

# 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 [3,13,24,30]. Choices should be made by voters with equal influence, and this must be ensured by voting systems [26,27,39]. 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, 14, 18, 38]. 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. Applicable definitions include [9, 10, 15, 17, 19, 20, 35, 36] and we build upon work by Smyth, Frink & Clarkson [35]. (The relative merits of definitions are considered in Section 2.) They present game-based security definitions in the computational model of cryptography [16], 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). Definitions of cryptographic primitives, security definitions and proofs are deferred to an accompanying technical report [34].

## 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.<sup>1</sup>

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

**Setup**, denoted  $(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa)$ ,<sup>2</sup> is run by the tallier. 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  $mc$ .

**Vote**, denoted  $b \leftarrow \text{Vote}(pk, v, nc, \kappa)$ , 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  $1, \dots, nc$  of candidates. The algorithm outputs a ballot  $b$  or error symbol  $\perp$ .

**Tally**, denoted  $(\mathbf{v}, pf) \leftarrow \text{Tally}(sk, \mathbf{bb}, nc, \kappa)$ , is run by the tallier. It takes as input a private key  $sk$ , a bulletin board  $\mathbf{bb}$ , some number of candidates  $nc$ , and a security parameter  $\kappa$ , where  $\mathbf{bb}$  is a set. And outputs an election

<sup>1</sup> 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 [32]. 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.

<sup>2</sup> Let  $A(x_1, \dots, x_n; r)$  denote the output of probabilistic algorithm  $A$  on inputs  $x_1, \dots, x_n$  and random coins  $r$ . Let  $A(x_1, \dots, x_n)$  denote  $A(x_1, \dots, x_n; r)$ , where  $r$  is chosen uniformly at random. And let  $\leftarrow$  denote assignment.

outcome  $\mathbf{v}$  and a non-interactive tallying proof  $pf$  demonstrating that the outcome corresponds to votes expressed in ballots on the bulletin board. The election outcome  $\mathbf{v}$  should be a vector of length  $nc$  such that  $\mathbf{v}[v]$  indicates the number of votes for candidate  $v$ .

**Verify**, denoted  $s \leftarrow \text{Verify}(pk, \mathbf{bb}, nc, \mathbf{v}, pf, \kappa)$ , is run to audit an election. It takes as input a public key  $pk$ , a bulletin board  $\mathbf{bb}$ , some number of candidates  $nc$ , an election outcome  $\mathbf{v}$ , 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  $\text{negl}$ , such that for all security parameters  $\kappa$ , integers  $nb$  and  $nc$ , and votes  $v_1, \dots, v_{nb} \in \{1, \dots, nc\}$ , it holds that, given a zero-filled vector  $\mathbf{v}$  of length  $nc$ , we have:

$$\begin{aligned} & \Pr[(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); \\ & \quad \text{for } 1 \leq i \leq nb \text{ do} \\ & \quad \left[ \begin{array}{l} b_i \leftarrow \text{Vote}(pk, v_i, nc, \kappa); \\ \mathbf{v}[v_i] \leftarrow \mathbf{v}[v_i] + 1; \end{array} \right. \\ & \quad (\mathbf{v}', pf) \leftarrow \text{Tally}(sk, \{b_1, \dots, b_{nb}\}, nc, \kappa) : \\ & \quad nb \leq mb \wedge nc \leq mc \Rightarrow \mathbf{v} = \mathbf{v}' ] > 1 - \text{negl}(\kappa). \end{aligned}$$

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  $\text{negl}$ , such that for all security parameters  $\kappa$ , we have  $\Pr[(pk, nc, v, v') \leftarrow \mathcal{A}(\kappa); b \leftarrow \text{Vote}(pk, nc, v, \kappa); b' \leftarrow \text{Vote}(pk, nc, v', \kappa) : b = b' \wedge b \neq \perp \wedge b' \neq \perp] \leq \text{negl}(\kappa)$ .

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.<sup>3</sup> 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.

**Definition 3 (Completeness [35]).** An election scheme (**Setup**, **Vote**, **Tally**, **Verify**) satisfies *completeness*, if for all probabilistic polynomial-time adversaries

<sup>3</sup> Quaglia & Smyth [29] provide a tutorial-style introduction to the individual and universal verifiability definitions by Smyth, Frink & Clarkson, and Smyth [33] provides a detailed technical introduction.

