# Verifiability of Helios Mixnet

Ben Smyth

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

**Abstract.** We study game-based definitions of individual and universal verifiability by Smyth, Frink & Clarkson. We prove that building voting systems from El Gamal coupled with proofs of correct key generation suffices for individual verifiability. We also prove that it suffices for an aspect of universal verifiability. Thereby eliminating the expense of individual-verifiability proofs and simplifying universal-verifiability proofs for a class of encryption-based voting systems. We use the definitions of individual and universal verifiability to analyse the mixnet variant of Helios. Our analysis reveals that universal verifiability is not satisfied by implementations using the weak Fiat-Shamir transformation. Moreover, we prove that individual and universal verifiability are satisfied when statements are included in hashes (i.e., when using the Fiat-Shamir transformation, rather than the weak Fiat-Shamir transformation).

## 1 Introduction

An election is a decision-making procedure to choose representatives [3,15,25,31]. Choices should be made by voters with equal influence, and this must be ensured by voting systems [27, 28, 38]. Many electronic voting systems build upon creativity and skill, rather than scientific foundations, and are routinely broken in ways that permit adversaries to unduly influence the selection of representatives, e.g., [5, 7, 16, 20, 37]. Breaks can be avoided by carefully formulating rigorous and precise security definitions that capture notions of voters voting with equal influence, and proving that systems satisfy these definitions. We consider game-based security definitions in the computational model of cryptography [18], whereby a benign challenger, a malicious adversary and a voting system engage in a series of interactions which task the adversary to complete a challenge. Successful completion of the task corresponds to an execution of the voting system in which security is broken. Thus, the task captures what the adversary should not be able to achieve.

Universal verifiability formalises a notion of checking whether voters voted with equal influence.

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

Smyth, Frink & Clarkson formalise a game-based definition of universal verifiability [35]. That game tasks the adversary to compute inputs to the tallying

procedure, including an election outcome and some ballots, that cause checks to succeed when the outcome does not correspond to the votes expressed by those ballots, or that cause checks to fail when the outcome does correspond to the votes expressed.

Merely casting a ballot is insufficient to ensure it is recorded, because an adversary may discard ballots. Individual verifiability formalises the notion of voters convincing themselves that their ballot is amongst those recorded.

– Individual verifiability. A voter can check whether their ballot is recorded.

Smyth, Frink & Clarkson formalise individual verifiability as a game that tasks the adversary to cause a collision between ballots [35]. That game proceeds as follows: First, the adversary provides any inputs necessary to construct a ballot, including a vote $v_0$. Secondly, the challenger constructs a ballot using those inputs. Finally, the adversary and the challenger repeat the process to construct a second ballot for vote $v_1$. The adversary wins if the two independently constructed ballots are equal. Hence, winning signifies the existence of a scenario in which two voters cannot uniquely identify their ballot, thus a voter cannot be convinced that their ballot is recorded.

Equipped with definitions of individual and universal verifiability, we can analyse existing voting systems to determine whether they are secure. As is exemplified by the following two voting systems. The first (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 number of votes for each candidate. The second (Enc2Vote$^*$) extends the former to include proofs of correct computation, in particular, the tallier computes proofs of correct key generation and decryption. The former system achieves neither individual nor universal verifiability. Indeed, a public key can be maliciously constructed such that ciphertexts collide and spurious outcomes need not correspond to the encrypted votes. By comparison, the latter system achieves both individual and universal verifiability, because well-formed ciphertexts are unique (individual verifiability) and anyone can check proofs to determine whether the election outcome corresponds to votes expressed in recorded ballots (universal verifiability). Voting systems Enc2Vote and Enc2Vote$^*$ leak the ballot-vote relation during tallying; more advanced voting systems, such as Helios, do not.

Helios is intended to satisfy verifiability and ballot secrecy. For ballot secrecy, each voter encrypts their vote using a homomorphic encryption scheme. Those encrypted votes are homomorphically combined and the homomorphic combination is decrypted to reveal the outcome [2]. Alternatively, a mixnet is applied to the encrypted votes and the mixed encrypted votes are decrypted to reveal the outcome [1,6]. We refer to the former voting system as *Helios* and the latter as *Helios Mixnet*. For universal verifiability, the encryption and decryption steps are accompanied by non-interactive proofs demonstrating correct computation.

*Contribution and structure.* Section 3 proves that individual verifiability and an aspect of universal verifiability are satisfied by voting systems built from

El Gamal coupled with proofs of correct key generation, thereby eliminating the expense of individual-verifiability proofs and simplifying universal-verifiability proofs for a class of encryption-based voting systems. Section 4 presents an analysis of Helios Mixnet that uncovers a vulnerability in implementations, discusses a fix, and proves that individual and universal verifiability are satisfied when the fix is applied. The remaining sections present syntax and definitions of individual and universal verifiability (§2) and a brief conclusion (§5). Appendix A defines cryptographic primitives and relevant security definitions, and Appendix B contains proofs.

## 2 Election scheme syntax and verifiability definitions

We recall syntax for *election schemes* (Definition 1) from Smyth, Frink & Clarkson [35]. Election schemes capture a class of 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 recorded on a bulletin board. Thirdly, the tallier tallies the recorded ballots and announces an outcome, i.e., a distribution of votes. Finally, voters and other interested parties check that the outcome corresponds to votes expressed in recorded ballots.[1]

**Definition 1 (Election scheme [35]).** An *election scheme* is a tuple of probabilistic polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{TallyVerify})$ such that:

$\mathsf{Setup}$, denoted $(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa)$,[2] is run by the tallier.[3] It takes a security parameter $\kappa$ as input and outputs a key pair $pk, sk$, a maximum number of ballots $mb$, and a maximum number of candidates $mc$.[4]

$\mathsf{Vote}$, denoted $b \leftarrow \mathsf{Vote}(pk, v, nc, \kappa)$, is run by voters. It takes as input a public key $pk$, a voter's vote $v$, some number of candidates $nc$, and a security parameter $\kappa$. The vote should be selected from a sequence $1, \ldots, nc$ of candidates.[5] The algorithm outputs a ballot $b$ or error symbol $\perp$.

[1] Smyth, Frink & Clarkson use the syntax to model first-past-the-post voting systems and Smyth shows the syntax is sufficiently versatile to capture ranked-choice voting systems [33]. Moreover, Smyth, Frink & Clarkson extend the syntax to voting systems with eligibility verifiability, which enables anyone to check whether counted votes were cast by voters. Eligibility verifiability seems to require expensive infrastructures for voter credentials and some systems – including Helios and Helios Mixnet – forgo eligibility verifiability in favour of cheaper, non-verifiable ballot authentication mechanisms. Hence, we do not pursue eligibility verifiability further.

[2] 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 $r$. Let $A(x_1, \ldots, x_n)$ denote $A(x_1, \ldots, x_n; r)$, where $r$ is chosen uniformly at random. And let $\leftarrow$ denote assignment.

[3] Some election schemes (e.g., Helios) permit the tallier's role to be distributed amongst several talliers. In this manuscript, we consider a single tallier for simplicity.

[4] Some voting systems bound the number of ballots $mb$, respectively candidates $mc$, to ensure correctness. For example, Helios requires $mb$ and $mc$ to be less than or equal to the size of the underlying encryption scheme's message space.

[5] Candidates are (abstractly) represented as integers, rather than alphanumeric strings, for brevity.

Tally, denoted $(\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$, is run by the tallier. It takes as input a private key $sk$, a bulletin board $\mathfrak{bb}$, some number of candidates $nc$, and a security parameter $\kappa$, where $\mathfrak{bb}$ is a set. And outputs an election outcome $\mathfrak{v}$ and a non-interactive tallying proof $pf$ demonstrating that the outcome corresponds to votes expressed in ballots on the bulletin board. The election outcome $\mathfrak{v}$ should be a vector of length $nc$ such that $\mathfrak{v}[v]$ indicates the number of votes for candidate $v$.[6]

Verify, denoted $s \leftarrow \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa)$, is run to audit an election. It 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 $\kappa$. And outputs a bit $s$, where 1 signifies success and 0 signifies failure.

Election schemes must satisfy *correctness*: there exists a negligible function $\mathsf{negl}$, such that for all security parameters $\kappa$, 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:

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

For individual verifiability, voters must be able to check whether their ballot is recorded. Smyth, Frink & Clarkson capture this notion using a game that challenges the adversary to provide inputs to algorithm Vote that cause ballots to collide.

**Definition 2 (Individual verifiability [35]).** An election scheme (Setup, Vote, Tally, Verify) satisfies *individual verifiability*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}$, such that for all security parameters $\kappa$, we have $\Pr[(pk, nc, v, v') \leftarrow \mathcal{A}(\kappa); b \leftarrow \mathsf{Vote}(pk, nc, v, \kappa); b' \leftarrow \mathsf{Vote}(pk, nc, v', \kappa) : b = b' \wedge b \neq \perp \wedge b' \neq \perp] \leq \mathsf{negl}(\kappa)$.

For universal verifiability, anyone must be able to check whether the election outcome represents the votes used to construct ballots on the bulletin board. The formal definition of universal verifiability by Smyth, Frink & Clarkson requires algorithm Verify to accept if and only if the election outcome is correct.[7] The *if* requirement is captured by completeness (Definition 3), which stipulates that election outcomes produced by algorithm Tally will actually be accepted by algorithm Verify. And the *only if* requirement is captured by soundness (Definition 5), which challenges an adversary to concoct a scenario in which algorithm Verify accepts, but the election outcome is not correct.

