(pk, nc, \mathfrak{bb}, \kappa)$;

We say Γ *satisfies* Soundness*, if* Γ *satisfies injectivity and for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{Soundness}(\Gamma, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$*.*

An election scheme satisfies Soundness when algorithm Verify only accepts outcomes that correspond to votes expressed in collected ballots.

> **Design guideline 1.** *Verification must only accept outcomes that correspond to votes expressed in collected ballots.*

Completeness. We formalise completeness (Definition 4) as a game that tasks the adversary to compute a bulletin board and some number of candidates (Line 2) such that the corresponding election outcome computed by algorithm Tally (Line 3) is rejected by algorithm Verify (Line 4), when the key pair is computed by algorithm Setup (Line 1).

Definition 4 (Completeness [SFC17])**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Completeness}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{Completeness}(\Gamma, \mathcal{A}, \kappa) =$

1 $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
2 $(\mathfrak{bb}, nc) \leftarrow \mathcal{A}(pk, \kappa)$;
3 $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$;
4 **return** $\mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) \neq 1 \wedge |\mathfrak{bb}| \leq mb \wedge nc \leq mc$;

We say Γ *satisfies* Completeness*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{Completeness}(\Gamma, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$*.*

An election scheme satisfies Completeness when algorithm Verify accepts outcomes computed by algorithm Tally, for key pairs computed by algorithm Setup. It follows that completeness implies an aspect of accountability. Indeed, if verification fails, then the tallier is responsible for that failure, in particular, they must have incorrectly computed their key pair or the election outcome.

> **Design guideline 2.** *Tallying must produce outcomes that will be accepted during verification.*

We formalise universal verifiability by combining the above notions.

Definition 5 (Universal-Verifiability [SFC17, Smy18b])**.** *An election scheme* Γ *satisfies* Universal-Verifiability*, if* Soundness *and* Completeness *are satisfied.*

3.2 Individual verifiability

Individual verifiability asserts that voters must be able to check whether their ballot is amongst those collected. Since ballots should be collected and recorded on a bulletin board, and since the board must be available to everyone, it suffices for voters to check that their ballot (i.e., the ballot they constructed) is on the bulletin board. Hence, it is necessary for voters to check that their ballot has not been omitted from the bulletin board. Yet, this is insufficient, because the presence of a ballot identical to a voter's ballot, does not imply the presence of the ballot constructed by the voter. Indeed, such a ballot might have been constructed by another voter. Thus, individual verifiability requires that voters must be able to uniquely identify their ballot, i.e., ballots do not collide. We formalise individual verifiability (Definition 6) as a game that tasks the adversary to compute inputs to algorithm Vote (Line 1) that cause the algorithm to output ballots (Lines 2 & 3) that collide (Line 4).

Definition 6 (Individual verifiability [SFC17])**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Individual\text{-}Verifiability}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{Individual\text{-}Verifiability}(\Gamma, \mathcal{A}, \kappa) =$

1 $(pk, nc, v, v') \leftarrow \mathcal{A}(\kappa)$;
2 $b \leftarrow \mathsf{Vote}(pk, nc, v, \kappa)$;
3 $b' \leftarrow \mathsf{Vote}(pk, nc, v', \kappa)$;
4 **return** $b = b' \wedge b \neq \perp \wedge b' \neq \perp$;

We say Γ *satisfies* Individual-Verifiability*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{Individual\text{-}Verifiability}(\Gamma, \mathcal{A}, \kappa)) \leq \mathsf{negl}(\kappa)$*.*

An election scheme satisfies Individual-Verifiability when algorithm Vote generates uniquely identifiable ballots, i.e., ballots that do not collide.[6]

> **Design guideline 3.** *Ballots must be distinct.*

3.3 Example

We model our Nonce voting system (§1) as the following election scheme.

Definition 7 (Nonce [SFC17])**.** Nonce *is defined as follows:*

[6]Correctness, individual verifiability and injectivity all require that ballots do not collide, albeit under different assumptions. Indeed, correctness requires that ballots do not collide, with overwhelming probability, for public keys computed by algorithm Setup; Injectivity requires that ballots for distinct votes never collide; and Individual-Verifiability requires that ballots do not collide with overwhelming probability. Hence, Individual-Verifiability implies that ballots do not collide in the context of correctness. But, Individual-Verifiability and Injectivity are orthogonal, in particular, Individual-Verifiability permits collisions with negligible probability and Injectivity permits collisions between ballots for the same vote.

- $\mathsf{Setup}(\kappa)$ *outputs* $(\perp, \perp, p_1(k), p_2(k))$*, where* p_1 *and* p_2 *are some polynomial functions.*
- $\mathsf{Vote}(pk, v, nc, \kappa)$ *samples nonce* $r \leftarrow_R \mathbb{Z}_{2^\kappa}$ *and outputs* (r, v)*.*
- $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$ *computes a vector* $\mathfrak{v}$ *of length* nc*, such that* $\mathfrak{v}$ *is the tally of each vote in set* $\mathfrak{bb}$ *which is paired with a nonce from* $\mathbb{Z}_{2^\kappa}$*, and outputs* $(\mathfrak{v}, \perp)$*.*
- $\mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa)$ *outputs* 1 *if* $(\mathfrak{v}, pf) = \mathsf{Tally}(\perp, \mathfrak{bb}, nc, \kappa)$ *and* 0 *otherwise.*

Lemma 1. Nonce *is an election scheme.*

To prove our lemma, we show that Nonce satisfies the correctness property of Definition 1.

Proof. Let Nonce = (Setup, Vote, Tally, Verify). Moreover, let κ be a security parameter, nb and nc be integers, $v_1, \dots, v_{nb} \in \{1, \dots, nc\}$ be votes, and $\mathfrak{v}$ be a zero-filled vector of length nc. Suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$ such that $nb \le mb \wedge nc \le mc$. Further suppose we compute **for** $1 \le i \le nb$ **do** $b_i \leftarrow \mathsf{Vote}(pk, v_i, nc, \kappa); \mathfrak{v}[v_i] \leftarrow \mathfrak{v}[v_i] + 1$. Moreover, suppose $(\mathfrak{v}', pf)$ is an output of $\mathsf{Tally}(sk, \{b_1, \dots, b_{nb}\}, nc, \kappa)$. To prove correctness, it suffices to prove $\mathfrak{v} = \mathfrak{v}'$, with overwhelming probability. We have that $\mathfrak{v}$ is the election outcome corresponding to votes $v_1, \dots, v_{nb}$. Moreover, by definition of algorithms Vote and Tally, we have $\mathfrak{v}'$ is the tally of the votes in set $\{(r_1, v_1), \dots, (r_{nb}, v_{nb})\}$, where $r_1, \dots, r_{nb}$ are nonces chosen uniformly at random from $\mathbb{Z}_{2^\kappa}$. Hence, $\mathfrak{v}'$ is the election outcome corresponding to votes $v_1, \dots, v_{nb}$, i.e., $\mathfrak{v} = \mathfrak{v}'$, assuming nonces $r_1, \dots, r_{nb}$ are pairwise distinct, which holds with overwhelming probability, thereby concluding our proof. □

Intuitively, election scheme Nonce satisfies Individual-Verifiability, because nonces collide with negligible probability, hence, ballots collide with negligible probability too, which suffices for Individual-Verifiability. Moreover, Universal-Verifiability is satisfied too, because outcomes correspond to votes expressed in collected ballots (Soundness) and such outcomes are accepted (Completeness).

Proposition 2. Nonce *satisfies* Individual-Verifiability *and* Universal-Verifiability.

Proof. Let Nonce = (Setup, Vote, Tally, Verify). We proceed by proving that Individual-Verifiability, Injectivity (which is a prerequisite for Soundness), Soundness, and Completeness are satisfied.

For individual verifiability, suppose an adversary outputs public key pk, some number of candidates nc, and votes v and v'. Further suppose b is an output of $\mathsf{Vote}(pk, nc, v, \kappa)$ and b' is an output of $\mathsf{Vote}(pk, nc, v', \kappa)$, for some security parameter κ. By definition of algorithm Vote, we have $b = (r, v)$ and $b' = (r', v')$, where r and r' are nonces chosen uniformly at random from $\mathbb{Z}_{2^\kappa}$. Hence, r and r' are distinct with overwhelming probability, thus, $b \neq b'$ with overwhelming probability, as required.

Our proof of injectivity is similar to our proof of individual verifiability, except we require $v \neq v'$, which ensures $b \neq b'$.

For soundness, let $\mathcal{A}$ be an adversary and κ be a security parameter. Suppose $(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf)$ is an output of $\mathcal{A}(\kappa)$ such that $\mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1$. By definition of algorithm Verify, we have $(\mathfrak{v}, pf) = \mathsf{Tally}(\perp, \mathfrak{bb}, nc, \kappa)$ and, by definition of algorithm Tally, we have $\mathfrak{v}$ is a vector of length nc such that $\mathfrak{v}$ is the tally of each vote in set $\mathfrak{bb}$ which is paired with a nonce from $\mathbb{Z}_{2^\kappa}$. Hence, for each $v \in \{1, \ldots, nc\}$ we have

$$\mathfrak{v}[v] = \ell \Leftrightarrow \exists^{=\ell} b \in \mathfrak{bb} : \exists r \in \mathbb{Z}_{2^\kappa} : b = (v, r)$$

and, by definition of algorithm Vote and since $\perp$ is not a pair, we have

$$\Leftrightarrow \exists^{=\ell} b \in \mathfrak{bb} \setminus \{\perp\} : \exists r : b = \mathsf{Vote}(pk, v, nc, \kappa; r)$$

Thus, $\mathfrak{v} = \textit{correct-outcome}(pk, nc, \mathfrak{bb}, \kappa)$, as required.

For completeness, let $\mathcal{A}$ be an adversary and κ be a security parameter. Suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$, $(\mathfrak{bb}, nc)$ is an output of $\mathcal{A}(pk, \kappa)$, and $(\mathfrak{v}, pf)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$. By definition of algorithm Setup, we have $pk = \perp$, hence, $(\mathfrak{v}, pf) = \mathsf{Tally}(\perp, \mathfrak{bb}, nc, \kappa)$. It follows by the definition of algorithm Verify that $\mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1$, as required. □

3.4 Related definitions of verifiability

Discussion of verifiability originates from Cohen & Fischer [CF85] and is advanced by Fujioka, Okamoto & Ohta [FOO92], Benaloh & Tuinstra [BT94a] and Sako & Kilian [SK95]. More recently, definitions of universal verifiability have been proposed by Juels, Catalano & Jakobsson [JCJ10], Cortier *et al.* [CGGI14] and Kiayias, Zacharias & Zhang [KZZ15]. Smyth, Frink & Clarkson [SFC17, §7] show that definitions by Juels, Catalano & Jakobsson and Cortier *et al.* do not detect vulnerabilities that arise when tallying and verification procedures are corrupt nor when verification procedures reject legitimate outcomes. Moreover, they show that the definition by Kiayias, Zacharias & Zhang does not detect the latter class of vulnerabilities. By comparison, the definition of universal verifiability that we consider (Definition 5) detects these vulnerabilities and appears to be the strongest definition in the literature.