$\mathcal{A}$ , there exists a negligible function  $\text{negl}$ , such that for all security parameters  $\kappa$ , we have  $\Pr[(pk, sk, mb, mc) \leftarrow \text{Setup}(\kappa); (\mathbf{bb}, nc) \leftarrow \mathcal{A}(pk, \kappa); (\mathbf{v}, pf) \leftarrow \text{Tally}(sk, \mathbf{bb}, nc, \kappa) : |\mathbf{bb}| \leq mb \wedge nc \leq mc \Rightarrow \text{Verify}(pk, \mathbf{bb}, nc, \mathbf{v}, pf, \kappa) = 1] > 1 - \text{negl}(\kappa)$ .

Smyth, Frink & Clarkson capture correct election outcomes using function *correct-outcome*, which is defined such that  $\text{correct-outcome}(pk, nc, \mathbf{bb}, \kappa)[v] = \ell$  iff  $\exists^{\ell} b \in \mathbf{bb} \setminus \{\perp\} : \exists r : b = \text{Vote}(pk, v, nc, \kappa; r)$ ,<sup>4</sup> where  $\text{correct-outcome}(pk, nc, \mathbf{bb}, \kappa)$  is a vector of length  $nc$  and  $1 \leq v \leq nc$ . Hence, component  $v$  of vector  $\text{correct-outcome}(pk, nc, \mathbf{bb}, \kappa)$  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.

**Definition 4 (Injectivity).** An election scheme  $(\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$  satisfies *injectivity*,<sup>5</sup> if for all probabilistic polynomial-time adversaries  $\mathcal{A}$ , security parameters  $\kappa$  and computations  $(pk, nc, v, v') \leftarrow \mathcal{A}(\kappa); b \leftarrow \text{Vote}(pk, v, nc, \kappa); b' \leftarrow \text{Vote}(pk, v', nc, \kappa)$  such that  $v \neq v' \wedge b \neq \perp \wedge b' \neq \perp$ , we have  $b \neq b'$ .

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

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

*Comparison with other verifiability definitions.* Other definitions of verifiability exist. In particular, definitions have been proposed by Juels, Catalano & Jakobsson [15], Cortier *et al.* [9] and Kiayias, Zacharias & Zhang [17]. 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.* [20–22] 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.* [23, §5.2] and by 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

<sup>4</sup> Function *correct-outcome* uses a *counting quantifier* [31] denoted  $\exists^{\ell}$ . 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.

<sup>5</sup> 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.

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.<sup>6</sup>

### 3 Encryption ensures individual verifiability & injectivity

Definitions of individual verifiability and injectivity only focus on properties of algorithm `Vote`, hence, to prove these properties for election scheme  $(\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$ , it suffices to prove the existence of an election scheme  $(\text{Setup}', \text{Vote}, \text{Tally}', \text{Verify}')$  satisfying both properties. We demonstrate the existence of such schemes by coupling election scheme `Enc2Vote` with proofs of correct key generation.<sup>7,8</sup>

**Definition 7** ( $\text{Enc2Vote}^+$ ). Suppose  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  is an asymmetric encryption scheme,  $\Sigma$  is a sigma protocol that proves correct key generation, and  $\mathcal{H}$  is a hash function. Let  $\text{FS}(\Sigma, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$ .<sup>9</sup> We define  $\text{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H}) = (\text{Setup}, \text{Vote}, \text{Tally}, \text{Verify})$  such that:

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

<sup>6</sup> 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.

<sup>7</sup> Election scheme  $\text{Enc2Vote}^*$  (§1) couples `Enc2Vote` with proofs of correct key generation *and* proofs of correct decryption, hence, it is distinguished from schemes produced by  $\text{Enc2Vote}^+$ . This distinction enables  $\text{Enc2Vote}^*$  to satisfy individual and universal verifiability, whereas  $\text{Enc2Vote}^+$  cannot produce schemes satisfying universal verifiability.

<sup>8</sup> Election scheme  $\text{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H})$  adopts the setup algorithm formalised by Smyth, Frink & Clarkson for Helios [35, Appendix C].