[6] Let $\mathfrak{v}[v]$ denote component $v$ of vector $\mathfrak{v}$.

[7] Quaglia & Smyth [30] provide a tutorial-style introduction to the individual and universal verifiability definitions by Smyth, Frink & Clarkson.

**Definition 3 (Completeness [35]).** An election scheme ($\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}$) satisfies *completeness*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}$, such that for all security parameters $\kappa$, we have $\Pr[(pk, sk, mb, mc) \leftarrow \mathsf{Setup}(\kappa); (\mathfrak{bb}, nc) \leftarrow \mathcal{A}(pk, \kappa); (\mathfrak{v}, pf) \leftarrow \mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa) : |\mathfrak{bb}| \leq mb \wedge nc \leq mc \Rightarrow \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1] > 1 - \mathsf{negl}(\kappa)$.

Smyth, Frink & Clarkson capture correct election outcomes using function *correct-outcome*, which is defined such that $\textit{correct-outcome}(pk, nc, \mathfrak{bb}, \kappa)[v] = \ell$ iff $\exists^{=\ell} b \in \mathfrak{bb} \setminus \{\bot\} : \exists r : b = \mathsf{Vote}(pk, v, nc, \kappa; r)$,[8] where $\textit{correct-outcome}(pk, nc, \mathfrak{bb}, \kappa)$ is a vector of length $nc$ and $1 \leq v \leq nc$. Hence, component $v$ of vector $\textit{correct-outcome}(pk, nc, \mathfrak{bb}, \kappa)$ equals $\ell$ iff there exist $\ell$ ballots on the bulletin board that are votes for candidate $v$. The function requires ballots to be interpreted for only one candidate, which can be ensured by injectivity.[9]

**Definition 4 (Injectivity).** An election scheme ($\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}$) satisfies *injectivity*,[10] if for all probabilistic polynomial-time adversaries $\mathcal{A}$, security parameters $\kappa$ 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 $b \neq b'$.

**Definition 5 (Soundness [35]).** An election scheme $\Gamma = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ satisfies *soundness*, if $\Gamma$ satisfies injectivity and for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}$, such that for all security parameters $\kappa$, we have $\Pr[(pk, nc, \mathfrak{bb}, \mathfrak{v}, pf) \leftarrow \mathcal{A}(\kappa) : \mathfrak{v} \neq \textit{correct-outcome}(pk, nc, \mathfrak{bb}, \kappa) \wedge \mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1] \leq \mathsf{negl}(\kappa)$.

**Definition 6 (Universal verifiability).** An election scheme $\Gamma$ satisfies *universal verifiability*, if completeness, injectivity and soundness are satisfied.

Other definitions of verifiability exist. In particular, definitions have been proposed by Juels, Catalano & Jakobsson [17], Cortier *et al.* [9] and Kiayias, Zacharias & Zhang [19]. Smyth, Frink & Clarkson [35, §7] show that definitions by Juels, Catalano & Jakobsson and Cortier *et al.* do not detect attacks 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 attacks. By comparison, Definition 6 detects these attacks, thereby motivating its adoption.

Küsters *et al.* [21–23] propose an alternative, holistic notion of verifiability called *global verifiability*, which must be instantiated with a goal. Smyth, Frink & Clarkson [35, §8] show that goals proposed by Küsters *et al.* [24, §5.2] and by

[8] Function *correct-outcome* uses a *counting quantifier* [32] denoted $\exists^{=}$. Predicate $(\exists^{=\ell} x : P(x))$ holds exactly when there are $\ell$ distinct values for $x$ such that $P(x)$ is satisfied. Variable $x$ is bound by the quantifier, whereas $\ell$ is free.

[9] Injectivity resembles individual verifiability, except individual verifiability allows votes to be equal.

[10] Smyth, Frink & Clarkson [35] consider a definition of injectivity which quantifies over all public keys, rather than public keys constructed by an adversary. That definition is stronger than necessary.

Cortier et al. [10, §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 proved that their notion of verifiability is strictly stronger than global verifiability with that goal (the "gap" is due to an uninteresting technical detail), which further motivates the adoption of Definition 6.[11]

## 3 Encryption ensures individual verifiability & injectivity

Definitions of individual verifiability and injectivity only focus on properties of algorithm $\mathsf{Vote}$, hence, to prove these properties for election scheme $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$, it suffices to prove the existence of an election scheme $(\mathsf{Setup}', \mathsf{Vote}, \mathsf{Tally}', \mathsf{Verify}')$ satisfying both properties. We demonstrate the existence of such schemes by coupling election scheme $\mathsf{Enc2Vote}$ with proofs of correct key generation.[12,13]

**Definition 7 ($\mathsf{Enc2Vote}^+$).** Suppose $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is an asymmetric encryption scheme, $\Sigma$ is a sigma protocol that proves correct key generation, and $\mathcal{H}$ is a hash function. Let $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$.[14] We define $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ such that:

- $\mathsf{Setup}(\kappa)$ selects coins $s$ uniformly at random, computes $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa; s); \rho \leftarrow \mathsf{ProveKey}((\kappa, pk, \mathfrak{m}), (sk, s), \kappa); pk' \leftarrow (pk, \mathfrak{m}, \rho); sk' \leftarrow (pk, sk)$, derives $mc$ as the largest integer such that $\{0, \ldots, mc\} \subseteq \{0\} \cup \mathfrak{m}$, and outputs $(pk', sk', p(\kappa), mc)$, where $p$ is a polynomial function.
- $\mathsf{Vote}(pk', v, nc, \kappa)$ parses $pk'$ as vector $(pk, \mathfrak{m}, \rho)$, outputting $\perp$ if parsing fails or $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \neq 1 \vee v \notin \{1, \ldots, nc\} \vee \{1, \ldots, nc\} \not\subseteq \mathfrak{m}$, computes $b \leftarrow \mathsf{Enc}(pk, v)$, and outputs $b$.
- $\mathsf{Tally}(sk', \mathfrak{bb}, nc, \kappa)$ parses $sk'$ as vector $(pk, sk)$, outputting $\perp$ if parsing fails, initialises vector $\mathfrak{v}$ of length $nc$, computes **for** $b \in \mathfrak{bb}$ **do** $v \leftarrow \mathsf{Dec}(sk, b)$; **if** $1 \leq v \leq nc$ **then** $\mathbf{v}[v] \leftarrow \mathbf{v}[v] + 1$, and outputs $(\mathfrak{v}, \epsilon)$, where $\epsilon$ is a constant symbol.

[11] Cortier *et al.* [10, §8.5 & §10.1] claim that definitions by Smyth, Frink & Clarkson are flawed. Those claims were discussed with Cortier *et al.* (email communication, April'16) and are believed to be false [35, §9]. Moreover, Smyth, Frink & Clarkson prove that any flaw in their definitions implies flaws in the context of global verifiability, which should increase confidence in their definitions.

[12] Election scheme $\mathsf{Enc2Vote}^*$ (§1) couples $\mathsf{Enc2Vote}$ with proofs of correct key generation *and* proofs of correct decryption, hence, it is distinguished from schemes produced by $\mathsf{Enc2Vote}^+$. This distinction enables $\mathsf{Enc2Vote}^*$ to satisfy individual and universal verifiability, whereas $\mathsf{Enc2Vote}^+$ cannot produce schemes satisfying universal verifiability.

[13] Election scheme $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H})$ adopts the setup algorithm formalised by Smyth, Frink & Clarkson for Helios [35, Appendix C].

[14] Let $\mathsf{FS}(\Sigma, \mathcal{H})$ denote the non-interactive proof system derived by application of the Fiat-Shamir transformation to sigma protocol $\Sigma$ and hash function $\mathcal{H}$.

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

**Lemma 1.** *Given an asymmetric encryption scheme $\Pi$, a sigma protocol $\Sigma$ that proves correct key generation, and a hash function $\mathcal{H}$, we have* $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H})$ *is an election scheme.*

A proof of Lemma 1 and all further proofs appear in Appendix B.

Secrecy properties of asymmetric encryption schemes ensure ciphertexts do not collide. Indeed, IND-CPA demands that no adversary can construct a ciphertext that collides with the challenge ciphertext.[15] But, such secrecy properties assume public keys are generated (by key generation algorithms) using coins chosen uniformly at random. By comparison, individual verifiability and injectivity assume public keys are constructed by the adversary. Thus, secrecy properties are insufficient to ensure election scheme $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H})$ satisfies individual verifiability and injectivity. Nonetheless, given that the scheme's $\mathsf{Vote}$ algorithm checks correct key generation, it suffices that ciphertexts do not collide for correctly generated keys.