Küsters *et al.* [KTV10, KTV11, KTV12b] propose an alternative, holistic notion of verifiability called *global verifiability*, which must be instantiated with a goal. Smyth, Frink & Clarkson [SFC17, §8] show that goals proposed by Küsters *et al.* [KTV15, §5.2] and by Cortier *et al.* [CGK$^+$16, §10.2] are too strong (in the sense that they cannot be satisfied by some verifiable voting systems, including Helios). Moreover, Smyth, Frink & Clarkson propose a slight weakening of the goal by Küsters *et al.* and prove that their notion of verifiability is strictly stronger than global verifiability with that goal. Nonetheless, the "gap" is due to an uninteresting technical detail and those definitions might coincide if the gap is filled.

Beyond the computational model of security, Smyth *et al.* [SRKK10] formulate a definition of verifiability in the applied pi calculus. The definition is amenable to automated reasoning, but it is stronger than necessary and cannot be satisfied by many election schemes, including Helios. Kremer *et al.* [KRS10] overcome this limitation with a weaker definition that sacrifices amenability to automated reasoning, and Smyth [Smy11, §3] extends this definition.

3.5 Further notions of verifiability

Individual verifiability formalises a notion of uniquely identifiable ballots, assuming ballots are constructed in the prescribed manner. We have seen that voting system Nonce satisfies this notion, but individual verifiability does not ensure ballots are unique when voters deviate from the prescribed voting procedure. Indeed, a ballot's nonce serves as its unique identifier in Nonce and substituting that nonce for some other value may compromise individual verifiability. Further notions of verifiability, such as cast-as-intended [AN06], are needed when deviations from the voting procedure are possible, e.g., when the voting procedure is subverted by an adversary.

The soundness aspect of universal verifiability formalises a notion of only accepting election outcomes that correspond to votes expressed in collected ballots. Our definition does not ensure outcomes include only voters' votes and at most one vote each. Indeed, an adversary that controls ballot collection can "stuff" the bulletin board with ballots and the corresponding votes will be included in the election outcome. Some voting systems rely on a trusted third party to authenticate ballots and only tally ballots that have been authenticated. E.g., Helios supports authentication via Facebook, Google and Twitter using OAuth.[7] Other voting systems use cryptography to ensure that only voters can construct authorised ballots. E.g., the voting system by Juels, Catalano & Jakobsson uses a combination of encrypted nonces and plaintext equality tests for authentication [JCJ10]. Albeit such systems seem to require expensive infrastructures for voter credentials and are reliant on a trusted third party to certify credentials. Our soundness definition is suitable for analysing the former class of voting systems, and Smyth, Frink & Clarkson [SFC17] formulate an alternative soundness definition to analyse the latter class. (Quaglia & Smyth [QS18a] define a transformation from schemes satisfying our soundness definition to schemes satisfying the alternative soundness definition.) Ultimately, we would prefer not to trust any third party, but that does not seem possible in a scalable manner.

4 Ballot secrecy

The Australian system is reliant on uniform ballots to ensure voters' votes are not revealed. Indeed, uniformity ensures ballots are indistinguishable during distribution and the isolation of polling booths ensures votes are not revealed whilst marking. Moreover, folded ballots are indistinguishable during collection

[7] Meyer & Smyth describe the application of OAuth in Helios [MS17].

and indistinguishability of markings can ensure votes are not revealed whilst tallying. Hence, the Australian system derives ballot secrecy from physical characteristics of the world. By comparison, election schemes cannot rely on such physical characteristics,[8] they must rely on cryptography to ensures voters' votes are not revealed.

Some scenarios inevitably reveal voters' votes: Unanimous election outcomes reveal how everyone voted and, more generally, election outcomes can be coupled with partial knowledge on the distribution of voters' votes to deduce voters' votes. For example, suppose Alice, Bob and Mallory participate in a referendum and the outcome has frequency two for 'yes' and one for 'no.' Mallory and Alice can deduce Bob's vote by pooling knowledge of their own votes. Similarly, Mallory and Bob can deduce Alice's vote. Furthermore, Mallory can deduce that Alice and Bob both voted yes, if she voted no. For simplicity, our informal definition of ballot secrecy (§1) deliberately omitted side-conditions which exclude these inevitable revelations and which are necessary for satisfiability. We now refine that definition as follows:

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

This refinement ensures the aforementioned examples are not violations of ballot secrecy. By comparison, if Mallory votes yes and she can deduce the vote of Alice, without knowledge of Bob's vote, then ballot secrecy is violated.

4.1 Security definition

We formalise ballot secrecy (Definition 8) as a game that tasks the adversary to select two distributions of votes, construct a bulletin board from ballots for one distribution, which is decided by a coin flip, and (non-trivially) determine the result of the coin flip from the resulting election outcome and tallying proof. That is, the game tasks the adversary to distinguish between an instance of the voting system for one distribution, from another instance with the other distribution, when the votes cast from each distribution are permutations of each other (hence, the distinction is non-trivial). The game proceeds as follows: The challenger generates a key pair (Line 1), the adversary chooses some number of candidates (Line 2), and the challenger flips a coin (Line 3). The adversary computes a bulletin board from ballots for one of two possible distributions (Line 5), where the distributions are chosen by the adversary, the choice between distributions is determined by the coin flip, and the ballots (for one of the distributions) are constructed by an oracle (further ballots may be constructed by the adversary). The challenger tallies the bulletin board to derive the election

[8] Some electronic voting systems are reliant on physical characteristics of the world. For instance, MarkPledge [Nef04], Pret à Voter [CRS05], Remotegrity [ZCC$^+$13], Scantegrity II [CCC$^+$08] and Three Ballot [RS07] are reliant on features implemented with paper, such as scratch-off surfaces and detachable columns. But these systems fall outside the scope of our election scheme syntax.

outcome and tallying proof (Line 6), which are given to the adversary and the adversary is tasked with determining the result of the coin flip (Line 7 & 8).

Definition 8 (Ballot-Secrecy [Smy18a])**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be a security parameter, and* $\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa) =$

1 $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
2 $nc \leftarrow \mathcal{A}(pk, \kappa)$;
3 $\beta \leftarrow_R \{0, 1\}$;
4 $L \leftarrow \emptyset$;
5 $\mathfrak{bb} \leftarrow \mathcal{A}^{\mathcal{O}}()$;
6 $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$;
7 $g \leftarrow \mathcal{A}(\mathfrak{v}, pf)$;
8 **return** $g = \beta \wedge balanced(\mathfrak{bb}, nc, L) \wedge 1 \leq nc \leq mc \wedge |\mathfrak{bb}| \leq mb$;

Predicate $balanced(\mathfrak{bb}, nc, L)$ *holds when: for all votes* $v \in \{1, \ldots, nc\}$ *we have* $|\{b \mid b \in \mathfrak{bb} \wedge \exists v_1 . (b, v, v_1) \in L\}| = |\{b \mid b \in \mathfrak{bb} \wedge \exists v_0 . (b, v_0, v) \in L\}|$*. And oracle* $\mathcal{O}$ *is defined as follows:*

- $\mathcal{O}(v_0, v_1)$ *computes* $b \leftarrow \mathsf{Vote}(pk, v_\beta, nc, \kappa); L \leftarrow L \cup \{(b, v_0, v_1)\}$ *and outputs* b*, where* $v_0, v_1 \in \{1, ..., nc\}$*.*

We say Γ *satisfies* Ballot-Secrecy*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{Ballot\text{-}Secrecy}(\Gamma, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$*.*

An election scheme satisfies ballot secrecy when algorithm Vote outputs ballots that do not reveal votes and algorithm Tally outputs election outcomes and proofs that do not reveal the relation between votes expressed in collected ballots and the outcome.

Game Ballot-Secrecy tasks the adversary to compute a bulletin board, from ballots constructed by an oracle for one of two possible distributions, and determine which distribution was used from the election outcome and proof generated from tallying that board. The choice between distributions is determined by the result β of a coin flip, and the oracle outputs a ballot for vote v_β on input of a pair of votes v_0, v_1. Hence, the oracle constructs ballots for one of two possible distributions, where the distributions are chosen by the adversary, and the bulletin board may contain those ballots in addition to ballots constructed by the adversary.

Election schemes reveal the number of votes for each candidate (i.e., the election outcome). Hence, to avoid trivial distinctions in game Ballot-Secrecy, we require that runs of the game are *balanced*: "left" and "right" inputs to the oracle are equivalent, when the corresponding outputs appear on the bulletin board. For example, suppose the inputs to the oracle are $(v_{1,0}, v_{1,1}), \ldots, (v_{n,0}, v_{n,1})$ and the corresponding outputs are $b_1, \ldots, b_n$, further suppose the bulletin board is $\{b_1, \ldots, b_\ell\}$ such that $\ell \leq n$. That game is balanced if the "left" inputs

$v_{1,0}, \ldots, v_{\ell,0}$ are a permutation of the "right" inputs $v_{1,1}, \ldots, v_{\ell,1}$. The balanced condition prevents trivial distinctions. For instance, an adversary that computes a bulletin board containing only the ballot output by an oracle query with input $(1, 2)$ cannot win the game, because it is unbalanced. Albeit, that adversary could trivially determine whether $\beta = 0$ or $\beta = 1$, given the tally of that bulletin board. (Formally defining a winning adversary is left as an exercise for the reader.)

Intuitively, if the adversary wins game Ballot-Secrecy, then there exists a strategy to distinguish ballots. Indeed, such an adversary can distinguish between an instance of the voting system in which voters cast some votes, from another instance in which voters cast a permutation of those votes, thus, voters' votes are revealed. Otherwise, the adversary is unable to distinguish between a voter casting a ballot for vote v_0 and another voter casting a ballot for vote v_1, hence, voters' votes cannot be revealed.

