Applying Divertibility to Blind Ballot Copying in the Helios Internet Voting System

Yvo Desmedt and Pyrros Chaidos*

University College London, UK

Abstract. Cortier & Smyth have explored ballot copying in the Helios e-voting platform as an attack against privacy. They also pointed out that their approach to ballot copying could be detected by a modified Helios. We revisit ballot copying from a different viewpoint: as a tool to prevent vote diffusion (the division of votes among multiple weak candidates) and to lessen the effect of established voting blocs. Our approach is based on *blinding* the ballot casting protocol to create an undetectable copy. A willing voter can cooperate with a prospective copier, helping the copier produce a blinded copy of his ballot without revealing his vote. We prove that Helios is unable to detect the copying. The possibility of such cooperation between voters is manifested only in internet voting and as such is a fundamental difference between internet and booth voting.

1 Introduction

Electronic voting, suggested by Chaum in [12] is one of the more important areas of computer security. It is identified as such in the "Four Grand Challenges in Trustworthy Computing" report [14]:

> There are many new systems planned or currently under design that have significant societal impact, and there is a high probability that we will come to rely on these systems immediately upon their deployment. Among these systems are electronic voting systems, ... A grand research challenge is to ensure that these systems are highly trustworthy despite being attractive targets for attackers.

The move towards electronic voting is justified by factors such as convenience, accessibility and ease of use but more importantly by the existence of provable security properties unavailable to conventional systems, for example *universal verifiability* [39,17] which satisfies the need for transparent elections by enabling election participants as well as outsiders to effectively audit an election. However, even as internet voting systems have been deployed alongside paper ballots in local as well as parliamentary elections [25,29,30], the security of electronic voting systems in general has been often found lacking [4,34].

* Pyrros Chaidos was supported by an EPSRC scholarship (EP/G037264/1 – Security Science DTC). A preliminary version of this work was presented at the CRYPTO 2011 rump session.

S. Foresti, M. Yung, and F. Martinelli (Eds.): ESORICS 2012, LNCS 7459, pp. 433–450, 2012.

Helios [1] is a state of the art web-based, universally verifiable Internet voting system. To facilitate universal verifiability, Helios ballots are encrypted and public. Ballots are cast over the Internet via a web browser. Since users have full control over the ballots they submit, Helios is susceptible to coercion, and is thus best suited for use in low-coercion environments. This might appear to limit its use to "low-stakes" elections but may not necessarily be the case. In Estonian parliamentary elections [25] for example, the ability of overwriting a vote with a later one is used in place of coercion resistance. As such, it is conceivable that a system like Helios might be used in a high profile election.

Even though Helios has been based on 30 years of sound cryptographic primitives, previous works have described attacks against Helios compromising both secrecy [15] and correctness [24]. The former, presented by Cortier & Smythe in the 2011 Computer Security Foundations Symposium exploits the lax checking of Helios against duplicate votes. In light of these attacks, security using an add-on approach may be unavoidable even for systems designed with security built-in. Our work has a different goal though: blinded ballot copying. A *blinded* copy of a ballot is a copy that cannot be detected as such. Instead of a forced relationship between coerced and coercer, this form of ballot copying relies on the cooperation of both parties, and is based on trust rather than threats or bribes. This demonstrates how an unspecified property of Helios (the ability to create blinded copies of votes) can be expanded upon to build a secondary system on top of it. The potential for this was also mentioned in [7], independently of [21].

Assume that Alice, Bob, Carol and Dianne are coworkers. Carol and Dianne are candidates for the "employee of the year award". Bob has recently returned from a project abroad and is unsure about the candidates. He would like to ask Alice whom he trusts. Alice does not want to reveal her choice so as not to upset the other candidate. Our goal is to provide a system where Alice (the voter) can assist Bob (the copier) in producing a copy of her ballot whilst ensuring that:

- Bob will not learn anything about Alice's vote that is not also revealed by the tally, but will know that the ballot produced by the copying system contains the same vote as Alice's.
- Alice cannot distinguish the ballot that is produced by the system from a random valid ballot. Therefore, the copier is explicitly given the option of backing out (by using his own choice instead of the copied one) undetected.
- Helios (or any observer) cannot recognise the ballot produced by the system as a copy.

Such a system would allow groups of voters to organise around a trusted figure, partly avoiding the spoiler effect [3] prevalent in plurality elections, thus increasing the weight of their vote and the possibility of obtaining a desired result (to the degree where trust in the original voter is well-deserved).

Using non-malleable encryption (instead of Elgamal) will only make our collaboration between a voter and copiers more complex, but cannot avoid it (due to secure multi-party computation). In fact the only solution seems to necessitate the use of private-key encryption (for example by using code voting) so that a potential copier is able to decrypt a copied vote, making blinded copying impossible.

Even with this caveat, this demonstrates internet voting introduces new features not foreseen by 30 years of research.