**Proposition 2.** *Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an asymmetric encryption scheme, $\Sigma$ be a sigma protocol that proves correct key generated, and $\mathcal{H}$ be a hash function. Election scheme* $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H})$ *satisfies individual verifiability if for all probabilistic polynomial-time adversaries $\mathcal{A}$ and security parameters $\kappa$ we have*

$$\begin{aligned}\Pr[(pk, \mathfrak{m}, \rho, m, m') \leftarrow \mathcal{A}(\kappa); c \leftarrow \mathsf{Enc}(pk, m); c' \leftarrow \mathsf{Enc}(pk, m') \\ : \mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1 \wedge m, m' \in \mathfrak{m} \Rightarrow c \neq c'] > 1 - \mathsf{negl}(\kappa),\end{aligned}$$

*where $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$. Moreover, the election scheme satisfies injectivity if the probability is 1 when plaintexts $m$ and $m'$ are distinct.*

Our proof of Proposition 2 (Appendix B.2) follows immediately from our preconditions. It is nevertheless useful, because the preconditions are defined over encryption scheme $\Pi$ and proof system $\mathsf{FS}(\Sigma, \mathcal{H})$, rather than election scheme $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H})$, which makes the preconditions easier to reason with. For El Gamal [12], the preconditions are ensured if the proof system checks parameters:

**Definition 8.** Let $(\mathsf{ProveKey}, \mathsf{VerKey})$ be a non-interactive proof system that proves correct key generation. The proof system *checks El Gamal parameters*, if for all security parameters $\kappa$, public keys $pk$, messages spaces $\mathfrak{m}$, and proofs $\rho$, we have $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1$ implies $pk$ is a vector $(p, q, g, h)$ such that $p = 2 \cdot q + 1$, $|q| = \kappa$, $g$ is a generator of $\mathbb{Z}_p^*$ of order $q$, $h \in \mathbb{Z}_p^*$, and $\mathfrak{m} = \{1, \ldots, p-1\}$.

**Theorem 3.** *Let $\Pi$ be El Gamal, $\Sigma$ be a sigma protocol that proves correct key generation, and $\mathcal{H}$ be a hash function. Suppose proof system $\mathsf{FS}(\Sigma, \mathcal{H})$ checks El Gamal parameters. We have* $\mathsf{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H})$ *satisfies individual verifiability and injectivity.*

[15] Correctness of asymmetric encryption schemes only ensures ciphertexts do not collide for distinct plaintexts.

We exploit Theorem 3 in the following section to derive a proof of individual verifiability for free and to simplify a proof of universal verifiability.

## 4 Case study: Helios Mixnet

Helios Mixnet can be informally modelled as an election scheme such that:

- Setup generates a key pair for an asymmetric homomorphic encryption scheme, proves correct key generation in zero-knowledge, and outputs the public key coupled with the 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 the ballots on the bulletin board for which proofs hold, mixes the ciphertexts in those ballots, decrypts the ciphertexts output by the mix to reveal the distribution of candidate preferences, and announces that distribution, along with zero-knowledge proofs demonstrating correct decryption and mixing.
- Verify checks the proofs and accepts the distribution if these checks succeed.

Neither Adida [1] nor Bulens, Giry & Pereira [6] have released an implementation of Helios Mixnet.[16] Tsoukalas *et al.* [36] released *Zeus* as a fork of Helios 3.1.4 spliced with mixnet code to derive an implementation,[17] and Yingtong Li released *helios-server-mixnet* as an extension of Zeus with threshold asymmetric encryption.[18] Both implementations use multiplicatively-homomorphic asymmetric encryption, rather than additively-homomorphic encryption.

Chang-Fong & Essex [8] show that Helios 2.0 does not satisfy completeness (because cryptographic parameters were not checked for suitability), 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 parameters).[19] Moreover, Bernhard, Pereira & Warinschi [4] show that Helios 3.1.4 does not satisfy soundness.[20,21] They also demonstrate a denial of

[16] The planned implementation of Helios Mixnet (`http://documentation.heliosvoting.org/verification-specs/mixnet-support`, published c. 2010, accessed 19 Dec 2017, and `https://web.archive.org/web/20110119223848/http://documentation.heliosvoting.org/verification-specs/helios-v3-1`, published Dec 2010, accessed 15 Sep 2017) has not been released.

[17] `https://github.com/grnet/zeus`, accessed 15 Sep 2017.

[18] `https://github.com/RunasSudo/helios-server-mixnet`, accessed 15 Sep 2017.

[19] `https://github.com/benadida/helios-server/pull/133`, published 31 May 2016, accessed 21 Sep 2017.

[20] Bernhard, Pereira & Warinschi show that a malicious tallier can add votes for their preferred candidate and remove votes for other candidates. Smyth, Frink & Clarkson formalise that attack and prove that soundness is not satisfied [35].

[21] A further soundness vulnerability is known [35], as are secrecy [11] and eligibility [26] vulnerabilities.

service attack. We exploit their denial of service attack to show that implementations of Helios Mixnet do not satisfy soundness: A malicious tallier can decrypt the ciphertexts output by the mix to reveal the distribution of candidate preferences, select the ciphertexts that decrypt to the tallier's preferred subdistribution, prove correct decryption of those ciphertexts, and exploit the attack by Bernhard, Pereira & Warinschi to falsify proofs that the remaining ciphertexts decrypt to arbitrary elements of the message space, thereby enabling the malicious tallier to exclude votes from the election outcome.

*Remark 4.* Zeus does not satisfy soundness.

Similarly, helios-server-mixnet does not satisfy soundness when a $(n, n)$-threshold is used. An informal proof of these claims follows from our discussion and a formal proof is omitted.

Helios 3.1.4 uses additively-homomorphic El Gamal, hence, an adversary can falsify that ciphertexts decrypt to arbitrary elements of the group, but cannot recover the corresponding messages, because solving discrete logarithms for arbitrary group elements is hard. Thus, the attack by Bernhard, Pereira & Warinschi leads to a denial of service attack against Helios 3.1.4, whereby the election outcome is not recovered, rather than an attack that violates soundness. By comparison, our attack against implementations of Helios Mixnet violates soundness, because a malicious tallier can exclude votes from the election outcome.

Bernhard, Pereira & Warinschi attribute vulnerabilities of Helios 3.1.4 to use of the Fiat-Shamir transformation without including statements in hashes (i.e., the weak Fiat-Shamir transformation), and recommend including statements in hashes (i.e., using the Fiat-Shamir transformation) as a defence. Implementations of Helios Mixnet can be extended to use the Fiat-Shamir transformation and we derive a formalisation of that extension from $\mathsf{Enc2Vote}^{+}(\Pi, \Sigma, \mathcal{H})$ by replacing its tallying and verification algorithms,[22] and by using a suitable asymmetric encryption algorithm. Using the Fiat-Shamir transformation (rather than the weak Fiat-Shamir transformation) ensures that proofs of correct decryption cannot be falsified, hence, the formalisation is not vulnerable to the aforementioned attack.

**Definition 9.** Suppose $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a homomorphic asymmetric encryption algorithm, $\Sigma_1$ is a sigma protocol that proves correct key construction, $\Sigma_2$ is a sigma protocol that proves plaintext knowledge, and $\mathcal{H}$ is a hash function. Let $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$ and $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$. Moreover, let $\pi(\Pi, \Sigma_2, \mathcal{H}) = (\mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}')$ be an asymmetric encryption scheme such that:

- $\mathsf{Enc}'(pk, v)$ selects coins $r$ uniformly at random, computes $c \leftarrow \mathsf{Enc}(pk, v; r)$; $\sigma \leftarrow \mathsf{ProveCiph}((pk, c), (v, r), \kappa)$, and outputs $(c, \sigma)$.

[22] The tallying and verification algorithms in Definition 9 adapt (unpublished) algorithms prepared by Quaglia & Smyth in the context of [29]. Quaglia & Smyth have since incorporated these adaptations into their work to take advantage of the results presented in this manuscript.

Suppose $\Sigma_3$ is a sigma protocol that proves correct decryption and $\Sigma_4$ is a sigma protocol that proves mixing. Let $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$ and $\mathsf{FS}(\Sigma_4, \mathcal{H}) = (\mathsf{ProveMix}, \mathsf{VerMix})$. We define $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$, where $\mathsf{Enc2Vote}^+(\pi(\Pi, \Sigma_2, \mathcal{H}), \Sigma_1, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}', \mathsf{Verify}')$ and algorithms $\mathsf{Tally}$ and $\mathsf{Verify}$ are defined below.

$\mathsf{Tally}(sk', nc, \mathfrak{bb}, \kappa)$ initialises $\mathfrak{v}$ as a zero-filled vector of length $nc$, parses $sk'$ as a vector $(pk, sk)$, outputting $(\mathfrak{v}, \perp)$ if parsing fails, and proceeds as follows:

1. *Remove invalid ballots.* Let $\{b_1, \ldots, b_\ell\}$ be the largest subset of $\mathfrak{bb}$ such that for all $1 \leq i \leq \ell$ we have $b_i$ is a pair and $\mathsf{VerCiph}((pk, b_i[1]), b_i[2], \kappa) = 1$. If $\{b_1, \ldots, b_\ell\} = \emptyset$, then output $(\mathfrak{v}, \perp)$.
2. *Mix.* Select a permutation $\chi$ on $\{1, \ldots, \ell\}$ uniformly at random, initialise $\mathbf{bb}$ and $\mathbf{r}$ as vectors of length $\ell$, fill $\mathbf{r}$ with coins chosen uniformly at random, and compute

   **for** $1 \leq i \leq \ell$ **do**
   $\quad \mathbf{bb}[i] \leftarrow b_{\chi(i)}[1] \otimes \mathsf{Enc}(pk, \mathfrak{e}; \mathbf{r}[i])$;
   $pf_1 \leftarrow \mathsf{ProveMix}((pk, (b_1[1], \ldots, b_\ell[1]), \mathbf{bb}), (\mathbf{r}, \chi), \kappa)$;

   where $\mathfrak{e}$ is an identity element of $\Pi$'s message space with respect to $\odot$.
3. *Decrypt.* Initialise $\mathbf{W}$ and $pf_2$ as vectors of length $\ell$ and compute:

   **for** $1 \leq i \leq \ell$ **do**
   $\quad \mathbf{W}[i] \leftarrow \mathsf{Dec}(sk, \mathbf{bb}[i])$;
   $\quad pf_2[i] \leftarrow \mathsf{ProveDec}((pk, \mathbf{bb}[i], \mathbf{W}[i]), sk, \kappa)$;
   $\quad$ **if** $1 \leq \mathbf{W}[i] \leq nc$ **then**
   $\quad\quad \mathfrak{v}[\mathbf{W}[i]] \leftarrow \mathfrak{v}[\mathbf{W}[i]] + 1$;

Output $(\mathfrak{v}, (\mathbf{bb}, pf_1, \mathbf{W}, pf_2))$.