4.2 Example

We recall syntax for asymmetric encryption scheme in the appendix and model our Enc2Vote voting system (§1) as the following election scheme.

Definition 9 (Enc2Vote [Smy18a])**.** *Given an asymmetric encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, we define* $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *such that:*

- $\mathsf{Setup}(\kappa)$ *computes* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa); pk' \leftarrow (pk, \mathfrak{m}); sk' \leftarrow (pk, sk)$*, derives* mc *as the largest integer such that* $\{0, \ldots, mc\} \subseteq \{0\} \cup \mathfrak{m}$ *and for all* $m_0, m_1 \in \{1, \ldots, mc\}$ *we have* $|m_0| = |m_1|$*, and outputs* $(pk', sk', p(\kappa), mc)$*, where* p *is a polynomial function.*

- $\mathsf{Vote}(pk', v, nc, \kappa)$ *parses* pk' *as pair* $(pk, \mathfrak{m})$*, outputting* $\perp$ *if parsing fails or* $v \notin \{1, \ldots, nc\} \vee \{1, \ldots, nc\} \not\subseteq \mathfrak{m}$*, computes* $b \leftarrow \mathsf{Enc}(pk, v)$*, and outputs* b*.*

- $\mathsf{Tally}(sk', \mathfrak{bb}, nc, \kappa)$ *initialises* $\mathfrak{v}$ *as a zero-filled vector of length* nc*, parses* sk' *as pair* (pk, sk)*, outputting* $(\mathfrak{v}, \perp)$ *if parsing fails, computes* **for** $b \in \mathfrak{bb}$ **do** $v \leftarrow \mathsf{Dec}(sk, b)$; **if** $1 \leq v \leq nc$ **then** $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$*, and outputs* $(\mathfrak{v}, \epsilon)$*, where* ϵ *is a constant symbol.*

- $\mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa)$ *outputs 1.*

To ensure $\mathsf{Enc2Vote}(\Pi)$ is an election scheme, we require asymmetric encryption scheme Π to produce distinct ciphertexts with overwhelming probability, otherwise correctness cannot be satisfied, as the following lemma demonstrates.

Lemma 3. *There exists an asymmetric encryption scheme* Π *such that* $\mathsf{Enc2Vote}(\Pi)$ *is not an election scheme.*

To prove our lemma, we show that colliding ciphertexts suffice to ensure that election scheme Enc2Vote cannot satisfy the correctness property of Definition 1.

Proof. Let $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$. Suppose (pk, sk, mb, mc) is an output of $\mathsf{Setup}(\kappa)$ and b and b' are outputs of $\mathsf{Vote}(pk, v, nc, \kappa)$ such that $2 \leq mb \wedge 1 \leq v \leq nc \leq mc$, where κ is a security parameter. Further suppose $\mathfrak{v}$ is a zero-filled vector of length nc, except for index v which contains two. Moreover, suppose $(\mathfrak{v}', pf)$ is an output of $\mathsf{Tally}(sk, \{b, b'\}, nc, \kappa)$. If b and b' collide, then outcome $\mathfrak{v}'$ is computed from set $\{b, b'\} = \{b\}$, therefore, the correct outcome cannot have been computed, we have $\mathfrak{v} \neq \mathfrak{v}'$, with non-negligible probability, hence, correctness is not satisfied. By definition of algorithm Setup, we have pk is a pair and, by definition of algorithm Vote, we have b and b' are ciphertexts on plaintext v. Thus, it remains to show that asymmetric encryption schemes can produce ciphertexts that collide. Indeed, they can: Consider $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that $\mathsf{Enc}(pk, m)$ outputs m and $\mathsf{Dec}(sk, c)$ outputs c. Although Π is clearly not secure, it is straightforward to see that Π satisfies correctness, because $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m; r)) = m$ for all key pairs (pk, sk), plaintexts m, and coins r, which concludes our proof. □

It follows from Lemma 3 that we must restrict the class of asymmetric encryption schemes used to instantiate $\mathsf{Enc2Vote}$. We could consider a broad class of schemes that produce distinct ciphertexts with overwhelming probability, but we favour the narrower class of non-malleable schemes, since we require non-malleability for ballot secrecy. Definitions of non-malleability are complex and proofs of non-malleability are relatively difficult, so we adopt the definition of indistinguishability under parallel attack ($\mathsf{IND\text{-}PA0}$) by Bellare & Sahai [BS99], which is simpler, yet equivalent to their definition of comparison based non-malleability ($\mathsf{CNM\text{-}CPA}$). We recall the definition of $\mathsf{IND\text{-}PA0}$ in the appendix.

Lemma 4. *Given an asymmetric encryption scheme Π satisfying $\mathsf{IND\text{-}PA0}$, we have $\mathsf{Enc2Vote}(\Pi)$ is an election scheme.*

To prove our lemma, we show that $\mathsf{Enc2Vote}$ satisfies the correctness property of Definition 1 when ciphertexts do not collide.

Proof. Let $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Moreover, let κ be a security parameter, nb and nc be integers, $v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$ be votes, and $\mathfrak{v}$ be a zero-filled vector of length nc. Suppose $(pk'$, $sk', mb, mc)$ is an output of $\mathsf{Setup}(\kappa)$ such that $nb \leq mb \wedge nc \leq mc$. Further suppose we compute **for** $1 \leq i \leq nb$ **do** $b_i \leftarrow \mathsf{Vote}(pk, v_i, nc, \kappa); \mathfrak{v}[v_i] \leftarrow \mathfrak{v}[v_i] + 1$. Moreover, suppose $(\mathfrak{v}', pf)$ is an output of $\mathsf{Tally}(sk, \{b_1, \ldots, b_{nb}\}, nc, \kappa)$. To prove correctness, it suffices to prove $\mathfrak{v} = \mathfrak{v}'$, with overwhelming probability.

By definition of algorithm Setup, we have pk' is a pair $(pk, \mathfrak{m})$ and sk' is a pair (pk, sk) such that $(pk, sk, \mathfrak{m})$ was output by $\mathsf{Gen}(\kappa)$. Moreover, mc is the largest integer such that $\{0, \ldots, mc\} \subseteq \{0\} \cup \mathfrak{m}$, hence, $\{1, \ldots, nc\} \subseteq \mathfrak{m}$. It follows by definition of algorithm Vote that for each $i \in \{1, \ldots, nb\}$ we have b_i is an output of $\mathsf{Enc}(pk, v_i)$. Moreover, by definition of algorithm Tally, outcome $\mathfrak{v}'$ is initialised as a zero-filled vector of length nc and computed as follows:

for $b \in \{b_1, \ldots, b_{nb}\}$ **do** $v \leftarrow \mathsf{Dec}(sk, b)$; **if** $1 \leq v \leq nc$ **then** $\mathfrak{v}'[v] \leftarrow \mathfrak{v}'[v] + 1$.

Since Π satisfies $\mathsf{IND\text{-}PA0}$, ciphertexts $b_1, \dots, b_{nb}$ are distinct with overwhelming probability, hence, that computation is equivalent to the following:

for $1 \leq i \leq nb$ **do** $v \leftarrow \mathsf{Dec}(sk, b_i)$; **if** $1 \leq v \leq nc$ **then** $\mathfrak{v}'[v] \leftarrow \mathfrak{v}'[v] + 1$.

Moreover, by correctness of Π, we have $\mathsf{Dec}(sk, b_i) = v_i$ for all $i \in \{1, \dots, nb\}$, with overwhelming probability. Thus, the above computation is equivalent to computing

for $1 \leq i \leq nb$ **do** $\mathfrak{v}'[v_i] \leftarrow \mathfrak{v}'[v_i] + 1$,

with overwhelming probability. It follows that outcomes $\mathfrak{v}$ and $\mathfrak{v}'$ are computed identically, with overwhelming probability, thereby concluding our proof. $\square$

Intuitively, scheme $\mathsf{Enc2Vote}(\Pi)$ satisfies ballot secrecy until tallying, because asymmetric encryption scheme Π can ensure that voters' votes are not revealed, and tallying maintains ballot secrecy by revealing only the election outcome.

Proposition 5. *Given an asymmetric encryption scheme* Π *satisfying* $\mathsf{IND\text{-}PA0}$*, election scheme* $\mathsf{Enc2Vote}(\Pi)$ *satisfies* $\mathsf{Ballot\text{-}Secrecy}$*.*

Proving this proposition and other ballot secrecy results is time consuming. Indeed, Quaglia & Smyth's proof of ballot secrecy for our simple $\mathsf{Enc2Vote}$ voting system fills over six and a half pages [QS18b, Appendix C.6] and Cortier *et al.* devoted one person-year to their proof of ballot secrecy for Helios [CSD+17]. To reduce the expense of ballot-secrecy proofs, the following section introduces sufficient conditions that enable the simplification of game $\mathsf{Ballot\text{-}Secrecy}$, which gives way to simpler proofs. Indeed, we prove Proposition 5 in just over a page.

4.3 Simplifying proofs

Tallying proofs may reveal voters' votes. For example, a variant of $\mathsf{Enc2Vote}$ might define tallying proofs that map ballots to votes. Hence, such proofs are rightly provided to the adversary in game $\mathsf{Ballot\text{-}Secrecy}$ (Line 7). Nevertheless, if tallying proofs reveal nothing about the votes expressed in ballots on the bulletin board, then they can be omitted from the game. This precondition is ensured by election schemes that use zero-knowledge tallying proofs. Thus, the adversary need not be provided with such proofs in game $\mathsf{Ballot\text{-}Secrecy}$ when analysing such schemes, which achieves our first reduction in the expense of ballot-secrecy proofs. Our second reduction involves modifying the computation of election outcomes.