2 Background

Our work is based on modifying the protocols used by Helios so that they involve three parties (voter, copier, Helios) instead of just two. As such, we need to explain the design and operation of Helios before describing the modifications.

Helios 3.0 [1,2] is a state of the art web-based voting system, based on decades of cryptographic research. An important feature implemented by Helios is universal verifiability [39,17] : any party, even one uninvolved in the election can opt to verify the integrity of an election that uses Helios. This is achieved by making the ballots cast by each voter public, albeit encrypted with the ElGamal [23] cryptosystem.

Each ballot also contains a proof of its validity which can be verified without requiring specific knowledge or access and without revealing the contents of the ballot. The proofs of validity are based on a disjunctive version of the Chaum-Pedersen protocol [16,11] previously used in [17]. This ensures that no invalid ballots have been accepted and that no ballot tampering has taken place. The public list of ballots also guards against the election officials injecting votes from unregistered voters if the registration list is public. As the encryption scheme used is additively homomorphic, the product of all encrypted votes is an encryption of the sum of all votes. Since the encrypted ballots are all public, there is no way for a corrupt server to tamper with the product in an undetected way. The vote sum is obtained by the election trustees using threshold decryption. Each trustee is able to provide a partial decryption factor along with a proof of correctness for his individual calculations. The partial decryption factors are then combined to arrive at the decrypted result. Again, once the partial decryption factors have been made public there is no opportunity for foul play.

In this section we will analyze the parts of Helios that are relevant to this work. We will start by briefly mentioning the relevant parts of the Helios Implementation before moving to the cryptographic design. The design of Helios 3.0 is based on the ElGamal cryptosystem [23], used for encrypting votes and homomorphic tallying. It also uses disjunctive zero-knowledge proofs of equality to ensure ballot validity.

2.1 Current Helios Implementation

As mentioned in [33] Helios has 4 main components: an election builder, a voting booth, a ballot casting server and an audit server. From the perspective of impelmenting ballot copying, we are mostly concerned with the inner workings of the voting booth since we need to be able to extract data in order to capture the encryption randomness and also inject it to allow the copied and blinded ballot to be actually submitted. The ballot casting server concerns us only with respect to the tests performed against incoming ballots, according to the Helios specifications [33]. The workings of the other two components are not relevant.

Voting Booth. The Helios voting booth is a web application that reads the parameters of an election, presents the user with the questions he can vote on, encrypts his choices and calculates the appropriate proofs to construct a valid ballot and then allows the user to either audit *or* submit it. The randomness used in the encryption is only revealed if the user chooses to audit his ballot in which case he will need to create a new one before voting. This is implemented as a weak form of coercion resistance, but is easily bypassed[1] if the voter executes a JavaScript command during the preparation of the ballot.

Ballot Casting Server. After the ballot is constructed and saved as a JSON (JavaScript Object Notation) [18] object, the voting booth submits it to the ballot casting server using the HTTP POST method. The ballot casting server then checks the ballot for validity and compares it against already cast ballots and rejects is if it is identical to one.

2.2 Additive Homomorphic ElGamal

The ElGamal [23] cryptosystem is a public-key encryption system based on the Diffie–Hellman [22] key exchange protocol. ElGamal is also homomorphic and can be used with threshold decryption, both desirable properties for e-voting. Helios relies on both of these properties. The operations of ElGamal are as follows:

- **Key Generation**: Choose a large prime[2] $p = bq + 1$ such that q is also prime and $b \geq 2$. Choose an element g of Z_p^* with order q. Choose a secret key $x < q$ and let $h = g^x \bmod p$. The public key is then (p, q, g, h) and the private key (p, q, g, h, x).
- **Encryption**: Given a public key (p, q, g, h), encrypt a message $m < q$ as such: Choose a random blind $0 \leq r < q$. Let $\alpha = g^r$ and $\beta = m \cdot h^r$. The ciphertext c is then $c = (\alpha, \beta)$.
- **Decryption**: Given a private key (p, q, g, h, x) and a ciphertext $c = (\alpha, \beta)$, the decrypted message is $\mu = \alpha^{-x} \cdot \beta$

Homomorphic Property. An important property of ElGamal is that it is homomorphic: the product of two ciphertexts is a cipher text which corresponds to the product of the messages in the original ciphertexts. Let two ciphertexts c_1, c_2 be the encryptions of m_i with blind r_i for $i = 1, 2$. Then $c_1 \cdot c_2 = (\alpha_1 \cdot \alpha_2, \beta_1 \cdot \beta_2) = (g^{r_1+r_2}, m_1 \cdot m_2 \cdot h^{r_1+r_2})$ i.e. $c_1 \cdot c_2$ is the encryption of $m_1 \cdot m_2$ with blind $r_1 + r_2$.

A special case of homomorphic operation is when c_2 corresponds to the message $m_2 = 1$ in which case the resulting ciphertext is a re-encryption of m_1 with a different blind.