<sup>9</sup> Let  $\text{FS}(\Sigma, \mathcal{H})$  denote the non-interactive proof system derived by application of the Fiat-Shamir transformation to sigma protocol  $\Sigma$  and hash function  $\mathcal{H}$ .

To ensure  $\text{Enc2Vote}^+(II, \Sigma, \mathcal{H})$  is an election scheme, we require asymmetric encryption scheme  $II$  to produce distinct ciphertexts with overwhelming probability. Hence, we restrict the class of asymmetric encryption schemes used to instantiate  $\text{Enc2Vote}^+$ .

**Lemma 1.** *Let  $II = (\text{Gen}, \text{Enc}, \text{Dec})$  be an asymmetric encryption scheme,  $\Sigma$  be a sigma protocol that proves correct key generation, and  $\mathcal{H}$  be a hash function. We have  $\text{Enc2Vote}^+(II, \Sigma, \mathcal{H})$  is an election scheme if for all security parameters  $\kappa$  and messages  $m$  and  $m'$ , we have  $\Pr[(pk, sk, \mathbf{m}) \leftarrow \text{Gen}(\kappa); c \leftarrow \text{Enc}(pk, m); c' \leftarrow \text{Enc}(pk, m') : m, m' \in \mathbf{m} \Rightarrow c \neq c'] > 1 - \text{negl}(\kappa)$ .*

Our correctness proof (Lemma 1) and all further proofs appear in [34, Appendix B].

Security 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.<sup>10</sup> But, such security 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, security properties are insufficient to ensure election scheme  $\text{Enc2Vote}^+(II, \Sigma, \mathcal{H})$  satisfies individual verifiability and injectivity. Nonetheless, given that the scheme's  $\text{Vote}$  algorithm checks correct key generation, it suffices that ciphertexts do not collide for correctly generated keys.

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

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

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

Our proof of Proposition 2 [34, Appendix B.2] follows immediately from our preconditions. It is nevertheless useful, because the preconditions are defined over encryption scheme  $II$  and proof system  $\text{FS}(\Sigma, \mathcal{H})$ , rather than election scheme  $\text{Enc2Vote}^+(II, \Sigma, \mathcal{H})$ , 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  $(\text{ProveKey}, \text{VerKey})$  be a non-interactive proof system that proves correct key generation. The proof system *checks El Gamal parameters*,

<sup>10</sup> Correctness of asymmetric encryption schemes only ensures ciphertexts do not collide for distinct plaintexts.

if for all security parameters  $\kappa$ , public keys  $pk$ , messages spaces  $\mathbf{m}$ , and proofs  $\rho$ , we have  $\text{VerKey}((\kappa, pk, \mathbf{m}), \rho, \kappa) = 1$  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  $\mathbf{m} = \{1, \dots, p - 1\}$ .

**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  $\text{FS}(\Sigma, \mathcal{H})$  checks El Gamal parameters. We have  $\text{Enc2Vote}^+(\Pi, \Sigma, \mathcal{H})$  satisfies individual verifiability and injectivity.*

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.<sup>11</sup> Tsoukalas *et al.* [37] released *Zeus* as a fork of Helios 3.1.4 spliced with mixnet code to derive an implementation, and Yingtong Li released *helios-server-mixnet* as an extension of Zeus with threshold asymmetric encryption. 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

<sup>11</sup> 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.



parameters).<sup>12</sup> Moreover, Bernhard, Pereira & Warinschi [4] show that Helios 3.1.4 does not satisfy soundness.<sup>13,14</sup> They also demonstrate a denial of 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  $\text{Enc2Vote}^+(II, \Sigma, \mathcal{H})$  by replacing its tallying and verification algorithms,<sup>15</sup> 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.

<sup>12</sup> <https://github.com/benadida/helios-server/pull/133>, published 31 May 2016, accessed 21 Sep 2017.

<sup>13</sup> 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].

<sup>14</sup> A further soundness vulnerability is known [35], as are secrecy [11] and eligibility [25] vulnerabilities.

<sup>15</sup> The tallying and verification algorithms in Definition 9 adapt (unpublished) algorithms prepared by Quaglia & Smyth in the context of [28]. Quaglia & Smyth have since incorporated these adaptations into their work to take advantage of the results presented in this manuscript.