Game $\mathsf{Ballot\text{-}Secrecy}$ computes the election outcome from ballots constructed by the oracle and ballots constructed by the adversary (Line 6). Intuitively, such an outcome can be equivalently computed as follows:

$(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
$(\mathfrak{v}', pf') \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \cap \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
$\mathfrak{v} \leftarrow \mathfrak{v} + \mathfrak{v}'$;

Yet, a poorly designed tallying algorithm might not ensure equivalence. In particular, ballots constructed by the adversary can cause the algorithm to behave unexpectedly. (Such algorithms are nonetheless compatible with our correctness requirement, because correctness does not consider an adversary.) Nevertheless, the equivalence holds when individual ballots are tallied correctly. Moreover, the above computation is equivalent to the following:

$(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb} \setminus \{b \mid (b, v_0, v_1) \in L\}, nc, \kappa)$;
for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ **do**
  $(\mathfrak{v}', pf') \leftarrow \mathsf{Tally}(sk, \{b\}, nc, \kappa)$;
  $\mathfrak{v} \leftarrow \mathfrak{v} + \mathfrak{v}'$;

Furthermore, by correctness of the election scheme, the above for-loop can be equivalently computed as follows:

for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ **do**
  $\mathfrak{v}[v_\beta] \leftarrow \mathfrak{v}[v_\beta] + 1$;

Indeed, for each $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$, we have b is an output of $\mathsf{Vote}(pk, v_\beta, nc, \kappa)$, hence, $\mathsf{Tally}(sk, \{b\}, nc, \kappa)$ outputs $(\mathfrak{v}, pf)$ such that $\mathfrak{v}$ is a zero-filled vector, except for index v_β which contains one, and this suffices to ensure equivalence. In addition, for any adversary that wins game Ballot-Secrecy, we are assured that $\mathit{balanced}(\mathfrak{bb}, nc, L)$ holds, hence, the above for-loop can be computed as

for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ **do**
  $\mathfrak{v}[v_0] \leftarrow \mathfrak{v}[v_0] + 1$;

or

for $b \in \mathfrak{bb} \wedge (b, v_0, v_1) \in L$ **do**
  $\mathfrak{v}[v_1] \leftarrow \mathfrak{v}[v_1] + 1$;

without weakening the game. Thus, perhaps surprisingly, tallying ballots constructed by the oracle does not provide the adversary with an advantage (in determining whether $\beta = 0$ or $\beta = 1$) and we can omit such ballots from tallying in game Ballot-Secrecy. That is, we need only consider the game derived from Ballot-Secrecy by replacing $\mathcal{A}(\mathfrak{v}, pf)$ with $\mathcal{A}(\mathfrak{v})$ and $\mathit{balanced}(\mathfrak{bb}, nc, L)$ with $b \notin \mathfrak{bb}$, where the former modification captures our first reduction in the expense of ballot-secrecy proofs and the latter captures our second.

Smyth [Smy18a] further reduces the expense of ballot-secrecy proofs by removing the oracle in favour of a single challenge ballot:

Definition 10 (IND-CVA [Smy18a])**.** *Let* $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *be an election scheme,* $\mathcal{A}$ *be an adversary,* κ *be the security parameter, and* IND-CVA$(\Gamma, \mathcal{A}, \kappa)$ *be the following game.*

IND-CVA$(\Gamma, \mathcal{A}, \kappa)$ =

$(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$;
$(v_0, v_1, nc) \leftarrow \mathcal{A}(pk, \kappa)$;
$\beta \leftarrow_R \{0, 1\}$;
$b \leftarrow \mathsf{Vote}(pk, v_\beta, nc, \kappa)$;
$\mathfrak{bb} \leftarrow \mathcal{A}(b)$;
$(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$;
$g \leftarrow \mathcal{A}(\mathfrak{v})$;
return $g = \beta \wedge b \notin \mathfrak{bb} \wedge 1 \leq v_0, v_1 \leq nc \leq mc \wedge |\mathfrak{bb}| \leq mb$;

We say Γ *satisfies* $\mathsf{IND\text{-}CVA}$*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{IND\text{-}CVA}(\Gamma, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$*.*

An election scheme satisfies $\mathsf{IND\text{-}CVA}$ when algorithm Vote outputs non-malleable ballots. Smyth proves that game $\mathsf{Ballot\text{-}Secrecy}$ is strictly stronger than game $\mathsf{IND\text{-}CVA}$, moreover, he proves that the games coincide for election schemes with zero-knowledge tallying proofs that tally individual ballots correctly [Smy18a, Theorems 2 & 5]. Furthermore, he proves that universally-verifiable election schemes tally individual ballots correctly [Smy18a, Lemmata 9 & 28].

Design guideline 4. *Ballots must be non-malleable.*

These results significantly reduce the expense of ballot-secrecy proofs and we now use them to prove Proposition 5.

Proof of Proposition 5. We proceed by contradiction: Suppose election scheme $\mathsf{Enc2Vote}(\Pi)$ does not satisfy $\mathsf{Ballot\text{-}Secrecy}$. Since $\mathsf{Ballot\text{-}Secrecy}$ is strictly stronger than $\mathsf{IND\text{-}CVA}$ [Smy18a, Theorem 2], scheme $\mathsf{Enc2Vote}(\Pi)$ does not satisfy $\mathsf{IND\text{-}CVA}$ either, hence, there exists a probabilistic polynomial-time adversary $\mathcal{A}$ that wins $\mathsf{IND\text{-}CVA}(\mathsf{Enc2Vote}(\Pi), \mathcal{A}, \kappa)$ with success greater than negligibly better than guessing, for some security parameter κ. From $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that wins $\mathsf{IND\text{-}PA0}$.

- $\mathcal{B}(pk, \mathfrak{m}, \kappa)$ computes $pk' \leftarrow (pk, \mathfrak{m}); (v_0, v_1, nc) \leftarrow \mathcal{A}(pk', \kappa)$ and outputs (v_0, v_1).
- $\mathcal{B}(b)$ computes $\mathfrak{bb} \leftarrow \mathcal{A}(b)$, parses $\mathfrak{bb}$ as a set $\{b_1, \dots, b_{|\mathfrak{bb}|}\}$, and outputs vector $(b_1, \dots, b_{|\mathfrak{bb}|})$.
- $\mathcal{B}(\mathbf{m})$ initialises $\mathfrak{v}$ as a zero-filled vector of length nc, computes **for** $1 \leq i \leq |\mathbf{m}|$ **do** $v \leftarrow \mathbf{m}[i]$; **if** $1 \leq v \leq nc$ **then** $\mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1$; $g \leftarrow \mathcal{A}(\mathfrak{v})$ and outputs g.

We prove that the success of adversary $\mathcal{B}$ is equivalent to the success of adversary $\mathcal{A}$, which contradicts our assumption that Π satisfies $\mathsf{IND\text{-}PA0}$.

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and $\mathsf{Enc2Vote}(\Pi) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$. Suppose $(pk, sk, \mathfrak{m})$ is an output of $\mathsf{Gen}(\kappa)$, (v_0, v_1) is an output of $\mathcal{B}(pk, \mathfrak{m}, \kappa)$, and b is an output of $\mathsf{Enc}(pk, v_\beta)$, for some bit β chosen uniformly at random. Moreover, suppose $(b_1, \dots, b_\ell)$ is an output of $\mathcal{B}(b)$. By inspection of algorithms Setup and Vote, and of adversary $\mathcal{B}$, it is trivial to see that $\mathcal{B}$ simulates the challenger in $\mathsf{IND\text{-}CVA}$ to adversary $\mathcal{A}$. Indeed, adversary $\mathcal{B}$ couples public key pk with message space $\mathfrak{m}$, and inputs the resulting pair pk' to $\mathcal{A}$, which corresponds to the public key computed by algorithm Setup, hence, the public key input to $\mathcal{A}$ by the challenger in $\mathsf{IND\text{-}CVA}$. Thus, adversary $\mathcal{A}$ behaves as if playing game $\mathsf{IND\text{-}CVA}$ and output (v_0, v_1) is indistinguishable from outputs that would be observed whilst playing that game. Moreover, since $\mathcal{A}$ wins with success greater than negligibly better than guessing, we have $1 \leq v_0, v_1 \leq nc \leq mc$, furthermore, $\{1, \ldots, nc\} \subseteq \mathfrak{m}$, where mc is the largest integer such that $\{0, \ldots, mc\} \subseteq \{0\} \cup \mathfrak{m}$ and for all $m_0, m_1 \in \{1, \ldots, mc\}$ we have $|m_0| = |m_1|$ (hence, $|v_0| = |v_1|$, which is required to win $\mathsf{IND\text{-}PA0}$), with the same probability. It follows that outputs of $\mathsf{Enc}(pk, v_\beta)$ and $\mathsf{Vote}(pk, v_\beta, nc, \kappa)$ are indistinguishable. Thus, output $(b_1, \ldots, b_\ell)$ is indistinguishable from outputs that would be observed in game $\mathsf{IND\text{-}CVA}$, with the same probability. Moreover, since $\mathcal{A}$ wins, we have $b \notin \{b_1, \ldots, b_\ell\}$, hence, $\bigwedge_{1 \leq i \leq \ell} b \neq b_i$, again with the same probability.

Let $\mathbf{m} = (\mathsf{Dec}(sk, b_1), \ldots, \mathsf{Dec}(sk, b_\ell))$ and suppose g is an output of $\mathcal{B}(\mathbf{m})$. By inspection of algorithm Tally and of adversary $\mathcal{B}$, it is straightforward to see that $\mathcal{B}$ simulates the challenger in $\mathsf{IND\text{-}CVA}$ to adversary $\mathcal{A}$. Indeed, both the algorithm and adversary initialise $\mathfrak{v}$ as a zero-filled vector of length nc, then the adversary $\mathcal{B}$ computes

$$\textbf{for } 1 \leq i \leq |\mathbf{m}| \textbf{ do } v \leftarrow \mathbf{m}[i]; \textbf{ if } 1 \leq v \leq nc \textbf{ then } \mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1;$$

which is equivalent to algorithm Tally computing

$$\textbf{for } b \in \{b_1, \ldots, b_\ell\} \textbf{ do } v \leftarrow \mathsf{Dec}(sk, b); \textbf{ if } 1 \leq v \leq nc \textbf{ then } \mathfrak{v}[v] \leftarrow \mathfrak{v}[v] + 1;$$

because algorithm Dec is deterministic. Thus, output g is indistinguishable from outputs that would be observed in game $\mathsf{IND\text{-}CVA}$. It follows that the success of adversary $\mathcal{B}$ is equivalent to the success of $\mathcal{A}$, and we conclude our proof by [Smy18a, Theorem 5], since Smyth proves that games $\mathsf{IND\text{-}CVA}$ and $\mathsf{Ballot\text{-}Secrecy}$ coincide for election schemes with zero-knowledge tallying proofs that tally individual ballots correctly. (We omit proving that election scheme $\mathsf{Enc2Vote}(\Pi)$ has zero-knowledge tallying proofs and tallies individual ballots correctly to avoid recalling formal definitions of those properties. Proving the former is trivial, because pf is a constant, hence, it reveals nothing about the votes expressed in ballots $b_1, \ldots, b_\ell$. And the latter follows from the definition of algorithm Tally, by correctness of Π, and since Π satisfies $\mathsf{IND\text{-}PA0}$, which is required only to ensure ciphertexts do not collide. Formally proving these details is left as an exercise for the reader.) $\square$