$\mathsf{Verify}(pk', nc, \mathfrak{bb}, \mathfrak{v}, pf, \kappa)$ derives the largest integer $mc$ such that $\{0, \ldots, mc\} \subseteq \{0\} \cup \mathfrak{m}$; parses $pk'$ as a vector $(pk, \mathfrak{m}, \rho)$ and $\mathfrak{v}$ parses as a vector of length $nc$, outputting 0 if parsing fails, $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) \neq 1$, $|\mathfrak{bb}| \not\leq p(\kappa)$, or $nc \not\leq mc$, where $p$ is the polynomial function used by algorithm $\mathsf{Setup}$ to bound the maximum number of ballots; and proceeds as follows:

1. *Remove invalid ballots.* Compute $\{b_1, \ldots, b_\ell\}$ as per Step 1 of algorithm $\mathsf{Tally}$. If $\{b_1, \ldots, b_\ell\} = \emptyset$ and $\mathfrak{v}$ is a zero-filled vector, then output 1. Otherwise, perform the following checks.
2. *Check mixing.* Parse $pf$ as a vector $(\mathbf{bb}, pf_1, \mathbf{W}, pf_2)$, outputting 0 if parsing fails, and check $\mathsf{VerMix}((pk, (b_1[1], \ldots, b_\ell[1]), \mathbf{bb}), pf_1, \kappa) = 1$.
3. *Check decryption.* Check $\mathbf{W}$ and $pf_2$ are vectors of length $\ell$, $\bigwedge_{i=1}^{\ell} \mathsf{VerDec}((pk, \mathbf{bb}[i], \mathbf{W}[i]), pf_2[i], \kappa) = 1$, and $\bigwedge_{v=1}^{nc} \exists^{=\mathfrak{v}[v]} i \in \{1, \ldots, \ell\} : v = \mathbf{W}[i]$.

If the above checks hold, then output 1, otherwise, output 0.

**Lemma 5.** *Suppose $\Pi$, $\Sigma_1$, $\Sigma_2$, $\Sigma_3$, $\Sigma_4$ and $\mathcal{H}$ satisfy the preconditions of Definition 9. We have $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ is an election scheme.*

Our formalisation of Helios Mixnet is similar to the formalisation of Helios by Smyth, Frink & Clarkson [35]. The main distinctions are as follows: First, given a vote $v$ from a sequence of candidates $1, \ldots, nc$, a Helios-Mixnet ballot contains an encryption of $v$, whereas a Helios ballot contains ciphertexts $c_1, \ldots, c_{nc-1}$ such that if $v < nc$, then $c_v$ encrypts plaintext one and the remaining ciphertexts all encrypt zero, otherwise, all ciphertexts encrypt zero. (Both Helios Mixnet and Helios ballots prove correct ciphertext construction, only Helios ballots prove the vote is selected from the sequence of candidates.) Secondly, Helios Mixnet decrypts individual ciphertexts after mixing, whereas Helios homomorphically combines ciphertexts and decrypts the resulting homomorphic combination. (Both Helios Mixnet and Helios prove correct decryption. For Helios Mixnet, decryption of individual votes is proved correct. Whereas Helios proves correct decryption of the election outcome.) Finally, the aforementioned distinctions lead to slight differences in the verification algorithms.

Since election schemes $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ and $\mathsf{Enc2Vote}^{+}(\pi(\Pi, \Sigma_2, \mathcal{H}), \Sigma_1, \mathcal{H})$ share the same voting algorithm, both schemes satisfy individual verifiability and injectivity by Proposition 2, assuming the proposition's preconditions are satisfied. Moreover, since the preconditions hold for El Gamal when parameters are checked (Corollary 2), we have:

**Corollary 6.** *Suppose $\Pi$, $\Sigma_1$, $\Sigma_2$, $\Sigma_3$, $\Sigma_4$ and $\mathcal{H}$ satisfy the preconditions of Definition 9. Further suppose $\Pi$ is El Gamal and $\Sigma_1$ checks El Gamal parameters. Election scheme $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ satisfies individual verifiability and injectivity.*

To evaluate whether universal verifiability is satisfied, it remains to consider completeness and soundness.

**Theorem 7.** *Suppose $\Pi$, $\Sigma_1$, $\Sigma_2$, $\Sigma_3$, $\Sigma_4$ and $\mathcal{H}$ satisfy the preconditions of Definition 9. Moreover, suppose $\Sigma_2$ satisfies special soundness and special honest verifier zero-knowledge and $\mathcal{H}$ is a random oracle. Election scheme $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ satisfies completeness. Further suppose, $\Pi$ is perfectly correct and perfectly homomorphic, $\Sigma_1$, $\Sigma_3$ and $\Sigma_4$ satisfy special soundness and special honest verifier zero-knowledge, and $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$ satisfies injectivity. The election scheme satisfies soundness.*

Theorem 7 requires perfect correctness, rather than computational correctness, because soundness quantifies over public keys constructed by an adversary and such an adversary might not construct the public key using coins chosen uniformly at random. (We can, nonetheless, verify whether public keys are constructed using the correct algorithm.) Thus, perfect correctness is required, because it quantifies over all coins. Moreover, perfect homomorphisms are similarly required.

These findings were reported to the Zeus developers, who conducted an investigation of their code and confirmed Zeus does not satisfy soundness.[23] They

[23] Email communication, Oct & Dec 2017.

promptly adopted and deployed the proposed fix,[24] this was straightforward, because they had already written code for the Fiat-Shamir transformation. These findings were also reported to the developer of helios-server-mixnet,[25] who confirmed soundness is not satisfied with a $(n, n)$-threshold, but is for other thresholds because code from the PloneVote[26] cryptographic library is used rather than code from Helios,[27] and has since adopted and deployed the proposed fix.[28]

## 5 Conclusion

We have introduced a construction that serves as a foundation for verifiable voting systems, and we have shown that it produces systems satisfying individual verifiability and the injectivity aspect of universal verifiability, when instantiated with El Gamal. Moreover, we have analysed verifiability of two implementations of Helios Mixnet and shown that the soundness aspect of universal verifiability is not satisfied, due to vulnerabilities in Helios. Finally, we propose a fix and exploit our construction to prove that the fix suffices for individual and universal verifiability.

*Acknowledgements.* I am grateful to Steve Kremer and the anonymous reviewers for feedback that helped improve this paper. I am also grateful to Yingtong Li (developer of helios-server-mixnet) and to Georgios Tsoukalas & Panos Louridas (developers of Zeus) for discussions about their voting systems.

## A Cryptographic primitives

**Definition 10 (Asymmetric encryption scheme [18]).** An *asymmetric encryption scheme* is a tuple of probabilistic polynomial-time algorithms ($\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$), such that:[29]

- ***Gen***, denoted $(pk, sk, \mathtt{m}) \leftarrow \mathsf{Gen}(\kappa)$, inputs a security parameter $\kappa$ and outputs a key pair $(pk, sk)$ and message space $\mathtt{m}$.
- ***Enc***, denoted $c \leftarrow \mathsf{Enc}(pk, m)$, inputs a public key $pk$ and message $m \in \mathtt{m}$, 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 $\mathsf{Dec}$ is deterministic.

[24] See commitments `d2653d4` (9 Oct 2017), `4fddcd3` (11 Oct 2017), and `aab1b6f` (9 Oct 2017), accessed 20 Dec 2017.
[25] Email communication, 22 Dec 2017.
[26] `https://github.com/HRodriguez/svelib`, accessed 4 Jan 2017.
[27] Email communication, 25 Dec 2017.
[28] See commitment `9af7674` (25 Dec 2017).
[29] Our definition differs from Katz and Lindell's original definition [18, Definition 10.1] in that we formally state the plaintext space.