[1] In fact, early versions of Helios included a "Coerce Me!" button which revealed the encryption randomness without invalidating the ballot.

[2] In the case of Helios b is fixed to 2.

Votes. In the context of Helios, the homomorphic property is used in order to calculate an encrypted *sum* of votes from individual encrypted ballots without the need to decrypt them individualy. However, as described above, ElGamal is multiplicatively homomorphic, whereas vote tallies are sums. In order to bridge this gap, in Helios a variant of the encoding used in [17]. Cramer et al. encode votes of "no" as 1 and votes of "yes" as g. In this way, the product of n votes v_i of which m are "yes" will be $\prod_{i=1}^{n} v_i = g^m$ i.e. the log of the product will be the sum of the votes i.e. the scheme is additively homomorphic. Most elections have more options than "yes" and "no", so Helios models them as a series of "yes-no" questions about each option, with a limit on the number of "yes" answers equal to the number of selections allowed in the original question. For example, given a question with 3 choices, from which exactly one may be selected, a vote would be of the form:

$$V = (\alpha_0, \beta_0), (\alpha_1, \beta_1), (\alpha_2, \beta_2) \tag{1}$$

$$= (g^{r_0}, g^{m_0} \cdot h^{r_0}), (g^{r_1}, g^{m_1} \cdot h^{r_1}), (g^{r_2}, g^{m_2} \cdot h^{r_2}) \tag{2}$$

In the above vote, r_i represents the randomness used in the encryption and m_i the answers of the voter to each of the 3 options. We note that a vote of the above form might be *invalid*, for example if $m_i > 1$ for some i, or if every m_i is 1, even though the election parameters only allow the voter to choose one option. A particularly insidious voter might even have $m_0 = -100$, making his vote cancel out 100 honest votes for the first option. Helios guards against this by requiring the voter to provide a *zero-knowledge* proof of his ballot's validity. *Note:* Helios supports a threshold variant of ElGamal, but this is not relevant to this work.

2.3 Proofs of Knowledge

As seen in the above example, a voter must provide a proof that the value of his vote falls into the range permitted by the election parameters. As such, he must prove that the individual vote for each option is either a "yes" or a "no" and furthermore that the total number of "yes" votes is within the range of the allowed number of selections. In more concrete terms, the voter is asked to prove that each m_i is either 0 or 1 (an *individual proof* in Helios terminology), and that the sum $\sum_{i=0}^{n-1} m_i$ is inside the range of allowed selections as specified in the election's definition (a *total proof*).

The proofs of validity used by Helios are offline disjunctive zero knowledge proofs of equality between discrete logs. In the rest of this section, we will offer a brief overview of the underlying concepts as well as their use in Helios.

Proofs of Knowledge. *Zero knowledge proofs of knowledge* [26] are a concept related to *zero knowledge interactive proofs* [32], the difference being that in proofs of knowledge the prover is supplied with an auxiliary input called a *witness* which enables it to convince the verifier. An algorithm called a *witness extractor*

can then output that witness if given oracle access to the prover. Intuitively, in a proof of knowledge we want three things to hold: first, an honest prover given a correct witness almost always convinces the verifier. Second, if a witness extractor is allowed to "interrogate" a successful prover, he will in most cases be able to extract the witness. Third, even if the verifier behaves in a dishonest way, he learns nothing useful.

Computational Assumptions. The prover and verifier in our setting are polynomially bounded. Furthermore, we assume the the Decisional Diffie Hellman assumption [6,22] holds. This is also a requirement for the semantic security of the ElGamal scheme.

Definition 1 (Decisional Diffie Helman assumption). *Let G be a cyclic group and g a generator with prime order q. Then, given $a, b, c \in_R Z_q$ the following tuples cannot be distinguished by a polynomially bounded turing machine: (g^a, g^b, g^c),(g^a, g^b, g^{ab}). The bound is on $|q| = \lceil \log_2 q \rceil$.*

Disjunctive Proofs of Equality between Discrete Logarithms. To prove that an encrypted individual vote (α, β) is valid one must prove that either the corresponding plaintext is either $g^0 = 1$, in which case $\log_g \alpha = \log_h \beta$, or g^1 in which case $\log_g \alpha = \log_h \beta/g$. As the prover needs to prove the *disjunction* of the two statements we have a *disjunctive proof of knowledge*.

Total proofs can be carried out in the same way, the difference being that for individual proofs the range of exponents is always $[0, 1]$ whereas for total votes it ranges from the minimum number of selections to the maximum. For total proofs the ciphertext used is the homomorphic product of the individual ciphertexts.

Therefore, the main component of the proofs of knowledge used in Helios is a protocol to prove equality between discrete logs [11] along with a construction that enables the prover to prove the disjunction of many statements without revealing which one is in fact true [16].