We can exploit Proposition 5 to achieve our fourth and final reduction in the expense of ballot-secrecy proofs. Indeed, if an election scheme tallies sets of ballots correctly (rather than individual ballots, as previously required), then we can compute the election outcome using function *correct-outcome* in game $\mathsf{IND\text{-}CVA}$, rather than the tallying algorithm, i.e., by replacing $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$ with $\mathfrak{v} \leftarrow \textit{correct-outcome}(pk, nc, \mathfrak{bb}, \kappa)$. It follows that election scheme $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ satisfies $\mathsf{Ballot\text{-}Secrecy}$ if and only if $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}', \mathsf{Verify}')$ does, assuming algorithms Tally and Tally' both tally sets of ballots correctly. Proposition 5 proves that election scheme $\mathsf{Enc2Vote}(\Pi)$ satisfies $\mathsf{Ballot\text{-}Secrecy}$, assuming Π is an asymmetric encryption scheme satisfying $\mathsf{IND\text{-}PA0}$, and Smyth [Smy18a, Lemma 14] proves that $\mathsf{Enc2Vote}(\Pi)$ tallies sets of ballots correctly, under the additional assumption that Π satisfies *well-definedness*, i.e., "ill-formed" ciphertexts are distinguishable from "well-formed" ciphertexts. Thus, Ballot-Secrecy is satisfied by any election scheme derived from Enc2Vote(Π) by replacing its tallying and verification algorithms, assuming the replacement tallying algorithm tallies sets of ballots correctly and uses zero-knowledge tallying proofs [Smy18a, Theorem 15]. Moreover, Smyth proves that universally-verifiable election schemes tally sets of ballots correctly [Smy18a, Lemma 28]. It follows that proofs of ballot secrecy are trivial for a class of universally-verifiable, encryption-based voting systems: Any universally-verifiable election scheme derived from Enc2Vote(Π) satisfies Ballot-Secrecy if Π satisfies IND-PA0 and well-definedness, and tallying proofs are zero-knowledge.

We will use our third simplification to prove that a variant of Helios satisfies Ballot-Secrecy and the fourth to prove that a variant of Helios Mixtnet does too (the original schemes have known vulnerabilities and they do not satisfy Ballot-Secrecy), thereby demonstrating the application of these results.

4.4 Related definitions of ballot secrecy

Discussion of ballot secrecy originates from Chaum [Cha81] and the earliest definitions of ballot secrecy are due to Benaloh *et al.* [BY86,BT94b,Ben96]. More recently, Bernhard *et al.* propose a series of ballot secrecy definitions [BPW12, SB13, SB14, BCG+15]. Smyth [Smy18a] shows that these definitions do not detect vulnerabilities that arise when an adversary controls the bulletin board or the communication channel. By comparison, the definition of ballot secrecy that we consider (Definition 8) detects such vulnerabilities and appears to be the strongest definition in the literature.

Beyond the computational model of security, Delaune, Kremer & Ryan formulate a definition of ballot secrecy in the applied pi calculus [DKR09] and Smyth *et al.* show that this definition is amenable to automated reasoning [DRS08, Smy11, BS16, BS17]. An alternative definition is proposed by Cremers & Hirschi, along with sufficient conditions which are also amenable to automated reasoning [CH17]. Albeit, the scope of automated reasoning is limited by analysis tools (e.g., ProVerif [BSCS16]), because the function symbols and equational theory used to model cryptographic primitives might not be suitable for automated analysis (cf. [DKRS11, PB12, ABR12]).

4.5 Further notions of privacy

Ballot secrecy formalises a notion of free-choice assuming ballots are constructed and tallied in the prescribed manner. Moreover, Smyth's definition assumes the adversary's capabilities are limited to casting ballots on behalf of some voters and controlling the distribution of votes cast by the remaining voters. We have seen that voting system Enc2Vote satisfies this definition, but ballot secrecy does not ensure free-choice when adversaries are able to communicate with voters nor when voters deviate from the prescribed voting procedure to follow instructions provided by adversaries. Indeed, the coins used for encryption serve as proof of how a voter voted in Enc2Vote and the voter may communicate those coins to the adversary. Stronger notions of free-choice, such as receipt-freeness [MN06, KZZ15, CCFG16] and coercion resistance [JCJ05, GGR09, UM10, KTV12a], are needed in the presence of such adversaries.

Ballot secrecy does not provide assurances when deviations from the prescribed tallying procedure are possible. Indeed, ballots can be tallied individually to reveal votes. Hence, the tallier must be trusted. Alternatively, we can design election schemes that distribute the tallier's role amongst several talliers and ensure free-choice assuming at least one tallier tallies ballots in the prescribed manner. Extending results in this direction is an opportunity for future work. Ultimately, we would prefer not to trust talliers. Unfortunately, this is only known to be possible for decentralised voting systems, e.g., [Sch99,KY02,Gro04,HRZ10,KSRH12], which are designed such that ballots cannot be individually tallied, but are unsuitable for large-scale elections.

5 Case study I: Helios

Helios can be informally modelled as the following election scheme:

Setup generates a key pair for an asymmetric additively-homomorphic encryption scheme, proves correct key generation in zero-knowledge, and outputs the key pair and proof.

Vote enciphers the vote's bitstring encoding to a tuple of ciphertexts, proves in zero-knowledge that each ciphertext is correctly constructed and that the vote is selected from the sequence of candidates, and outputs the ciphertexts coupled with the proofs. (Figure 2 provides further details.)

Tally selects ballots from the bulletin board for which proofs hold, homomorphically combines the ciphertexts in those ballots, decrypts the homomorphic combination to reveal the election outcome, and announces the outcome, along with a zero-knowledge proof of correct decryption. (Figure 2 provides further details.)

Verify checks the proofs and accepts the distribution if these checks succeed.

Helios was first released in 2009 as *Helios 2.0*, the current release is *Helios 3.1.4*, and a new release is planned.[9] Henceforth, we'll refer to the planned release as *Helios'12*.

5.1 Helios 2.0

Cortier & Smyth show that Helios 2.0 does not satisfy ballot secrecy [CS13, CS11]. Thus, we would not expect our definition of ballot secrecy to hold. Indeed, Smyth [Smy18a] adopts the formal description of Helios 2.0 by Smyth,

[9] `http://documentation.heliosvoting.org/verification-specs/helios-v4`, published c. 2012, accessed 21 Sep 2017.

Figure 2 Helios: Ballot construction and tallying

Algorithm Vote inputs a vote v selected from candidates $1, \dots, nc$ and computes ciphertexts $c_1, \dots, c_{nc-1}$ such that if $v < nc$, then ciphertext c_v contains plaintext 1 and the remaining ciphertexts contain plaintext 0, otherwise, all ciphertexts contain plaintext 0. The algorithm also computes proofs $\sigma_1, \dots, \sigma_{nc}$ demonstrating correct computation. Proof σ_j demonstrates that ciphertext c_j contains 0 or 1, where $1 \leq j \leq nc - 1$. And proof σ_{nc} demonstrates that the homomorphic combination of ciphertexts $c_1 \otimes \cdots \otimes c_{nc-1}$ contains 0 or 1. The algorithm outputs the ciphertexts and proofs.

Algorithm Tally inputs a bulletin board $\mathfrak{bb}$; selects all the ballots $b_1, \dots, b_k \in \mathfrak{bb}$ for which proofs hold, i.e., ballots $b_i = \mathsf{Enc}(pk, m_{i,1}), \dots, \mathsf{Enc}(pk, m_{i,nc-1})$, $\sigma_{i,1}, \dots, \sigma_{i,nc}$ such that proofs $\sigma_{i,1}, \dots, \sigma_{i,nc}$ hold, where $1 \leq i \leq k$; forms a matrix of the encapsulated ciphertexts, i.e.,

$$\begin{array}{ccc} \mathsf{Enc}(pk, m_{1,1}), & \dots, & \mathsf{Enc}(pk, m_{1,nc-1}) \\ \vdots & & \vdots \\ \mathsf{Enc}(pk, m_{k,1}), & \dots, & \mathsf{Enc}(pk, m_{k,nc-1}); \end{array}$$

homomorphically combines the ciphertexts in each column to derive the encrypted outcome, i.e.,

$$\mathsf{Enc}(pk, \Sigma_{i=1}^{k} m_{i,1}), \quad \dots, \quad \mathsf{Enc}(pk, \Sigma_{i=1}^{k} m_{i,nc-1});$$

decrypts the homomorphic combinations to reveal the frequency of votes $1, \dots, nc - 1$, i.e.,

$$\Sigma_{i=1}^{k} m_{i,1}, \quad \dots, \quad \Sigma_{i=1}^{k} m_{i,nc-1};$$

computes the frequency of vote nc by subtracting the frequency of any other vote from the number of ballots for which proofs hold, i.e., $k - \sum_{j=1}^{nc-1} \sum_{i=1}^{k} m_{i,j}$; and announces the outcome as those frequencies, along with a proof demonstrating correctness of decryption.

Frink & Clarkson [SFC17] and uses that description to prove that $\mathsf{Ballot\text{-}Secrecy}$ is not satisfied.

Theorem 6. *Helios 2.0 does not satisfy* $\mathsf{Ballot\text{-}Secrecy}$.

Cortier & Smyth attribute the vulnerability to tallying meaningfully related ballots. Indeed, Helios uses malleable ballots: Given a ballot $c_1, \dots, c_{nc-1}$, $\sigma_1, \dots, \sigma_{nc}$, we have $c_{\chi(1)}, \dots, c_{\chi(nc-1)}, \sigma_{\chi(1)}, \dots, \sigma_{\chi(nc-1)}, \sigma_{nc}$ is a ballot for all permutations χ on $\{1, \dots, nc - 1\}$. Thus, ballots are malleable, which is incompatible with ballot secrecy (§4.3).