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

**Definition 11 (Homomorphic encryption [35]).** An asymmetric encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *homomorphic*, with respect to ternary operators $\odot$, $\oplus$, and $\otimes$,[30] if there exists a negligible function $\mathsf{negl}$, such that for all security parameters $\kappa$, we have the following.[31] First, for all messages $m_1$ and $m_2$ we have $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa); c_1 \leftarrow \mathsf{Enc}(pk, m_1); c_2 \leftarrow \mathsf{Enc}(pk, m_2) : m_1, m_2 \in \mathfrak{m} \Rightarrow \mathsf{Dec}(sk, c_1 \otimes_{pk} c_2) = \mathsf{Dec}(sk, c_1) \odot_{pk} \mathsf{Dec}(sk, c_2)] > 1 - \mathsf{negl}(\kappa)$. Secondly, for all messages $m_1$ and $m_2$, and all coins $r_1$ and $r_2$, we have $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(\kappa) : m_1, m_2 \in \mathfrak{m} \Rightarrow \mathsf{Enc}(pk, m_1; r_1) \otimes_{pk} \mathsf{Enc}(pk, m_2; r_2) = \mathsf{Enc}(pk, m_1 \odot_{pk} m_2; r_1 \oplus_{pk} r_2)] > 1 - \mathsf{negl}(\kappa)$. A scheme is *perfectly homomorphic* if the aforementioned probabilities are 1.

**Definition 12 (Non-interactive proof system [35]).** A *non-interactive proof system* for a relation $R$ is a tuple of algorithms $(\mathsf{Prove}, \mathsf{Verify})$, such that:

- ***Prove***, denoted $\sigma \leftarrow \mathsf{Prove}(s, w, \kappa)$, is executed by a prover to prove $(s, w) \in R$.
- ***Verify***, denoted $v \leftarrow \mathsf{Verify}(s, \sigma, \kappa)$, is executed by anyone to check the validity of a proof. We assume $\mathsf{Verify}$ is deterministic.

Moreover, the system must be *complete*: there exists a negligible function $\mathsf{negl}$, such that for all statement and witnesses $(s, w) \in R$ and security parameters $\kappa$, we have $\Pr[\sigma \leftarrow \mathsf{Prove}(s, w, \kappa) : \mathsf{Verify}(s, \sigma, \kappa) = 1] > 1 - \mathsf{negl}(\kappa)$.

**Definition 13 (from [35]).** Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a homomorphic asymmetric encryption scheme and $\Sigma$ be a sigma protocol for a binary relation $R$.[32]

- $\Sigma$ *proves correct key generation* if a $((\kappa, pk, \mathfrak{m}), (sk, s)) \in R \Leftrightarrow (pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; s)$.

Further, suppose that $(pk, sk, \mathfrak{m})$ is the output of $\mathsf{Gen}(\kappa; s)$, for some security parameter $\kappa$ and coins $s$.

- $\Sigma$ *proves plaintext knowledge* if $((pk, c), (m, r)) \in R \Leftrightarrow c = \mathsf{Enc}(pk, m; r) \wedge m \in \mathfrak{m}$.

[30] Henceforth, we implicitly bind ternary operators, i.e., we write *$\Pi$ is a homomorphic asymmetric encryption scheme* as opposed to the more verbose *$\Pi$ is a homomorphic asymmetric encryption scheme, with respect to ternary operators $\odot$, $\oplus$, and $\otimes$.*

[31] We write $X \circ_{pk} Y$ for the application of ternary operator $\circ$ to inputs $X$, $Y$, and $pk$. We occasionally abbreviate $X \circ_{pk} Y$ as $X \circ Y$, when $pk$ is clear from the context.

[32] Given a binary relation $R$, we write $((s_1, \ldots, s_l), (w_1, \ldots, w_k)) \in R \Leftrightarrow P(s_1, \ldots, s_l, w_1, \ldots, w_k)$ for $(s, w) \in R \Leftrightarrow P(s_1, \ldots, s_l, w_1, \ldots, w_k) \wedge s = (s_1, \ldots, s_l) \wedge w = (w_1, \ldots, w_k)$, hence, $R$ is only defined over pairs of vectors of lengths $l$ and $k$.

– $\Sigma$ *proves mixing* if $((pk, \mathbf{c}, \mathbf{c}'), (\mathbf{r}, \chi)) \in R \Leftrightarrow \bigwedge_{1 \le i \le |\mathbf{c}|} \mathbf{c}'[i] = \mathbf{c}[\chi(i)] \otimes \mathsf{Enc}(pk, \mathfrak{e}; \mathbf{r}[i]) \wedge |\mathbf{c}| = |\mathbf{c}'| = |\mathbf{r}|$, where $\mathbf{c}$ and $\mathbf{c}'$ are both vectors of ciphertexts encrypted under $pk$, $\mathbf{r}$ is a vector of coins, $\chi$ is a permutation on $\{1, \ldots, |\mathbf{c}|\}$, and $\mathfrak{e}$ is an identity element of the encryption scheme's message space with respect to $\odot$.
– $\Sigma$ *proves correct decryption* if $((pk, c, m), sk) \in R \Leftrightarrow m = \mathsf{Dec}(sk, c)$.

**Definition 14 (Fiat-Shamir transformation [13]).** Given a sigma protocol $\Sigma = (\mathsf{Comm}, \mathsf{Chal}, \mathsf{Resp}, \mathsf{Verify}_\Sigma)$ for relation $R$ and a hash function $\mathcal{H}$, the *Fiat-Shamir transformation*, denoted $\mathsf{FS}(\Sigma, \mathcal{H})$, is the non-interactive proof system $(\mathsf{Prove}, \mathsf{Verify})$, defined as follows:

$\mathsf{Prove}(s, w, \kappa) =$
  $(\mathsf{comm}, t) \leftarrow \mathsf{Comm}(s, w, \kappa)$;
  $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s)$;
  $\mathsf{resp} \leftarrow \mathsf{Resp}(\mathsf{chal}, t, \kappa)$;
  **return** $(\mathsf{comm}, \mathsf{resp})$;

$\mathsf{Verify}(s, (\mathsf{comm}, \mathsf{resp}), \kappa) =$
  $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s)$;
  **return** $\mathsf{Verify}_\Sigma(s, (\mathsf{comm}, \mathsf{chal}, \mathsf{resp}), \kappa)$;

A string $m$ can be included in the hashes computed by algorithms $\mathsf{Prove}$ and $\mathsf{Verify}$. That is, the hashes are computed in both algorithms as $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s, m)$. We write $\mathsf{Prove}(s, w, m, \kappa)$ and $\mathsf{Verify}(s, (\mathsf{comm}, \mathsf{resp}), m, k)$ for invocations of $\mathsf{Prove}$ and $\mathsf{Verify}$ which include string $m$.

**Definition 15 (Simulation sound extractability [4,14,35]).** Suppose $\Sigma$ is a sigma protocol for relation $R$, $\mathcal{H}$ is a random oracle, and $(\mathsf{Prove}, \mathsf{Verify})$ is a non-interactive proof system, such that $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}, \mathsf{Verify})$. Further suppose $\mathcal{S}$ is a simulator for $(\mathsf{Prove}, \mathsf{Verify})$ and $\mathcal{H}$ can be patched by $\mathcal{S}$. Proof system $(\mathsf{Prove}, \mathsf{Verify})$ satisfies *simulation sound extractability* if there exists a probabilistic polynomial-time algorithm $\mathcal{K}$, such that for all probabilistic polynomial-time adversaries $\mathcal{A}$ and coins $r$, there exists a negligible function $\mathsf{negl}$, such that for all security parameters $\kappa$, we have:[33]

$$\begin{aligned}\Pr[\mathbf{P} \leftarrow (); \mathbf{Q} \leftarrow \mathcal{A}^{\mathcal{H},\mathcal{P}}(—; r); \mathbf{W} \leftarrow \mathcal{K}^{\mathcal{A}'}(\mathbf{H}, \mathbf{P}, \mathbf{Q}) : \\ |\mathbf{Q}| \neq |\mathbf{W}| \vee \exists j \in \{1, \ldots, |\mathbf{Q}|\} \,.\, (\mathbf{Q}[j][1], \mathbf{W}[j]) \notin R \wedge \\ \forall (s, \sigma) \in \mathbf{Q}, (t, \tau) \in \mathbf{P} \,.\, \mathsf{Verify}(s, \sigma, \kappa) = 1 \wedge \sigma \neq \tau] \le \mathsf{negl}(\kappa)\end{aligned}$$

where $\mathcal{A}(—; r)$ denotes running adversary $\mathcal{A}$ with an empty input and coins $r$, where $\mathbf{H}$ is a transcript of the random oracle's input and output, and where oracles $\mathcal{A}'$ and $\mathcal{P}$ are defined below:

– $\mathcal{A}'()$. Computes $\mathbf{Q}' \leftarrow \mathcal{A}(—; r)$, forwarding any of $\mathcal{A}$'s oracle queries to $\mathcal{K}$, and outputs $\mathbf{Q}'$. By running $\mathcal{A}(—; r)$, $\mathcal{K}$ is rewinding the adversary.

[33] We extend set membership notation to vectors: we write $x \in \mathbf{x}$ if $x$ is an element of the set $\{\mathbf{x}[i] : 1 \le i \le |\mathbf{x}|\}$.

– $\mathcal{P}(s)$. Computes $\sigma \leftarrow \mathcal{S}(s,\kappa); \mathbf{P} \leftarrow (\mathbf{P}[1],\ldots,\mathbf{P}[|\mathbf{P}|],(s,\sigma))$ and outputs $\sigma$.

Algorithm $\mathcal{K}$ is an *extractor* for $(\mathsf{Prove},\mathsf{Verify})$.