The Chaum-Pedersen protocol [11] for discrete log equality is essentially a parallel version of the Schnorr protocol [40]. We note that the Chaum-Pedersen protocol (as well as the underlying Schnorr protocol) is only provably zero knowledge against adversaries who behave honestly. There is no known simulator for dishonest adversaries [11].

Cramer et al. provide a construction for disjunctive proofs [16,27] where the prover can prove one statement from a set and simulate proofs for the other ones, without the verifier knowing which of the subproofs are simulated. In the context of Helios, this allows the voter to indicate that the plaintext of his ballot is one out of a number of allowed values without revealing which one.

We explain the details of the construction of Cramer et al. [16] as applied to the Chaum-Pedersen protocol [11]. We take advantage of the fact that since the Chaum-Pedersen protocol is honest-verifier zero knowledge, if the voter can choose the challenge c, he can simulate the proofs as follows:

Protocol 1. Simulated Chaum-Pedersen Protocol.

Step 1 **Simulated Proof:** Choose random challenge c and response s. Let the commitments be $a = g^s/\alpha^c$ and $b = h^s/(\beta/g^v)$.
Step 2 **Partial Verification:** Check that $g^s = a \cdot \alpha^c$ and that $h^s = b \cdot (\beta/g^v)^c$.

In order to force the voter to provide at least one honest proof, he is not given complete choice of the challenges. The Verifier is allowed to specify the sum of the challenges used in the subproofs. This allows the voter to simulate all but one of the Chaum-Pedersen proofs and let the challenge of the real subproof be as a balancing factor in the sum. Suppose the voter needs to prove that the value v encoded by $(\alpha, \beta) = (g^r, h^r g^v)$ is in $[\min, \max]$. He will simulate the proofs for $i \in [\min, \max] \setminus \{v\}$ and produce a real proof for $i = v$.

Protocol 2. Disjunctive Chaum-Pedersen Protocol.

Step 1 **Prover** (Voter): For $i \in [\min, \max] \setminus \{v\}$: Choose random challenge c_i and response s_i. Let the commitments be $a_i := g^{s_i}/\alpha^{c_i}$ and $b_i := h^{s_i}/(\beta/g^i)^{c_i}$. Choose commitment (a_v, b_v) such that $(a_v, b_v) = (g^w, h^w)$ for some w.
Step 2 **Verifier** (Helios): Choose $T \in_R Z_q$.
Step 3 **Prover** (Voter): Let $c_v := T - \sum_{i \neq v} c_i$ and $s_v := rc_v + w$.
Step 4 **Verifier** (Helios): Check if $g^{s_i} \stackrel{?}{=} a \cdot \alpha^{c_i}$ and that $h^{s_i} \stackrel{?}{=} b \cdot (\beta/g^i)^{c_i}$ for $i \in [\min, \max]$. Check if $T \stackrel{?}{=} \sum_{i=min}^{max} c_i$.

Non-Interactive Proofs. For practical reasons, Helios implements the above protocol offline rather than online. This requires less communication with the Helios server and does not require the Helios server to hold the state of proof protocols in progress. This is done by way of the Fiat-Shamir heuristic [28] which replaces the random challenge issued by the verifier with a hash of the commitments. This also facilitates universal verifiability since the generation of the challenge is beyond the control of the (potentially dishonest) Helios server. The result of this modification is that the protocol can be performed entirely by the voter with the final ballot submitted to Helios for verification.

The offline version of the protocol is zero knowledge in the *random oracle model*. For zero knowledge proofs under the random oracle model, the hash function used in the protocol is assumed to be an oracle under the control of the simulator. As such, the simulator can choose the value returned by the hash function on any input with the only limitation being consistency (i.e. after setting $H(x) = y$, the simulator is not allowed to set $H(x) = y' \neq y$. The random oracle model has been criticised as [31,9] have shown that it is possible to construct protocols that are secure under the random oracle model but provably insecure in general. Nonetheless, these results have not led to a vulnerability being found in a currently used protocol.

For example, suppose we have an election with 3 options of which exactly one may be selected (as in the vote example). We follow the notation of protocol 2

in that a_i, b_i represent the commitments of a proof, s_i the solution and c_i the challenge. The individual proofs would then be of the form:

$$P_i = ((a_{i,0}, b_{i,0}, a_{i,1}, b_{i,1}), (c_{i,0}, s_{i,0}, , c_{i,1}, s_{i,1})), \text{ for } i \in \{0, 1, 2\} \tag{3}$$

And the total proof would be of the form:

$$P_\Sigma = ((a_{\Sigma,1}, b_{\Sigma,1}), (c_{\Sigma,1}, s_{\Sigma,1}))$$

To check if one of the above proofs is valid, Helios would set $T := H(a_{\min}, b_{\min}, \cdots, a_{\max}, b_{\max})$ and run the last step of protocol 2.