Proof sketch. Suppose an adversary queries the oracle with inputs v_0 and v_1 to derive a ballot for v_β, where bit β is chosen by the challenger. Further suppose the adversary abuses malleability to derive a related ballot b for v_β

and outputs bulletin board $\{b\}$. The board is balanced, because it does not contain the ballot output by the oracle. Suppose the adversary performs the following computation on input of election outcome $\mathfrak{v}$: if $\mathfrak{v}[v_0] = 1$, then output 0, otherwise, output 1. Since b is a ballot for v_β, it follows by correctness that $\mathfrak{v}[v_0] = 1$ iff $\beta = 0$, and $\mathfrak{v}[v_1] = 1$ iff $\beta = 1$, hence, the adversary wins the game. □

For simplicity, the proof sketch considers an adversary that omits ballots from the bulletin board. Voters might detect such an adversary, because Helios satisfies individual verifiability, hence, voters can discover if their ballot is omitted. The proof sketch can be extended to avoid such detection: Let b_1 be the ballot output by the oracle in the proof sketch and suppose b_2 is the ballot output by a (second) oracle query with inputs v_1 and v_0. Further suppose the adversary outputs (the balanced) bulletin board $\{b, b_1, b_2\}$ and performs the following computation on input of election outcome $\mathfrak{v}$: if $\mathfrak{v}[v_0] = 2$, then output 0, otherwise, output 1. Hence, the adversary wins the game.

Chang-Fong & Essex show that Helios 2.0 does not satisfy universal verifiability [CE16, §4.1], and Smyth, Frink & Clarkson use their result to prove that the completeness aspect of Universal-Verifiability is not satisfied [SFC17].[10]

Theorem 7. *Helios 2.0 does not satisfy* Completeness.

Chang-Fong & Essex attribute the vulnerability to not checking the suitability of cryptographic parameters nor checking that ballots are constructed from such parameters.

Proof sketch. Suppose an adversary computes a ciphertext and masks a term of that ciphertext. Moreover, suppose the adversary falsifies a proof of correct construction in a manner that hides malice. In particular, the adversary computes the proof such that an exponent will evaluate to zero during verification, which causes cancellation of the mask. (This is possible because verification does not check that ballots are constructed from suitable cryptographic parameters.) Suppose the adversary computes a bulletin board containing the masked ciphertext and proof. Moreover, suppose that the challenger tallies that board. The masked ciphertext will be homomorphically combined with other ciphertexts and decrypted, because the proof holds. Yet, the prove of correct decryption constructed by the challenger will fail, due to the masked ciphertext, hence, the adversary wins the game. □

The vulnerability was mitigated against in Helios 3.1.4 by performing the necessary checks.

[10] Chang-Fong & Essex present a vulnerability [CE16, §4.2] that should violate Soundness, proving this result is left as an exercise for the reader.

5.2 Helios 3.1.4

Ballots remain malleable in Helios 3.1.4, hence, ballot secrecy is not satisfied, and Smyth [Smy18a] proves that Ballot-Secrecy is not satisfied, using the formal description of Helios 3.1.4 by Smyth, Frink & Clarkson [SFC17].

Corollary 8. *Helios 3.1.4 does not satisfy* Ballot-Secrecy.

A proof of Corollary 8 follows from Theorem 6, because Helios 3.1.4 does not address issues arising from related ballots.

Bernhard, Pereira & Warinschi show that Helios 3.1.4 does not satisfy universal verifiability [BPW12, §3], and Smyth, Frink & Clarkson use their result to prove that the soundness aspect of Universal-Verifiability is not satisfied.[11]

Theorem 9. *Helios 3.1.4 does not satisfy* Soundness.

Bernhard *et al.* attribute vulnerabilities to application of the Fiat-Shamir transformation without inclusion of statements in hashes (i.e., weak Fiat-Shamir).

Proof sketch. Suppose an adversary partially computes a proof of ciphertext construction, before computing a ciphertext and without computing a key pair. In particular, suppose the adversary computes the challenge hash. (This is possible because weak Fiat-Shamir does not include statements in hashes, hence, ciphertexts are not included in hashes.) Further suppose the adversary computes a private key as a function of that hash, challenges as functions of the hash and the private key, and responses as functions of the challenges and some coins. Moreover, suppose the adversary computes a public key (from the private key) and a proof of correct key generation. That proof is valid, because the private key could have been correctly computed. Suppose the adversary enciphers some plaintext m (such that $m > 1$) to a ciphertext, using the aforementioned coins. Further suppose the adversary proves correct decryption of that ciphertext. That proof is valid, because the ciphertext is well-formed. Finally, suppose the adversary claims $(m, m-1)$ is the election outcome corresponding to the ballot containing the ciphertext and falsified proof of correct construction. The verification procedure will accept that outcome, because all proofs hold, yet the election outcome is clearly invalid, hence, the adversary wins the game. □

5.3 Helios'12

Helios'12 is intended to mitigate against vulnerabilities. In particular, the specification incorporates the Fiat-Shamir transformation (rather than weak Fiat-Shamir), and there are plans to incorporate *ballot weeding*, i.e., to omit meaningfully related ballots from tallying. Smyth, Frink & Clarkson show that Helios'12

[11] Bernhard, Pereira & Warinschi present a vulnerability [CE16, p632] that should violate Completeness, proving this result is left as an exercise for the reader.

does not satisfy universal verifiability [SFC17], and Smyth shows that ballot secrecy is not satisfied either [Smy18a].[12]

Remark 10. *Helios'12 does not satisfy* Soundness.

Proof sketch. Suppose an adversary constructs a ballot. Further suppose the adversary abuses malleability to derive a related ballot. Moreover, suppose those ballots are tallied by the adversary. Ballot weeding will omit at least one of those ballots. (Helios'12 does not yet define a particular ballot weeding mechanism, hence, the precise behaviour is unknown. Nonetheless, we are assured that at least one ballot will be omitted, because the ballots are related.) Hence, tallying produces an election outcome that omits a vote, which soundness forbids, thus, the adversary wins the game. □

Remark 11. *Helios'12 does not satisfy* Ballot-Secrecy.

Proof sketch. Neither ballot weeding nor the Fiat-Shamir transformation eliminate the vulnerability we identified in Helios 3.1.4, because related ballots need not be tallied (as shown in the proof sketch of Theorem 6). Hence, we conclude by Corollary 8. □

5.4 Helios'16

Smyth, Frink & Clarkson [SFC17] propose Helios'16, a variant of Helios that uses the Fiat-Shamir transformation and non-malleable ballots, to overcome the aforementioned vulnerabilities. They prove that Helios'16 satisfies verifiability, and Smyth [Smy18a] proves that ballot secrecy is satisfied too.

Theorem 12. *Helios'16 satisfies both* Individual-Verifiability *and* Universal-Verifiability.

Proof sketch. Smyth *et al.* [SFC17, Smy18c] prove that El Gamal produces ciphertexts that do not collide for correctly generated keys. Hence, Helios'16 ballots do not collide, because they contain El Gamal ciphertexts constructed using such keys. Thus, Helios'16 satisfies Individual-Verifiability. Smyth, Frink & Clarkson also prove that Universal-Verifiability is satisfied. Their proof shows that tallying discards ill-formed ballots and that the remaining ballots all contain ciphertexts that encipher bitstring encodings of votes, hence, the homomorphic combination of those ciphertexts contain the encrypted outcome, which is decrypted to reveal the correct outcome (Soundness). Moreover, they show that such outcomes are always accepted (Completeness). □

Theorem 13. *Helios'16 satisfies* Ballot-Secrecy.

[12]Remarks 10 & 11 are stated informally, because there is no formal description of Helios'12. Such a description can be derived as a straightforward variant of Helios 3.1.4 that uses ballot weeding and applies the Fiat-Shamir transformation (rather than the weak Fiat-Shamir transformation). But, these details provide little value, so we do not pursue them.

	Helios 2.0	Helios 3.1.4	Helios'12	Helios'16
Ballot secrecy	✗	✗	✗	✓
Individual verifiability	✓	✓	✓	✓
Universal verifiability	✗	✗	✗	✓

Cortier & Smyth identify a secrecy vulnerability in Helios 2.0 and Helios 3.1.4 [CS13], and Smyth shows the vulnerability is exploitable in Helios'12 when the adversary controls ballot collection [Smy18a]. Moreover, Smyth proves that Helios'16 satisfies ballot secrecy. Bernhard, Pereira & Warinschi identify universal-verifiability vulnerabilities in Helios 2.0 and Helios 3.1.4 [BPW12], Chang-Fong & Essex identify vulnerabilities in Helios 2.0 [CE16], and Smyth, Frink, & Clarkson identify a vulnerability in Helios'12 [SFC17]. Moreover, Smyth, Frink, & Clarkson prove that Helios'16 satisfies individual and universal verifiability.

Table 1: Summary of Helios security results

Proof sketch. Smyth proves that Helios'16 satisfies IND-CVA [Smy18a, Proposition 25] and Smyth, Frink & Clarkson prove that Universal-Verifiability is satisfied too (Theorem 12), moreover, Smyth proves that Helios'16 uses zero-knowledge tallying proofs, which suffices for Ballot-Secrecy (§4.3). □

These results (summarised in Table 1) provide strong motivation for future Helios releases being based upon Helios'16, since it is the only variant of Helios which is proven to satisfy both ballot secrecy and verifiability.[13]

6 Case study II: Helios Mixnet

Helios Mixnet can be informally modelled as the following election scheme:

Setup generates a key pair for an asymmetric homomorphic encryption scheme, proves correct key generation in zero-knowledge, and outputs the key pair and proof.

Vote enciphers the vote to a ciphertext, proves correct ciphertext construction in zero-knowledge, and outputs the ciphertext coupled with the proof.

Tally selects ballots from the bulletin board for which proofs hold, mixes the ciphertexts in those ballots, decrypts the ciphertexts output by the mix to reveal the election outcome (i.e., the distribution of votes) and any ill-formed votes (i.e., votes that are not selected from the sequence of candidates), and announces that outcome, along with zero-knowledge proofs demonstrating correct decryption.

Verify checks the proofs and accepts the distribution if these checks succeed.

Neither Adida [Adi08] nor Bulens, Giry & Pereira [BGP11] have released an implementation of Helios Mixnet. Tsoukalas *et al.* [TPLT13] released *Zeus* as a fork of Helios spliced with mixnet code to derive an implementation, and

[13] Beyond secrecy and verifiability, eligibility is known not to be satisfied [SP13,SP15,MS17].

Yingtong Li released *helios-server-mixnet* as an extension of Zeus with threshold asymmetric encryption and some other minor changes.

We have seen that Helios 2.0 does not satisfy completeness (Theorem 7), hence, implementations of Helios Mixnet did not satisfy completeness until Helios was patched (because the implementations fork Helios and do not add code to check cryptographic parameters). Moreover, Smyth [Smy18b, Smy18c] identifies a soundness vulnerability in Helios Mixnet.