**Theorem 8 (from [4]).** *Let $\Sigma$ be a sigma protocol for relation $R$, and let $\mathcal{H}$ be a random oracle. Suppose $\Sigma$ satisfies special soundness and special honest verifier zero-knowledge. Non-interactive proof system $\mathsf{FS}(\Sigma,\mathcal{H})$ satisfies zero-knowledge and simulation sound extractability.*

# B Proofs

## B.1 Proof of Lemma 1

Let $\mathit{\Pi} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$, $\mathsf{FS}(\Sigma,\mathcal{H}) = (\mathsf{ProveKey},\mathsf{VerKey})$, and $\mathsf{Enc2Vote}^+(\mathit{\Pi}, \Sigma,\mathcal{H}) = (\mathsf{Setup},\mathsf{Vote},\mathsf{Tally},\mathsf{Verify})$. Suppose $\kappa$ is a security parameter, $nb$ and $nc$ are integers, and $v_1,\ldots,v_{nb} \in \{1,\ldots,nc\}$ are votes. Let $\mathfrak{v}$ be the election outcome that corresponds to those votes. Suppose $(pk', sk', mb, mc)$ is an output of $\mathsf{Setup}(\kappa)$ such that $nb \leq mb \wedge nc \leq mc$. By definition of algorithm $\mathsf{Setup}$, there exist coins $s$ such that $(pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; s)$, $\rho$ is an output of $\mathsf{ProveKey}((\kappa, pk, \mathfrak{m}), (sk, s), \kappa)$, $pk' = (pk, \mathfrak{m}, \rho)$, $sk' = (pk, sk)$, and $mc$ is the largest integer such that $\{0,\ldots,mc\} \subseteq \{0\} \cup \mathfrak{m}$. By completeness of $(\mathsf{ProveKey},\mathsf{VerKey})$, we have $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1$, with overwhelming probability. Suppose for each $i \in \{1,\ldots,nb\}$ that $b_i$ is an output of $\mathsf{Vote}(pk, v_i, nc, \kappa)$, hence, $b_i$ is an output of $\mathsf{Enc}(pk, v_i)$. Further suppose $(\mathfrak{v}', pf)$ is an output of $\mathsf{Tally}(sk, \{b_1,\ldots,b_{nb}\}, nc, \kappa)$. By definition of algorithm $\mathsf{Tally}$, we have $\bigwedge_{v=1}^{nc} \exists^{=\mathfrak{v}'[v]} i \in \{1,\ldots,nb\} : v = \mathsf{Dec}(sk, b_i)$. It follows by correctness of $\mathit{\Pi}$ that $\bigwedge_{v=1}^{nc} \exists^{=\mathfrak{v}'[v]} i \in \{1,\ldots,nb\} : v = v_i$, with overwhelming probability. Hence, $\mathfrak{v}'$ is the election outcome that corresponds to votes $v_1,\ldots,v_{nb}$, i.e., $\mathfrak{v} = \mathfrak{v}'$, concluding our proof. □

## B.2 Proof of Proposition 2

Let $\mathsf{Enc2Vote}^+(\mathit{\Pi}, \Sigma, \mathcal{H}) = (\mathsf{Setup},\mathsf{Vote},\mathsf{Tally},\mathsf{Verify})$. Suppose $\mathcal{A}$ is a probabilistic polynomial-time adversary and $\kappa$ is a security. Further suppose $(pk, nc, v, v')$ is an output of $\mathcal{A}(\kappa)$, $b$ is an output of $\mathsf{Vote}(pk', nc, v, \kappa)$, and $b'$ is an output of $\mathsf{Vote}(pk', nc, v', \kappa)$, such that $b \neq \perp$ and $b' \neq \perp$. By definition of algorithm $\mathsf{Vote}$, public key $pk'$ is a vector $(pk, \mathfrak{m}, \rho)$ such that $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1$ and $v, v' \in \{1,\ldots,nc\} \subseteq \mathfrak{m}$. Moreover, $b$ is an output of $\mathsf{Enc}(pk, v)$ and $b'$ is an output of $\mathsf{Enc}(pk, v')$. Thus, $b \neq b'$ by our precondition, with overwhelming probability, therefore, individual verifiability is satisfied. For injectivity, we further suppose $v \neq v'$, hence, $b \neq b'$ by our precondition, which concludes our proof. □

## B.3 Proof of Theorem 3

Suppose $\mathcal{A}$ is a probabilistic polynomial-time adversary and $\kappa$ is a security parameter. Further suppose $(pk, \mathfrak{m}, \rho, m, m')$ is an output of $\mathcal{A}(\kappa)$, $c$ is an output of $\mathsf{Enc}(pk, m)$, and $c'$ is an output of $\mathsf{Enc}(pk, m')$, such that $\mathsf{VerKey}((\kappa, pk,$

$\mathfrak{m}), \rho, \kappa) = 1$ and $m, m' \in \mathfrak{m}$. Since $\mathsf{FS}(\Sigma, \mathcal{H})$ checks El Gamal parameters, public key $pk$ is a vector $(p, q, g, h)$ such that $p = 2 \cdot q + 1$, $|q| = \kappa$, $g$ is a generator of $\mathbb{Z}_p^*$ of order $q$, $h \in \mathbb{Z}_p^*$, and $\mathfrak{m} = \{1, \ldots, p-1\}$. By definition of El Gamal, we have $c[1] = g^r \pmod p$ and $c'[1] = g^{r'} \pmod p$ for some coins $r$ and $r'$ chosen uniformly at random from $\mathbb{Z}_q^*$. It follows that coins $r$ and $r'$ are distinct, with overwhelming probability. Hence, $c[1] \neq c'[1]$, therefore, $c \neq c'$, with overwhelming probability. Thus, individual verifiability is satisfied. For injectivity, if $r \neq r'$, then $c[1] \neq c'[1]$, therefore, $c \neq c'$, otherwise ($r = r'$), $c[2] = h^r \cdot m \pmod p$ and $c'[2] = h^r \cdot m' \pmod p$ by definition of El Gamal, hence, $c \neq c'$ when $m$ and $m'$ are distinct. Thus, injectivity is satisfied too, which concludes our proof. □

### B.4 Proof of Lemma 5

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, $\pi(\Pi, \Sigma_1, \mathcal{H}) = (\mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}')$, $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$, and $\mathcal{H}$ is a hash function. Suppose $\kappa$ is a security parameter, $nb$ and $nc$ are integers, and $v_1, \ldots, v_{nb} \in \{1, \ldots, nc\}$ are votes. Let $\mathfrak{v}$ be the election outcome that corresponds to those votes. Suppose $(pk', sk', mb, mc)$ is an output of $\mathsf{Setup}(\kappa)$ such that $nb \leq mb \wedge nc \leq mc$. Further suppose for each $i \in \{1, \ldots, nb\}$ that $b_i$ is an output of $\mathsf{Vote}(pk, v_i, nc, \kappa)$. Hence, by reasoning similar to that given in the proof of Lemma 1, each ballot is a ciphertext produced by $\mathsf{Enc}'$. By definition of algorithm $\mathsf{Enc}'$, we have for each $i \in \{1, \ldots, nb\}$ there exist coins $r_i$ such that $b_i[1] = \mathsf{Enc}(pk, v_i; r_i)$ and $b_i[2]$ is an output of $\mathsf{ProveCiph}((pk, c), (v, r), \kappa)$. It follows by completeness of $(\mathsf{ProveCiph}, \mathsf{VerCiph})$ that $\bigwedge_{i=1}^{nb} \mathsf{VerCiph}((pk, b_i[1]), b_i[2], \kappa) = 1$.

Suppose $(\mathfrak{v}', pf)$ is an output of $\mathsf{Tally}(sk, \{b_1, \ldots, b_{nb}\}, nc, \kappa)$. By definition of algorithm $\mathsf{Tally}$, we have $pf$ is a vector $(\mathbf{bb}, pf_1, \mathbf{W}, pf_2)$ such that for all $1 \leq i \leq \ell$ we have $\mathbf{bb}[i] = b_{\chi(i)}[1] \otimes \mathsf{Enc}(pk, \mathfrak{e}; \mathbf{r}[i])$, where $\chi$ is a permutation on $\{1, \ldots, \ell\}$, $\mathbf{r}$ is a vector of coins, and $\mathfrak{e}$ is an identity element. Moreover, we have for all $1 \leq i \leq \ell$ that $\mathbf{W}[i] = \mathsf{Dec}(sk, \mathbf{bb}[i])$. Since $\mathfrak{v}'$ is derived by initialising $\mathbf{W}$ as a vector of length $\ell$ and computing **for** $1 \leq i \leq \ell$ **do if** $1 \leq \mathbf{W}[i] \leq nc$ **then** $\mathfrak{v}'[\mathbf{W}[i]] \leftarrow \mathfrak{v}'[\mathbf{W}[i]] + 1$, we have $\bigwedge_{v=1}^{nc} \exists^{=\mathfrak{v}'[v]} i \in \{1, \ldots, \ell\} : v = \mathsf{Dec}(sk, \mathbf{bb}[i])$. Moreover, since $\Pi$ is homomorphic, we have for all $1 \leq i \leq \ell$ that $\mathbf{bb}[i] = \mathsf{Enc}(pk, v_{\chi(i)}; r_{\chi(i)} \oplus \mathbf{r}[i])$, with overwhelming probability. It follows by correctness of $\Pi$ that $\bigwedge_{v=1}^{nc} \exists^{=\mathfrak{v}'[v]} i \in \{1, \ldots, \ell\} : v = v_{\chi(i)}$, with overwhelming probability. Since $\chi$ is a permutation on $\{1, \ldots, nc\}$, election outcome $\mathfrak{v}'$ corresponds to votes $v_1, \ldots, v_{nb}$, i.e., $\mathfrak{v} = \mathfrak{v}'$, with overwhelming probability, thereby concluding our proof. □

### B.5 Proof of Theorem 7

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$, $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$, $\mathsf{FS}(\Sigma_4, \mathcal{H}) = (\mathsf{ProveMix}, \mathsf{VerMix})$, and $\mathsf{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$. We prove completeness and soundness below. The proofs are similar to proofs of completeness and soundness for Helios [35, Appendix F], due to the aforementioned similarities between Helios Mixnet and Helios (§4).