Note: w.l.o.g. and in the interest of readability we will limit ourselves to ballots consisting of a single encrypted "yes"-"no" vote and it's corresponding proof of validity.

Blind Signatures and Diverted Proofs. Blind signatures [10,11], involve signing a document through an intermediary (in our case, the copier) without the original signer (the voter) being able to trace the end product. Blind signatures have been suggested by Chaum [10] for use with anonymous electronic cash, where banks sign "coins" proving their authenticity but are unable to trace their use, and voting where authorities can supply signed blank ballots to authenticated voters but are then unable to track them once filled.

Divertible proofs [37,19] are a similar notion to blind signatures, but in an online setting. An intermediate party is introduced between the prover and verifier, playing the role of the verifier against the prover and that of the prover against the verifier. The intermediate is called a *warden* in some cases (for example, if he is introduced to enforce to ensure honest behaviour) or a man in the middle in others.

2.4 Related Work

Even though we do not regard our work as an actual attack against Helios, previous attacks highlight some of the techniques used as well as some of the assumptions in Helios' specification that enable data extraction and injection.

A Ballot Replay Attack. Cortier & Smyth [15] attack a voter's privacy by means of a replay attack. In the base version of their attack, a ballot is recast either verbatim or with minor differences in the representation of the signatures by a number of parties under the control of the attackers. The existing checks performed by the Hellios ballot casting server were somewhat lax. In some scenarios the additional votes for the original voter's chosen party or candidate will significantly bias the election result, thus violating privacy. The authors offer the French legislative elections as an example of such a scenario. A more complex version of their attack involves a permutation of the voter's choices making the malicious ballots slightly harder to detect. Our work is similar in that it also involves effectively replaying a vote but different in that the original voter consents to that. Also, the replayed vote *cannot* be detected as such.

An Attack against the Voter's Web-Browser. Esteghari & Desmedt [24] describe an attack which essentially installs a rootkit in the user's web-browser by exploiting a vulnerability in Adobe Reader. The rootkit then secretly changes the user's vote to a different one, and also hides any evidence of foul play. Helios, operating under the assumption that the user's browser is trustworthy, accepts the changed vote instead of the intended one.

While our work also affects the voting booth running at the user's browser, an important difference is that participation from both parties is *consensual* and not based on deception or exploits.

3 A Ballot-Blinding Protocol

We will split the description of our ballot-blinding protocol in two parts. We will first describe how a copier can re-encrypt an encrypted vote (making it indistinguishable to a random one assuming the DDH problem is hard) along with the appropriate modifications to keep the corresponding proof valid.

Note that since the randomness used in the ElGamal encryption is required to construct the real subproof it is impossible to simply copy and blind a cast ballot without extra information. On the other hand, if a voter were to publish the randomness used in his ballot to enable blinded copying he would be sacrificing his privacy! For that reason, we will describe an online protocol between a willing voter (who has already cast a ballot) and a copier. The protocol allows the copier to produce a "new" proof of knowledge for the encrypted vote.

The copier can combine the two parts: first he obtains a new (indistinguishable) proof of validity of the voter's encrypted vote and then he re-encrypts the encrypted vote making it indistinguishable as well. The result is a ballot that is equivalent to the original in that it contains the same vote but indistinguishable from it. Moreover, it does not leak the original vote.

3.1 Vote Blinding

We describe a transformation that a copier can perform to an already cast ballot that is based on re-encrypting the vote contained in the ballot. Because of the re-encryption, the proof contained in the ballot must also be modified to stay valid.

Given a vote $(\alpha, \beta) = (g^r, h^r g^v), v \in 0, 1$, a copier is able to re-encrypt it as $(\alpha', \beta') = (g^{r+z}, h^{r+z} g^v), v \in 0, 1$. To do that, he does not need knowledge of r as he can simply calculate $(\alpha', \beta') = (g^z \alpha, h^z \beta)$.

Lemma 1. *If z is chosen to be uniformly random in Z_q then (α', β') is indistinguishable from a random vote by adversaries who cannot solve the DDH problem, regardless of them knowing r or v.*

Proof. Since z is uniformly random in Z_q, it follows that $g^z \alpha$ is uniformly random in $\langle g \rangle$. Since α' is g^s for some s, $\beta' / g^v = h^s$ and $s, x = \log_g h$ are independently chosen, then $(h, g^s, \beta' / g^v)$ is a DDH problem instance which the adversary could solve if he was able to distinguish (α', β') from a random encrypted vote. □

Furthermore, if the copier has access to a valid proof for (α, β) he can transform it to a valid proof for (α', β').