**Definition 9.** Suppose  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  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  $\text{FS}(\Sigma_1, \mathcal{H}) = (\text{ProveKey}, \text{VerKey})$  and  $\text{FS}(\Sigma_2, \mathcal{H}) = (\text{ProveCiph}, \text{VerCiph})$ . Moreover, let  $\pi(\Pi, \Sigma_2, \mathcal{H}) = (\text{Gen}, \text{Enc}', \text{Dec}')$  be an asymmetric encryption scheme such that:

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

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

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

1. *Remove invalid ballots.* Let  $\{b_1, \dots, b_\ell\}$  be the largest subset of  $\mathbf{bb}$  such that for all  $1 \leq i \leq \ell$  we have  $b_i$  is a pair and  $\text{VerCiph}((pk, b_i[1]), b_i[2], \kappa) = 1$ . If  $\{b_1, \dots, b_\ell\} = \emptyset$ , then output  $(\mathbf{v}, \perp)$ .
2. *Mix.* Select a permutation  $\chi$  on  $\{1, \dots, \ell\}$  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**
  - $\mathbf{bb}[i] \leftarrow b_{\chi(i)}[1] \otimes \text{Enc}(pk, \mathbf{e}; \mathbf{r}[i]);$
  - $pf_1 \leftarrow \text{ProveMix}((pk, (b_1[1], \dots, b_\ell[1]), \mathbf{bb}), (\mathbf{r}, \chi), \kappa);$
  - where  $\mathbf{e}$  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**
  - $\mathbf{W}[i] \leftarrow \text{Dec}(sk, \mathbf{bb}[i]);$
  - $pf_2[i] \leftarrow \text{ProveDec}((pk, \mathbf{bb}[i], \mathbf{W}[i]), sk, \kappa);$
  - if**  $1 \leq \mathbf{W}[i] \leq nc$  **then**
  - $\mathbf{v}[\mathbf{W}[i]] \leftarrow \mathbf{v}[\mathbf{W}[i]] + 1;$

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

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

1. *Remove invalid ballots.* Compute  $\{b_1, \dots, b_\ell\}$  as per Step 1 of algorithm  $\text{Tally}$ . If  $\{b_1, \dots, b_\ell\} = \emptyset$  and  $\mathbf{v}$  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  $\text{VerMix}((pk, (b_1[1], \dots, b_\ell[1]), \mathbf{bb}), pf_1, \kappa) = 1$ .
3. *Check decryption.* Check  $\mathbf{W}$  and  $pf_2$  are vectors of length  $\ell$ ,  $\bigwedge_{i=1}^{\ell} \text{VerDec}(pk, \mathbf{bb}[i], \mathbf{W}[i], pf_2[i], \kappa) = 1$ , and  $\bigwedge_{v=1}^{nc} \exists^{=v} i \in \{1, \dots, \ell\} : v = \mathbf{W}[i]$ .

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 and  $\Pi$  satisfies the condition in Lemma 1. We have  $\text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$  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  $1, \dots, nc$ , a Helios-Mixnet ballot contains an encryption of  $v$ , whereas a Helios ballot contains ciphertexts  $c_1, \dots, c_{nc-1}$  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  $\text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$  and  $\text{Enc2Vote}^+(\pi(\Pi, \Sigma_2, \mathcal{H}), \Sigma_1, \mathcal{H})$  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  $\text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$  satisfies individual verifiability and injectivity.*

To evaluate whether universal verifiability is satisfied, it remains to consider completeness and soundness.<sup>16</sup>

**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  $\text{HeliosM}(\Pi,$*

<sup>16</sup> Our proof of Theorem 7 uses a result by Bernhard *et al.* [4] that shows non-interactive proof systems derived by application of the Fiat-Shamir transformation satisfy zero-knowledge, assuming the underlying sigma protocols satisfy *special soundness* and *special honest-verifier zero-knowledge*. We will not need the details of those properties, so we omit formal definitions; see Bernhard *et al.* for formalisations.

$\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H}$ ) 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  $\text{HeliosM}(\Pi, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \mathcal{H})$  satisfies injectivity. The election scheme satisfies soundness.