*Completeness.* Suppose $\mathcal{A}$ is an adversary and $\kappa$ is a security parameter. Further suppose $(pk', sk', mb, mc)$ is an output of $\mathsf{Setup}(\kappa)$, $(\mathfrak{bb}, nc)$ is an output of $\mathcal{A}(pk', \kappa)$ such that $|\mathfrak{bb}| \leq mb$ and $nc \leq mc$, and $(\mathfrak{v}, pf)$ is an output of $\mathsf{Tally}(sk, \mathfrak{bb}, nc, \kappa)$. By definition of algorithm $\mathsf{Setup}$, there exist coins $s$ such that $(pk, sk, \mathfrak{m}) = \mathsf{Gen}(\kappa; s)$, $\rho$ is an output of $\mathsf{ProveKey}((\kappa, pk, \mathfrak{m}), (sk, s), \kappa)$, $pk'$ is the vector $(pk, \mathfrak{m}, \rho)$, $sk'$ is the vector $(pk, sk)$, $mc$ is the largest integer such that $\{0, \ldots, mc\} \subseteq \{0\} \cup \mathfrak{m}$, and $mb = p(\kappa)$, where $p$ is a polynomial function. By completeness of $(\mathsf{ProveKey}, \mathsf{VerKey})$, we have $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1$, with overwhelming probability. Suppose $\{b_1, \ldots, b_\ell\}$ is computed as per Step 1 of algorithm $\mathsf{Tally}$. If $\{b_1, \ldots, b_\ell\} = \emptyset$, then we have $\mathfrak{v}$ as a zero-filled vector of length $nc$ and $\mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1$, hence, completeness is satisfied. Otherwise, we proceed as follows.

Since $\Sigma_2$ satisfies special soundness and special honest verifier zero-knowledge, proof system $\mathsf{FS}(\Sigma_2, \mathcal{H})$ satisfies simulation should extractability (Theorem 8), hence, we have for each $i \in \{1, \ldots, \ell\}$ that $b_i$ is a ciphertext. By definition of algorithm $\mathsf{Tally}$, we have $pf$ is a vector $(\mathbf{bb}, pf_1, \mathbf{W}, pf_2)$ such that for all $1 \leq i \leq \ell$ we have $\mathbf{bb}[i] = b_{\chi(i)}[1] \otimes \mathsf{Enc}(pk, \mathfrak{e}; \mathbf{r}[i])$, where $\chi$ is a permutation on $\{1, \ldots, \ell\}$, $\mathfrak{e}$ is an identity element of $\Pi$'s message space with respect to $\odot$, and $\mathbf{r}$ is a vector of coins. Moreover, $pf_1$ is an output of $\mathsf{ProveMix}((pk, (b_1[1], \ldots, b_\ell[1]), \mathbf{bb}), (\mathbf{r}, \chi), \kappa)$. By completeness of $(\mathsf{ProveMix}, \mathsf{VerMix})$, we have $\mathsf{VerMix}((pk, (b_1[1], \ldots, b_\ell[1]), \mathbf{bb}), pf_1, \kappa) = 1$, with overwhelming probability. It follows that checks hold in Step 2 of algorithm $\mathsf{Verify}$.

By Step 3 of algorithm $\mathsf{Tally}$, we have for all $1 \leq i \leq \ell$ that $\mathbf{W}[i]$ is an output of $\mathsf{Dec}(sk, \mathbf{bb}[i])$ and $pf_2[i]$ is an output of $\mathsf{ProveDec}((pk, \mathbf{bb}[i], \mathbf{W}[i]), sk, \kappa)$. By completeness of $(\mathsf{ProveDec}, \mathsf{VerDec})$, we have $\bigwedge_{i=1}^{\ell} \mathsf{VerDec}((pk, \mathbf{bb}[i], \mathbf{W}[i]), pf_2[i], \kappa) = 1$, with overwhelming probability. Moreover, since $\mathfrak{v}$ is derived by initialising $\mathbf{W}$ as a vector of length $\ell$ and computing **for** $1 \leq i \leq \ell$ **do if** $1 \leq \mathbf{W}[i] \leq nc$ **then** $\mathfrak{v}[\mathbf{W}[i]] \leftarrow \mathfrak{v}[\mathbf{W}[i]] + 1$, we have $\bigwedge_{v=1}^{nc} \exists^{=\mathfrak{v}[v]} i \in \{1, \ldots, \ell\} : v = \mathbf{W}[i]$. It follows that checks hold in Step 3 of algorithm $\mathsf{Verify}$.

Since the above checks succeed, algorithm $\mathsf{Verify}$ outputs 1, with overwhelming probability, hence, completeness is satisfied.

*Soundness.* We suppose each of the proof systems satisfies simulation sound extractability by Theorem 8. Moreover, since $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ shares algorithm $\mathsf{Vote}$ with $\mathsf{Enc2Vote}^+(\pi(\Pi, \Sigma_2, \mathcal{H}), \Sigma_1, \mathcal{H})$, it follows from Proposition 2 that both schemes satisfy injectivity, hence, a prerequisite of soundness is satisfied.

Suppose $\mathcal{A}$ is a probabilistic polynomial-time adversary, $\kappa$ is a security parameter, and $(pk', nc, \mathfrak{bb}, \mathfrak{v}, pf)$ is an output of $\mathcal{A}(\kappa)$ such that $\mathsf{Verify}(pk, \mathfrak{bb}, nc, \mathfrak{v}, pf, \kappa) = 1$. By definition of algorithm $\mathsf{Verify}$, public key $pk'$ is a vector $(pk, \mathfrak{m}, \rho)$ and $\mathfrak{v}$ parses as a vector of length $nc$. Let $\{b_1, \ldots, b_\ell\}$ be computed as per Step 1 of algorithm $\mathsf{Tally}$. We have for all $b = \mathsf{Vote}(pk', nc, v, \kappa; r)$ that $b \notin \{b_1, \ldots, b_\ell\}$ with overwhelming probability, since such an occurrence would imply a contradiction: $\{b_1, \ldots, b_\ell\}$ is not the largest subset of $\mathfrak{bb}$ satisfying the conditions in Step 1 of algorithm $\mathsf{Tally}$, because $b$ is a pair and $\mathsf{VerCiph}((pk, b_i[1]), b_i[2], \kappa) = 1$, with overwhelming probability, but $b \notin \{b_1, \ldots, b_\ell\}$. It follows with overwhelming

probability that

$$\textit{correct-outcome}(pk, nc, \mathfrak{bb}, \kappa) = \textit{correct-outcome}(pk, nc, \{b_1, \ldots, b_\ell\}, \kappa) \quad (1)$$

A proof of (1) follows from the definition of function *correct-outcome*. If $\{b_1, \ldots, b_\ell\} = \emptyset$, then $\mathfrak{v}$ and $\textit{correct-outcome}(pk, nc, \{b_1, \ldots, b_\ell\}, \kappa)$ are zero-filled vectors of length $nc$, with overwhelming probability, hence, soundness is satisfied. Otherwise, we proceed as follows.

By definition of $\mathsf{Verify}$, we have $\mathsf{VerKey}((\kappa, pk, \mathfrak{m}), \rho, \kappa) = 1$, hence, by simulation sound extractability, public key $pk$ is an output of $\mathsf{Gen}$, with overwhelming probability. Moreover, we have for each $i \in \{1, \ldots, \ell\}$ that $\mathsf{VerCiph}((pk, b_i[1]), b_i[2], \kappa) = 1$, hence, by simulation sound extractability, there exists a message $v_i \in \mathfrak{m}$ and coins $r_i$ such that

$$b_i[1] = \mathsf{Enc}(pk, v_i; r_i)$$

and $b_i[2]$ is an output of $\mathsf{ProveCiph}((pk, b_i[1]), (v_i, r_i), \kappa)$, with overwhelming probability.

By Step 2 of algorithm $\mathsf{Verify}$, proof $pf$ parses as a vector $(\mathbf{bb}, pf_1, \mathbf{W}, pf_2)$ such that $\mathsf{VerMix}((pk, (b_1[1], \ldots, b_\ell[1]), \mathbf{bb}), pf_1, \kappa) = 1$. It follows from simulation sound extractability that there exists a permutation $\chi$ on $\{1, \ldots, \ell\}$ and a vector of coins $\mathbf{r}$ such that $\mathbf{bb}$ is a vector of length $\ell$ and for each $i \in \{1, \ldots, \ell\}$ we have

$$\mathbf{bb}[i] = b_{\chi(i)}[1] \otimes \mathsf{Enc}(pk, \mathfrak{e}; \mathbf{r}[i]),$$

where $\mathfrak{e}$ is an identity element of $\Pi$'s message space with respect to $\odot$, with overwhelming probability. Although public key $pk$ might not have been constructed using coins chosen uniformly at random, we nevertheless, with overwhelming probability, have for each $i \in \{1, \ldots, \ell\}$ that

$$\mathbf{bb}[i] = \mathsf{Enc}(pk, v_{\chi(i)}; r_{\chi(i)} \oplus \mathbf{r}[i])$$

because $\Pi$ is perfectly homomorphic and $\mathfrak{e}$ is an identity element.