Lemma 2. *If (V, P) is a valid ballot, with $V = (\alpha, \beta)$ as in (1) and $P = ((a_0, b_0, a_1, b_1), (c_0, s_0, c_1, s_1))$ as in Protocol 2, then $((g^z\alpha, h^z\beta), (a_0, b_0, a_1, b_1), (c_0, s_0 + c_0 z, c_1, s_1 + c_1 z))$ is also a valid ballot and vice versa.*

Proof. If $a_i = g^{s_i}/\alpha^{c_i}$ holds then $a_i = g^{s_i+c_i z}/(g^z\alpha)^{c_i}$ also holds. Similarly, if $b_i = h^{s_i}/(\beta/g^i)^{c_i}$ holds then $b_i = h^{s_i+c_i z}/(h^z\beta/g^i)^{c_i}$ also holds. For the opposite direction we note that re-applying the transformation for $-z$ produces the original ballot. □

The above transformation can be used as a variant of the attack described in [15] since it provides another way of replaying ballots without copying them verbatim. Nonetheless, the attack variant can be stopped in a similar way to the one suggested by Cortier and Smyth. Since the commitments a_i, b_i and challenges c_i of the proof are unchanged, a future version of Helios could defend against the attack by modifying the ballot casting server to reject votes which reuse past commitment values.

It is clear from the above discussion that blinding the entire ballot is necessary. Towards that, we describe a protocol that blinds the proofs of a ballot. The proof blinding protocol requires two assumptions: First, that original voter cooperates with the the copier and second, that the voter has access to the randomness used in encrypting his ballot. Fortunately, the second assumption can be fulfilled in the current Helios implementation.

3.2 Proof Blinding

Blinding the proof of a ballot is more involved: on one hand, creating a valid proof requires access to the randomness used in the encryption but on the other, revealing that *witness* would compromise a voter's privacy. Our solution is based on the concepts of *divertible protocols* [37,19] and *blind signatures* [10,11].

Suppose the voter has cast an encrypted vote $(\alpha, \beta) = (g^r, h^r g^v), v \in 0, 1$ with an appropriate proof, and the copier is requesting a different proof in order to copy it. Note that (α, β) is public but (r, v) is private to the voter. Since the hash function H() is public, Helios does not take part in the protocol. The notation used for the commitments is the same as in Protocol 2, but the roles of the parties are different. The voter still takes the role of the prover, but the copier takes the role of an intermediate verifier who ultimately submits the resulting ballot to Helios.

Protocol 3. Proof Blinding Protocol

Step 1 Voter: Choose $w \in_R Z_q$ and let $a_v := g^w, b_v := h^w$. Let $\lambda := 1 - v$ and choose $c_\lambda, s_\lambda \in_R Z_q$ and let $a_\lambda := g^{s_\lambda}/\alpha^{c_\lambda}$ and $b_\lambda := h^{s_\lambda}/(\beta/g^\lambda)^{c_\lambda}$. Send (a_0, b_0, a_1, b_1) as a commitment to the copier.

Step 2 Copier: Choose $\Delta_0, \Delta_1, k_0, k_1 \in_R Z_q$. Let $A_i := a_i g^{k_i}/\alpha^{\Delta_i}$, $B_i := b_i h^{k_i}/(\beta/g^i)^{\Delta_i}$ for $i = 0, 1$. Let $c := H(A_0, B_0, A_1, B_1)$ be the challenge that Helios would issue. Let $C := c - \Delta_0 - \Delta_1$, send C to voter as a challenge.

Step 3 Voter: Let $c_v := C - c_\lambda$ and let $s_v := w + rc_v$, send (c_0, s_0, c_1, s_1) to copier as a reply.

Step 4 Copier: Check that $C \stackrel{?}{=} c_0 + c_1$, $a_i \stackrel{?}{=} g^{s_i}/\alpha^{c_i}$ and $b_i \stackrel{?}{=} h^{s_i}/(\beta/g^i)^{c_i}$ for $i = 0, 1$. If yes, accept and let $C_i := c_i + \Delta_i$ and $S_i := s_i + k_i$. Let $V := ((\alpha, \beta), (A_0, B_0, A_1, B_1), (C_0, S_0, C_1, S_1))$ and send V to Helios. Otherwise, reject.

We will now examine Protocol 3 with regard to correctness, indistinguishability and security.

Correctness. We will first prove that our protocol satisfies *completeness* and (special) *soundness*.

Lemma 3. *The proof blinding protocol is complete and furthermore if an honest copier accepts then the resulting ballot V will be accepted by Helios.*

Proof. Completeness holds trivially. Indeed, we have:

- $C = c_0 + c_1$ *since the voter calculates* $c_v := C - c_\lambda$ *in Step 3.*
- *For* $i = \lambda$, *we have* $a_\lambda = g^{s_\lambda}/\alpha^{c_\lambda}$ *and* $b_\lambda = h^{s_\lambda}/(\beta/g^\lambda)^{c_\lambda}$ *from Step 1.*
- *For* $i = v$, *we must check if* $a_v \stackrel{?}{=} g^{s_v}/\alpha^{c_v} = g^{w+rc_v}/\alpha^{c_v}$ *which holds since* $a_v = g^w$ *(from Step 1) and* $\alpha^{c_v} = g^{rc_v}$. *Similarly:* $b_v \stackrel{?}{=} h^{s_v}/(\beta/g^v)^{c_v} = h^{w+rc_v}/(\beta/g^v)^{c_v}$ *holds since* $b_v = h^w$ *and* $\beta/g^v = h^r$.