Theorem 7 requires perfect correctness, rather than computational correctness,<sup>17</sup> 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. They promptly adopted and deployed the proposed fix,<sup>18</sup> 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, who confirmed soundness is not satisfied with a  $(n, n)$ -threshold, but is for other thresholds because code from the PloneVote cryptographic library is used rather than code from Helios, and has since adopted and deployed the proposed fix.<sup>19</sup>

## 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 useful feedback that helped improve this paper. I am also grateful to Ying-tong Li (developer of helios-server-mixnet) and to Georgios Tsoukalas & Panos Louridas (developers of Zeus) for discussions about their voting systems.

## References

1. Adida, B.: Helios: Web-based Open-Audit Voting. In: USENIX Security'08: 17th USENIX Security Symposium. pp. 335–348. USENIX Association (2008)

<sup>17</sup> Properties such as correctness are typically required to hold with overwhelming probability. A property is perfect if the probability is 1.

<sup>18</sup> See commitments [d2653d4](#) (9 Oct 2017), [4fddcd3](#) (11 Oct 2017), and [aab1b6f](#) (9 Oct 2017), accessed 20 Dec 2017.

<sup>19</sup> See commitment [9af7674](#) (25 Dec 2017).

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 (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: Use of voting computers in 2005 Bundestag election unconstitutional (March 2009), press release 19/2009
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. Gumbel, A.: *Steal This Vote: Dirty Elections and the Rotten History of Democracy in America*. Nation Books (2005)
14. 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)
15. 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)
16. Katz, J., Lindell, Y.: *Introduction to Modern Cryptography*. Chapman & Hall/CRC (2007)
17. 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)
18. 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)

19. Kremer, S., Ryan, M.D., Smyth, B.: Election verifiability in electronic voting protocols. In: ESORICS'10: 15th European Symposium on Research in Computer Security. LNCS, vol. 6345, pp. 389–404. Springer (2010)
20. 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)
21. 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)
22. 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)
23. Küsters, R., Truderung, T., Vogt, A.: Accountability: Definition and relationship to verifiability. Cryptology ePrint Archive, Report 2010/236 (version 20150202:163211) (2015)
24. Lijphart, A., Grofman, B.: Choosing an electoral system: Issues and Alternatives. Praeger (1984)
25. Meyer, M., Smyth, B.: An attack against the helios election system that exploits re-voting. arXiv, Report 1612.04099 (2017)
26. Organization for Security and Co-operation in Europe: Document of the Copenhagen Meeting of the Conference on the Human Dimension of the CSCE (1990)
27. Organization of American States: American Convention on Human Rights, “Pact of San Jose, Costa Rica” (1969)
28. Quaglia, E.A., Smyth, B.: Secret, verifiable auctions from elections. Cryptology ePrint Archive, Report 2015/1204 (2017)
29. Quaglia, E.A., Smyth, B.: A short introduction to secrecy and verifiability for elections. arXiv, Report 1702.03168 (2017)
30. 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)
31. Schweikardt, N.: Arithmetic, first-order logic, and counting quantifiers. ACM Transactions on Computational Logic 6(3), 634–671 (Jul 2005)
32. Smyth, B.: First-past-the-post suffices for ranked voting (2017), <https://bensmyth.com/publications/2017-FPTP-suffices-for-ranked-voting/>
33. Smyth, B.: A foundation for secret, verifiable elections (2018), <https://bensmyth.com/publications/2018-secrecy-verifiability-elections-tutorial/>
34. Smyth, B.: Verifiability of helios mixnet. Cryptology ePrint Archive, Report 2018/017 (2018)
35. 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)
36. Smyth, B., Ryan, M.D., Kremer, S., Kourjeh, M.: 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. LNCS, vol. 6186, pp. 165–182. Springer (2010)
37. Tsoukalas, G., Papadimitriou, K., Louridas, P., Tsanakas, P.: From Helios to Zeus. Journal of Election Technology and Systems 1(1) (2013)
38. UK Electoral Commission: Key issues and conclusions: May 2007 electoral pilot schemes (May 2007),
39. United Nations: Universal Declaration of Human Rights (1948)