Let $\mathfrak{v}' = \textit{correct-outcome}(pk, nc, \{b_1, \ldots, b_\ell\}, \kappa)$. Given that $\{b_1, \ldots, b_\ell\}$ is a set of pairs, error symbol $\perp$ is not an element of that set. Hence, by definition of function *correct-outcome*, we have for each $v \in \{1, \ldots, nc\}$ that $\exists^{=\mathfrak{v}'[v]} b \in \{b_1, \ldots, b_\ell\} : \exists r : b = \mathsf{Vote}(pk, v, nc, \kappa; r)$, moreover, $\exists^{=\mathfrak{v}'[v]} j \in \{1, \ldots, \ell\} : v = v_j$ and, since $\chi$ is a permutation on $\{1, \ldots, \ell\}$,

$$\exists^{=\mathfrak{v}'[v]} j \in \{1, \ldots, \ell\} : v = v_{\chi(j)} \quad (2)$$

By Step 3 of algorithm $\mathsf{Verify}$, we have $\mathbf{W}$ and $pf_2$ are vectors of length $\ell$ such that $\bigwedge_{i=1}^{\ell} \mathsf{VerDec}((pk, \mathbf{bb}[i], \mathbf{W}[i]), pf_2[i], \kappa) = 1$ and $\bigwedge_{v=1}^{nc} \exists^{=\mathfrak{v}[v]} i \in \{1, \ldots, \ell\} : v = \mathbf{W}[i]$. Hence, by simulation sound extractability, we have for each $v \in \{1, \ldots, nc\}$ that

$$\bigwedge_{v=1}^{nc} \exists^{=\mathfrak{v}[v]} i \in \{1, \ldots, \ell\} : v = \mathsf{Dec}(sk, \mathsf{Enc}(pk, v_{\chi(i)}; r_{\chi(i)} \oplus \mathbf{r}[i]))$$

with overwhelming probability. Although ciphertexts $\mathbf{bb}[1], \ldots, \mathbf{bb}[\ell]$ may not have been constructed using coins chosen uniformly at random nor using a public key that was constructed using coins chosen uniformly at random, we nonetheless, with overwhelming probability, have for each $i \in \{1, \ldots, \ell\}$ that $\mathsf{Dec}(sk, \mathsf{Enc}(pk, v_{\chi(i)}; r_{\chi(i)} \oplus \mathbf{r}[i])) = v_{\chi(i)}$, because $\Pi$ is perfectly correct. Thus,

$$\exists^{=\mathfrak{v}[v]} j \in \{1, \ldots, \ell\} : v = v_{\chi(j)} \tag{3}$$

with overwhelming probability. Soundness follows from (1)–(3), which concludes our proof. □

## References


1. Adida, B.: Helios: Web-based Open-Audit Voting. In: USENIX Security'08: 17th USENIX Security Symposium. pp. 335–348. USENIX Association (2008)
2. Adida, B., Marneffe, O., Pereira, O., Quisquater, J.: 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)
3. Alvarez, R.M., Hall, T.E.: Electronic Elections: The Perils and Promises of Digital Democracy. Princeton University Press (2010)
4. Bernhard, D., Pereira, O., Warinschi, B.: 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. LNCS, vol. 7658, pp. 626–643. Springer (2012)
5. Bowen, D.: Secretary of State Debra Bowen Moves to Strengthen Voter Confidence in Election Security Following Top-to-Bottom Review of Voting Systems. California Secretary of State, press release DB07:042 `http://admin.cdn.sos.ca.gov/press-releases/prior/2007/DB07_111.pdf` (August 2007)
6. Bulens, P., Giry, D., Pereira, O.: Running Mixnet-Based Elections with Helios. In: EVT/WOTE'11: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections. USENIX Association (2011)
7. Bundesverfassungsgericht (Germany's Federal Constitutional Court): Use of voting computers in 2005 Bundestag election unconstitutional (March 2009), press release 19/2009 `http://www.bundesverfassungsgericht.de/en/press/bvg09-019en.html`
8. Chang-Fong, N., Essex, A.: 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. pp. 324–335. ACM Press (2016)
9. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Election Verifiability for Helios under Weaker Trust Assumptions. In: ESORICS'14: 19th European Symposium on Research in Computer Security. LNCS, vol. 8713, pp. 327–344. Springer (2014)
10. Cortier, V., Galindo, D., Küsters, R., Mueller, J., Truderung, T.: SoK: Verifiability Notions for E-Voting Protocols. In: S&P'16: 37th IEEE Symposium on Security and Privacy. pp. 779–798. IEEE Computer Society (2016)
11. Cortier, V., Smyth, B.: Attacking and fixing Helios: An analysis of ballot secrecy. Journal of Computer Security 21(1), 89–148 (2013)

12. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
13. Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO'86: 6th International Cryptology Conference. LNCS, vol. 263, pp. 186–194. Springer (1987)
14. Groth, J.: Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In: ASIACRYPT'02: 12th International Conference on the Theory and Application of Cryptology and Information Security. LNCS, vol. 4284, pp. 444–459. Springer (2006)
15. Gumbel, A.: Steal This Vote: Dirty Elections and the Rotten History of Democracy in America. Nation Books (2005)
16. Jones, D.W., Simons, B.: Broken Ballots: Will Your Vote Count?, CSLI Lecture Notes, vol. 204. Center for the Study of Language and Information, Stanford University (2012)
17. Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y. (eds.) Towards Trustworthy Elections: New Directions in Electronic Voting, LNCS, vol. 6000, pp. 37–63. Springer (2010)
18. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman & Hall/CRC (2007)
19. Kiayias, A., Zacharias, T., Zhang, B.: End-to-end verifiable elections in the standard model. In: EUROCRYPT'15: 34th International Conference on the Theory and Applications of Cryptographic Techniques. LNCS, vol. 9057, pp. 468–498. Springer (2015)
20. Kohno, T., Stubblefield, A., Rubin, A.D., Wallach, D.S.: Analysis of an Electronic Voting System. In: S&P'04: 25th Security and Privacy Symposium. pp. 27–40. IEEE Computer Society (2004)
21. Küsters, R., Truderung, T., Vogt, A.: Accountability: Definition and relationship to verifiability. In: CCS'10: 17th ACM Conference on Computer and Communications Security. pp. 526–535. ACM Press (2010)
22. Küsters, R., Truderung, T., Vogt, A.: Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In: S&P'11: 32nd IEEE Symposium on Security and Privacy. pp. 538–553. IEEE Computer Society (2011)
23. Küsters, R., Truderung, T., Vogt, A.: Clash Attacks on the Verifiability of E-Voting Systems. In: S&P'12: 33rd IEEE Symposium on Security and Privacy. pp. 395–409. IEEE Computer Society (2012)
24. Küsters, R., Truderung, T., Vogt, A.: Accountability: Definition and relationship to verifiability. Cryptology ePrint Archive, Report 2010/236 (version 20150202:163211) (2015)
25. Lijphart, A., Grofman, B.: Choosing an electoral system: Issues and Alternatives. Praeger (1984)
26. Meyer, M., Smyth, B.: An attack against the helios election system that violates eligibility. arXiv, Report 1612.04099 (2016)
27. American Convention on Human Rights, "Pact of San Jose, Costa Rica" (1969)
28. Document of the Copenhagen Meeting of the Conference on the Human Dimension of the CSCE (1990)
29. Quaglia, E.A., Smyth, B.: Secret, verifiable auctions from elections. Cryptology ePrint Archive, Report 2015/1204 (2017)
30. Quaglia, E.A., Smyth, B.: A short introduction to secrecy and verifiability for elections. arXiv, Report 1702.03168 (2017)

31. Saalfeld, T.: On Dogs and Whips: Recorded Votes. In: Döring, H. (ed.) Parliaments and Majority Rule in Western Europe, chap. 16. St. Martin's Press (1995)
32. Schweikardt, N.: Arithmetic, first-order logic, and counting quantifiers. ACM Transactions on Computational Logic 6(3), 634–671 (Jul 2005)
33. Smyth, B.: First-past-the-post suffices for ranked voting (2017), https://bensmyth.com/publications/2017-FPTP-suffices-for-ranked-voting/
34. Smyth, B., Frink, S., Clarkson, M.R.: Election Verifiability: Cryptographic Definitions and an Analysis of Helios and JCJ. Cryptology ePrint Archive, Report 2015/233 (version 20170111:122701) (2017)
35. Tsoukalas, G., Papadimitriou, K., Louridas, P., Tsanakas, P.: From Helios to Zeus. Journal of Election Technology and Systems 1(1) (2013)
36. UK Electoral Commission: Key issues and conclusions: May 2007 electoral pilot schemes (May 2007), http://www.electoralcommission.org.uk/elections/pilots/May2007
37. Universal Declaration of Human Rights (1948)