For the second property, we need to show that $C_0 + C_1 = H(A_0, B_0, A_1, B_1)$ *and that given that* $a_i = g^{s_i}/\alpha^{c_i}$ *and* $b_i = h^{s_i}/(\beta/g^i)^{c_i}$ *hold (since the copier has access to a valid vote) it also holds that:* $A_i = g^{S_i}/\alpha^{C_i}$ *and* $B_i = h^{S_i}/(\beta/g^i)^{C_i}$. *This is straightforward by substituting the blinded variables* A_i, B_i, C_i, S_i *with their definitions.* □

Lemma 4. *Protocol 3 has the special soundness property.*

Proof. Suppose a voter can (given the same commitments (a_0, b_0, a_1, b_1)*) provide answers to two different challenges* C, C'. *This means that for the two answers* (c_0, s_0, c_1, s_1) *and* (c'_0, s'_0, c'_1, s'_1), *we must have* $c_i \neq c'_i$ *for at least one* $i \in \{0, 1\}$. *We will now show that such a voter can calculate a witness for the vote's validity (i.e. the encryption randomness used in encrypting the vote).:*

$$a_i = g^{s_i}/\alpha_i^c \text{ and } a_i = g^{s'_i}/\alpha^{c'_i} \text{ we have:}$$

$$g^{s_i}/\alpha^{c_i} = g^{s'_i}/\alpha^{c'_i}$$

$$g^{s_i - s'_i} = \alpha^{c_i - c'_i} \text{ thus:}$$

$$\log_g \alpha = \frac{c_i - c'_i}{s_i - s'_i}.$$

□

Indistinguishability

Lemma 5. *Given the view of the original voter, the blinded proof of knowledge* $((A_0, B_0, A_1, B_1), (C_0, S_0, C_1, S_1))$ *is unconditionally indistinguishable from a valid proof produced independently.*

Proof. We observe that $C_i = c_i + \Delta_i$ *and* $S_i = s_i + k_i$ *with* Δ_i *and* k_i *being uniformly random in* Z_q*. Thus, the challenges and responses are independent of the ones used in the original proof. For the commitments, we note that given* (C_0, S_0, C_1, S_1)*, the values of* (A_0, B_0, A_1, B_1) *are uniquely determined (because for any valid proof:* $A_i = g^{S_i}/\alpha^{C_i}$ *and* $B_i = h^{S_i}/(\beta/g^i)^{C_i}$*), so if a voter is able to distinguish* $(A_0, B_0, A_1, B_1), (C_0, S_0, C_1, S_1)$ *from an independent proof he would also be able to distinguish* (C_0, S_0, C_1, S_1)*.* □

Security. Our goal is to ensure that the blinded protocol does not leak the value of the voter's vote to one of the other parties. The proof of knowledge protocol used by Helios is based on applying the construction of Cramer et al. [16] to the Chaum-Pedersen protocol [11] for proving the equivalence of discrete logs.

Lemma 6. *Under the random oracle model, the voter-copier interaction is zero-knowledge for a copier who follows the protocol.*

Proof. Under the random oracle model, and assuming that the DDH problem is hard, we will describe a simulator for Voter-Copier interactions when the voter is honest. We note that even an honest prover needs to simulate the proof corresponding to $i = \lambda$*. The main difference is that the simulator will simulate both proofs and rely on its control of the hash function via the random oracle model to match the challenge. The simulator proceeds as follows:*

1. *For* $i \in 0, 1$ *choose* $c_i, s_i \in_R Z_q$ *and let* $a_i := g^{s_i}/\alpha^{c_i}$ *and* $b_i := h^{s_i}/(\beta/g^i)^{c_i}$*.*
2. *Choose* $\Delta_0, \Delta_1, k_0, k_1 \in_R Z_q$*. Let* $A_i := a_i g^{k_i}/\alpha^{\Delta_i}$*,* $B_i := b_i h^{k_i}/(\beta/g^i)^{\Delta_i}$ *for* $i = 0, 1$*. Set* $H(A_0, B_0, A_1, B_1) := c_0 + c_1 + \Delta_0 + \Delta_1$*. And let* $c := H(A_0, B_0, A_1, B_1)$ *Let* $C := c - \Delta_0 - \Delta_1$
3. $C_i := c_i + \Delta_i$ *and* $S_i := s_i + k_i$*.*