Remark 14. *Zeus does not satisfy* Soundness*.*

Smyth attributes the vulnerability to the weak Fiat-Shamir transformation.

Proof sketch. Suppose an adversary constructs some ballots and mixes the ciphertexts in those ballots. Further suppose the adversary decrypts the ciphertexts output by the mix to reveal the distribution of votes, selects some ciphertexts that decrypt to a (strict) subdistribution, proves correct decryption of those ciphertexts, and falsifies proofs that the remaining ciphertexts decrypt to arbitrary elements of the message space (by exploiting a vulnerability against Helios [BPW12], which exists due to the weak Fiat-Shamir transformation). Finally, suppose the adversary claims the subdistribution of votes is the election outcome. The verification procedure will accept that outcome, because all proofs hold, yet the election outcome excludes votes, hence, the adversary wins the game. □

Similarly, voting system helios-server-mixnet does not satisfy Soundness when a (n, n)-threshold is used [Smy18b, Smy18c].

Smyth proposes a formal description of Helios Mixnet that uses the Fiat-Shamir transformation and proves that Ballot-Secrecy, Individual-Verifiability, and Universal-Verifiability are satisfied [Smy18a, Smy18c].

Theorem 15. *Helios Mixnet satisfies both* Individual-Verifiability *and* Universal-Verifiability*.*

Proof sketch. As per Helios (Theorem 12), ballots do not collide, because they contain El Gamal ciphertexts constructed using correctly generated keys, hence, Individual-Verifiability is satisfied. Moreover, Universal-Verifiability is satisfied too, because tallying discards ill-formed ballots and votes, hence, the mix outputs the correct outcome (Soundness), and such outcomes are always accepted (Completeness). □

Theorem 16. *Helios Mixnet satisfies* Ballot-Secrecy*.*

Proof sketch. Smyth [Smy18c,Smy18b] shows that Helios Mixnet can be derived from Enc2Vote(Π) using suitable tallying and verification algorithms, moreover, he proves that Universal-Verifiability is satisfied, which suffices for Ballot-Secrecy (§4.3), assuming Π satisfies IND-PA0 and well-definedness. □

Smyth reported these findings to the developers of Zeus and helios-server-mixnet, who promptly adopted and deployed the proposed fix [Smy18c, §4].

7 Reflection

We have studied definitions of secrecy and verifiability, and shown how these definitions can be used to detect subtle vulnerabilities in the Helios and Helios Mixnet voting systems. Moreover, we have seen how analysis drives development and ultimately leads to systems that are proven secure. Thereby demonstrating the necessity of security definitions and accompanying analysis to ensure security of voting systems, especially those used in binding elections. I hope this manuscript advances the reader's understanding of these matters and, ultimately, aids democracy-builders in deploying their systems securely.

A Asymmetric encryption

Definition 11 (Asymmetric encryption scheme [KL07, SFC17]). *An* asymmetric encryption scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, such that:*

- **Gen***, denoted* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa)$*, inputs a security parameter* κ *and outputs a key pair* (pk, sk) *and message space* $\mathfrak{m}$*.*
- **Enc***, denoted* $c \leftarrow \mathsf{Enc}(pk, m)$*, inputs a public key* pk *and message* $m \in \mathfrak{m}$*, and outputs a ciphertext* c*.*
- **Dec***, denoted* $m \leftarrow \mathsf{Dec}(sk, c)$*, inputs a private key* sk *and ciphertext* c*, and outputs a message* m *or an error symbol. We assume* Dec *is deterministic.*

Moreover, the scheme must be correct*: there exists a negligible function* negl*, such that for all security parameters* κ *and messages* m*, we have* $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa); c \leftarrow \mathsf{Enc}(pk, m) : m \in \mathfrak{m} \Rightarrow \mathsf{Dec}(sk, c) = m] > 1 - \mathsf{negl}(\kappa)$*.*

Definition 12 (IND-PA0 [BS99]). *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an asymmetric encryption scheme,* $\mathcal{A}$ *be an adversary,* κ *be the security parameter, and* $\mathsf{IND\text{-}PA0}(\Pi, \mathcal{A}, \kappa)$ *be the following game.*

$\mathsf{IND\text{-}PA0}(\Pi, \mathcal{A}, \kappa) =$

$(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa)$;
$(m_0, m_1) \leftarrow \mathcal{A}(pk, \mathfrak{m}, \kappa)$;
$\beta \leftarrow_R \{0, 1\}$;
$c \leftarrow \mathsf{Enc}(pk, m_\beta)$;
$\mathbf{c} \leftarrow \mathcal{A}(c)$;
$\mathbf{m} \leftarrow (\mathsf{Dec}(sk, \mathbf{c}[1]), \ldots, \mathsf{Dec}(sk, \mathbf{c}[|\mathbf{c}|]))$;
$g \leftarrow \mathcal{A}(\mathbf{m})$;
return $g = \beta \wedge \bigwedge_{1 \leq i \leq |\mathbf{c}|} c \neq \mathbf{c}[i]$;

In the above game, we require $m_0, m_1 \in \mathfrak{m}$ *and* $|m_0| = |m_1|$*. We say* Π *satisfies* indistinguishability under parallel attack ($\mathsf{IND\text{-}PA0}$)*, if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl*, such that for all security parameters* κ*, we have* $\mathsf{Succ}(\mathsf{IND\text{-}PA0}(\Pi, \mathcal{A}, \kappa)) \leq \frac{1}{2} + \mathsf{negl}(\kappa)$*.*

References

[ABR12] Myrto Arapinis, Sergiu Bursuc, and Mark Ryan. Reduction of Equational Theories for Verification of Trace Equivalence: Re-encryption, Associativity and Commutativity. In *POST'12: First Conference on Principles of Security and Trust*, volume 7215 of *LNCS*, pages 169–188. Springer, 2012.

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

[AH10] R. Michael Alvarez and Thad E. Hall. *Electronic Elections: The Perils and Promises of Digital Democracy*. Princeton University Press, 2010.

[AMPQ09] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In *EVT/WOTE'09: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX Association, 2009.

[AN06] Ben Adida and C. Andrew Neff. Ballot casting assurance. In *EVT'06: Electronic Voting Technology Workshop*. USENIX Association, 2006.

[BBP07] Romain Bertrand, Jean-Louis Briquet, and Peter Pels. Introduction: Towards a Historical Ethnography of Voting. In *The Hidden History of the Secret Ballot*. Indiana University Press, 2007.

[BCG+15] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. SoK: A comprehensive analysis of game-based ballot privacy definitions. In *S&P'15: 36th Security and Privacy Symposium*, pages 499–516. IEEE Computer Society, 2015.

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

[BGP11] Philippe Bulens, Damien Giry, and Olivier Pereira. Running Mixnet-Based Elections with Helios. In *EVT/WOTE'11: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX Association, 2011.

[Bjo04] Eric C. Bjornlund. *Beyond Free and Fair: Monitoring Elections and Building Democracy*. Woodrow Wilson Center Press / Johns Hopkins University Press, 2004.

[Bow07] Debra Bowen. Secretary of State Debra Bowen Moves to Strengthen Voter Confidence in Election Security Following Top-to-Bottom Review of Voting Systems. California Secretary of State, press release DB07:042, August 2007.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *ASIACRYPT'12: 18th International Conference on the Theory and Application of Cryptology and Information Security*, volume 7658 of *LNCS*, pages 626–643. Springer, 2012.

[Bre06] Peter Brent. The Australian ballot: Not the secret ballot. *Australian Journal of Political Science*, 41(1):39–50, 2006.

[BS99] Mihir Bellare and Amit Sahai. Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *CRYPTO'99: 19th International Cryptology Conference*, volume 1666 of *LNCS*, pages 519–536. Springer, 1999.

[BS16] Bruno Blanchet and Ben Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. In *CSF'16: 29th Computer Security Foundations Symposium*, pages 310–324. IEEE Computer Society, 2016.

[BS17] Bruno Blanchet and Ben Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. *Journal of Computer Security*, 2017. To appear.

[BSCS16] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. *ProVerif 1.96: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, 2016.

[BT94a] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC '94: Twenty-sixth Annual ACM Symposium on Theory of Computing*, pages 544–553, New York, NY, USA, 1994. ACM Press.

[BT94b] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *STOC'94: 26th Theory of computing Symposium*, pages 544–553. ACM Press, 1994.

[BVQ10] Josh Benaloh, Serge Vaudenay, and Jean-Jacques Quisquater. Final Report of IACR Electronic Voting Committee. International Association for Cryptologic Research. `http://www.iacr.org/elections/eVoting/finalReportHelios_2010-09-27.html`, Sept 2010.

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

[CCC+08] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *EVT'08: Electronic Voting Technology Workshop*. USENIX Association, 2008.

[CCFG16] Pyrros Chaidos, Véronique Cortier, Georg Fuschbauer, and David Galido. BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. In *CCS'16: 23rd ACM Conference on Computer and Communications Security*, pages 1614–1625. ACM Press, 2016.

[CE16] Nicholas Chang-Fong and Aleksander Essex. The Cloudier Side of Cryptographic End-to-end Verifiable Voting: A Security Analysis of Helios. In *ACSAC'16: 32nd Annual Conference on Computer Security Applications*, pages 324–335. ACM Press, 2016.

[CF85] Josh Daniel Cohen and Michael J. Fischer. A Robust and Verifiable Cryptographically Secure Election Scheme. In *FOCS'85: 26th Symposium on Foundations of Computer Science*, pages 372–382. IEEE Computer Society, 1985.

[CGGI14] Véronique Cortier, David Galindo, Stephane Glondu, and Malika Izabachène. Election Verifiability for Helios under Weaker Trust Assumptions. In *ESORICS'14: 19th European Symposium on Research in Computer Security*, volume 8713 of *LNCS*, pages 327–344. Springer, 2014.

[CGK+16] Veronique Cortier, David Galindo, Ralf Küsters, Johannes Mueller, and Tomasz Truderung. SoK: Verifiability Notions for E-Voting Protocols. In *S&P'16: 37th IEEE Symposium on Security and Privacy*, pages 779–798. IEEE Computer Society, 2016.

[CH17] Cas Cremers and Lucca Hirschi. Improving Automated Symbolic Analysis for E-voting Protocols: A Method Based on Sufficient Conditions for Ballot Secrecy. arXiv, Report 1709.00194, 2017.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–90, 1981.

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A Practical Voter-Verifiable Election Scheme. In *ESORICS'05: 10th European Symposium On Research In Computer Security*, volume 3679 of *LNCS*, pages 118–139. Springer, 2005.

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

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

[CSD+17] Véronique Cortier, Benedikt Schmidt, Constantin Cătălin Drăgan, Pierre-Yves Strub, Francois Dupressoir, and Bogdan Warinschi. Machine-Checked Proofs of Privacy for Electronic Voting Protocols. In *S&P'17: 37th IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2017.

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[DKRS11] Stéphanie Delaune, Steve Kremer, Mark D. Ryan, and Graham Steel. Formal analysis of protocols based on TPM state registers. In *CSF'11: 24th Computer Security Foundations Symposium*, pages 66–80. IEEE Computer Society, 2011.

[DRS08] Stéphanie Delaune, Mark D. Ryan, and Ben Smyth. Automatic verification of privacy properties in the applied pi-calculus. In *IFIPTM'08: 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, volume 263 of *International Federation for Information Processing (IFIP)*, pages 263–278. Springer, 2008.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques*, volume 718 of *LNCS*, pages 244–251. Springer, 1992.

[Ger09] Bundesverfassungsgericht (Germany's Federal Constitutional Court). *Use of voting computers in 2005 Bundestag election unconstitutional*, March 2009. Press release 19/2009.

[GGR09] Ryan W. Gardner, Sujata Garera, and Aviel D. Rubin. Coercion Resistant End-to-end Voting. In *FC'09: 13th International Conference on Financial Cryptography and Data Security*, volume 5628 of *LNCS*, pages 344–361. Springer, 2009.

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

[Gro04] Jens Groth. Efficient maximal privacy in boardroom voting and anonymous broadcast. In *FC'04: 8th International Conference on Financial Cryptography*, volume 3110 of *LNCS*, pages 90–104. Springer, 2004.

[Gum05] Andrew Gumbel. *Steal This Vote: Dirty Elections and the Rotten History of Democracy in America*. Nation Books, 2005.

[HBH10] Stuart Haber, Josh Benaloh, and Shai Halevi. The Helios e-Voting Demo for the IACR. International Association for Cryptologic Research. `http://www.iacr.org/elections/eVoting/heliosDemo.pdf`, May 2010.

[HRZ10] Fao Hao, Peter Y. A. Ryan, and Piotr Zieliński. Anonymous voting by two-round public discussion. *Journal of Information Security*, 4(2):62 – 67, 2010.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In *WPES'05: 4th Workshop on Privacy in the Electronic Society*, pages 61–70. ACM Press, 2005.

[JCJ10] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In David Chaum, Markus Jakobsson, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Towards Trustworthy Elections: New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 37–63. Springer, 2010.

[JS12] Douglas W. Jones and Barbara Simons. *Broken Ballots: Will Your Vote Count?*, volume 204 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford University, 2012.

[Kel12] Judith G. Kelley. *Monitoring Democracy: When International Election Observation Works, and Why It Often Fails*. Princeton University Press, 2012.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.

[KRS10] Steve Kremer, Mark D. Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *ESORICS'10: 15th European Symposium on Research in Computer Security*, volume 6345 of *LNCS*, pages 389–404. Springer, 2010.

[KSRH12] Dalia Khader, Ben Smyth, Peter Y. A. Ryan, and Feng Hao. A Fair and Robust Voting System by Broadcast. In *EVOTE'12: 5th International Conference on Electronic Voting*, volume 205 of *Lecture Notes in Informatics*, pages 285–299. Gesellschaft für Informatik, 2012.

[KSRW04] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an Electronic Voting System. In *S&P'04: 25th Security and Privacy Symposium*, pages 27–40. IEEE Computer Society, 2004.

[KTV10] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. In *CCS'10: 17th ACM Conference on Computer and Communications Security*, pages 526–535. ACM Press, 2010.

[KTV11] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In *S&P'11: 32nd IEEE Symposium on Security and Privacy*, pages 538–553. IEEE Computer Society, 2011.

[KTV12a] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A Game-Based Definition of Coercion-Resistance and its Applications. *Journal of Computer Security*, 20(6):709–764, 2012.

[KTV12b] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Clash Attacks on the Verifiability of E-Voting Systems. In *S&P'12: 33rd IEEE Symposium on Security and Privacy*, pages 395–409. IEEE Computer Society, 2012.

[KTV15] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. Cryptology ePrint Archive, Report 2010/236 (version 20150202:163211), 2015.

[KY02] Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In *PKC'01: 3rd International Workshop on Practice and Theory in Public Key Cryptography*, volume 2274 of *LNCS*, pages 141–158. Springer, 2002.

[KZZ15] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *EUROCRYPT'15: 34th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 9057 of *LNCS*, pages 468–498. Springer, 2015.

[Lep08] Jill Lepore. Rock, Paper, Scissors: How we used to vote. *Annals of Democracy, The New Yorker*, October 2008.

[LG84] Arend Lijphart and Bernard Grofman. *Choosing an electoral system: Issues and Alternatives*. Praeger, 1984.

[Mil30] James Mill. The Ballot. In *The Westminster Review*, volume 13. Robert Heward, 1830.

[MN06] Tal Moran and Moni Naor. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In *CRYPTO'06: 26th International Cryptology Conference*, volume 4117 of *LNCS*, pages 373–392. Springer, 2006.

[MS17] Maxime Meyer and Ben Smyth. An attack against the helios election system that exploits re-voting. arXiv, Report 1612.04099, 2017.

[NA03] C. Andrew Neff and Jim Adler. Verifiable e-Voting: Indisputable electronic elections at polling places. Technical report, VoteHere, 2003.

[Nef04] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. Technical report, VoteHere, 2004.

[Nor15] Pippa Norris. *Why Elections Fail*. Cambridge University Press, 2015.

[OAS69] Organization of American States. *American Convention on Human Rights, "Pact of San Jose, Costa Rica"*, 1969.

[OSC90] Organization for Security and Co-operation in Europe. *Document of the Copenhagen Meeting of the Conference on the Human Dimension of the CSCE*, 1990.

[PB12] Miriam Paiola and Bruno Blanchet. Verification of Security Protocols with Lists: From Length One to Unbounded Length. In *POST'12: First Conference on Principles of Security and Trust*, volume 7215 of *LNCS*, pages 69–88. Springer, 2012.

[QS18a] Elizabeth A. Quaglia and Ben Smyth. Authentication with weaker trust assumptions for voting systems. In *AFRICACRYPT'17: 9th International Conference on Cryptology in Africa*, LNCS. Springer, 2018.

[QS18b] Elizabeth A. Quaglia and Ben Smyth. Secret, verifiable auctions from elections. Cryptology ePrint Archive, Report 2015/1204, 2018.

[RS07] Ronald L. Rivest and Warren D. Smith. Three Voting Protocols: ThreeBallot, VAV, and Twin. In *EVT'07: Electronic Voting Technology Workshop*. USENIX Association, 2007.

[Saa95] Thomas Saalfeld. On Dogs and Whips: Recorded Votes. In Herbert Döring, editor, *Parliaments and Majority Rule in Western Europe*, chapter 16. St. Martin's Press, 1995.

[SB13] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. In *ESORICS'13: 18th European Symposium on Research in Computer Security*, volume 8134 of *LNCS*, pages 463–480. Springer, 2013.

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

[Sch99] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *CRYPTO'99: 19th International Cryptology Conference*, volume 1666 of *LNCS*, pages 148–164. Springer, 1999.

[Sch05] Nicole Schweikardt. Arithmetic, first-order logic, and counting quantifiers. *ACM Transactions on Computational Logic*, 6(3):634–671, July 2005.

[SFC17] Ben Smyth, Steven Frink, and Michael R. Clarkson. Election Verifiability: Cryptographic Definitions and an Analysis of Helios and JCJ. Cryptology ePrint Archive, Report 2015/233 (version 20170213:132559), 2017.

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

[SK95] Kazue Sako and Joe Kilian. Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth. In *EUROCRYPT'95: 12th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 921 of *LNCS*, pages 393–403. Springer, 1995.

[Smy11] Ben Smyth. *Formal verification of cryptographic protocols with automated reasoning*. PhD thesis, School of Computer Science, University of Birmingham, 2011.

[Smy17] Ben Smyth. First-past-the-post suffices for ranked voting. `https://bensmyth.com/publications/2017-FPTP-suffices-for-ranked-voting/`, 2017.

[Smy18a] Ben Smyth. Ballot secrecy: Security definition, sufficient conditions, and analysis of Helios. Cryptology ePrint Archive, Report 2015/942, 2018.

[Smy18b] Ben Smyth. Verifiability of Helios Mixnet. In *Voting'18: 3rd Workshop on Advances in Secure Electronic Voting*, LNCS. Springer, 2018.

[Smy18c] Ben Smyth. Verifiability of Helios Mixnet. Cryptology ePrint Archive, Report 2018/017, 2018.

[SP13] Ben Smyth and Alfredo Pironti. Truncating TLS Connections to Violate Beliefs in Web Applications. In *WOOT'13: 7th USENIX Workshop on Offensive Technologies*. USENIX Association, 2013. (First appeared at Black Hat USA 2013.).

[SP15] Ben Smyth and Alfredo Pironti. Truncating TLS Connections to Violate Beliefs in Web Applications. Technical Report hal-01102013, INRIA, 2015.

[SRKK10] Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjieh. Towards automatic analysis of election verifiability properties. In *ARSPA-WITS'10: Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, volume 6186 of *LNCS*, pages 165–182. Springer, 2010.

[Sta14] CACM Staff. ACM's 2014 General Election: Please Take This Opportunity to Vote. *Communications of the ACM*, 57(5):9–17, May 2014.

[TPLT13] Georgios Tsoukalas, Kostas Papadimitriou, Panos Louridas, and Panayiotis Tsanakas. From Helios to Zeus. *Journal of Election Technology and Systems*, 1(1), 2013.

[UK07] UK Electoral Commission. *Key issues and conclusions: May 2007 electoral pilot schemes*, May 2007.

[UM10] Dominique Unruh and Jörn Müller-Quade. Universally Composable Incoercibility. In *CRYPTO'10: 30th International Cryptology Conference*, volume 6223 of *LNCS*, pages 411–428. Springer, 2010.

[UN48] United Nations. *Universal Declaration of Human Rights*, 1948.

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

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

[ZCC+13] Filip Zagórski, Richard T. Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L. Vora. Remotegrity: Design and Use of an End-to-End Verifiable Remote Voting System. In *ACNS'13: 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *LNCS*, pages 441–457. Springer, 2013.