The communication transcript between the simulated voter and copier is then $((a_0, b_0, a_1, b_1), c, (c_0, s_0, c_1, s_1))$ *and the simulated output of the copier to Helios is* $((\alpha, \beta), (A_0, B_0, A_1, B_1), (C_0, S_0, C_1, S_1))$*. Against adversaries who cannot solve the DDH problem (and thus distinguish* $(a_i, b_i) = (g^{s_i}/\alpha^{c_i}, h^{s_i}/(\beta/g^i)^{c_i})$ *from* $(a_i, b_i) = (g^w, h^w)$*) the output of the simulator is indistinguishable to genuine transcripts and outputs, since after the commitments are issued the simulator follows the same steps as the ones taken by the copier.* □

We can also use the above simulator to prove that the protocol is also zero-knowledge with respect to the copier "interacting" with Helios. Since the Fiat-Shamir heuristic [28] is used to replace the verifier's challenge with a hash of the the prover's commitments, Helios is unable to deviate from honest behaviour.

We also note that the simulator's control of the output of the hash function is explicitly allowed under the random oracle model (see [5, Sect. 5.1]).

The copier however has the option to ignore the protocol and issue arbitrary challenges to the voter. On the other hand, achieving indistinguishability requires that the copier keeps his coin rolls private. As such, it is conceivable that a dishonest voter can craft challenges in a way that compromises the voter's privacy. Given any limited-size subset of the challenge space (the limit being polynomial in the bit-length of q) we can achieve zero-knowledge even against cheating copiers (we explore this option in the next section). As such, we follow [11] in conjecturing that even in the unrestricted case, a cheating copier gains no useful information. This conjecture is also supported by the fact that the copier's control is weakened compared to the Chaum-Pedersen protocol, since he cannot control the individual challenged but only dictate their sum.

Note: The disjunctive proof construction in [16] can provide *witness indistinguishability* [27], but in the case of the Helios disjunctive proofs there is a unique valid witness $w = (r, v)$ for every encrypted vote $(\alpha, \beta) = (g^r, h^r g^v)$. As such the witness indistinguishability property is inconsequential.

3.3 A Combined Protocol for Blinded Copying

The vote blinding transformation of Sect. 3.1 and the proof blinding protocol (Protocol 3) can each partially blind a ballot (the vote and the proof respectively). They can be easily combined to completely blind a ballot as follows: The copier executes the proof blinding protocol with the cooperation of the voter but does not submit the resulting ballot V. Instead, he proceeds to apply the vote blinding transformation to V, producing V' which he then submits to Helios.

Theorem 1. *The combined ballot copying protocol is complete, sound and zero-knowledge for honest-verifiers under the random oracle model. Furthermore, assuming the DDH assumption holds, the ballots produced are accepted by Helios and indistinguishable from random valid ballots, even for the voter.*

Proof. Completeness, soundness and honest-verifier zero-knowledge under the random oracle model are satisfied by the proof copying protocol and are not impacted by the transformation (Lemma 2). Indistinguishability holds because of Lemmas 1 and 5. □

4 A Multi-round Variant with Short Challenges

Since blinding the offline protocol does not achieve zero-knowledge, we explore a variant that can guarantee it. Furthermore, we avoid the use of the random oracle model in order to achieve a stronger proof.

The main obstruction to achieving zero-knowledge lies with the use of the Schnorr [40] protocol as the basis of the proof construction (since the Chaum-Pedersen protocol is a parallel version of Schnorr's). Our approach to guaranteeing the voter's privacy with regard to a dishonest copier is to adapt ideas

from [8] while keeping the rest of the proof construction. For this we reduce the challenge space so that $c < \log q$. Since the challenge space is now polynomial in size compared to the security factor (the bit-size of q), the protocol can be simulated thus making it zero-knowledge.

We first present the modified protocol used by a voter for submitting a vote to Helios. It is repeated t times, and a dishonest voter cannot succeed with probability greater that $\log q^{-t}$. Compared to the original, offline protocol the only difference is the challenge generation: instead of using a hash function, a (short) challenge is selected randomly.

We then *divert* the online protocol in order to achieve an equivalent result to blinding. We follow the notation used in Protocol 2 for the proof commitments and (1) for the vote.

Protocol 4. Online Ballot Verification.

Step 1. Voter: Choose $w \in_R Z_q$ and let $a_v := g^w, b_v := h^w$. Let $\lambda := 1 - v$, choose $c_\lambda, s_\lambda \in_R Z_q$ and let $a_\lambda := g^{s_\lambda}/\alpha_\lambda^c$ and $b_\lambda := h^{s_\lambda}/(\beta/g^\lambda)^{c_\lambda}$. Send (a_0, b_0, a_1, b_1) as a commitment to Helios.

Step 2. Helios: Choose $c \in_R Z_{\lceil \log q \rceil}$, send c to the voter as a challenge.

Step 3. Voter: Let $c_v := c - c_\lambda$ and let $s_v := w + rc_v$, send (c_0, s_0, c_1, s_1) to copier as reply.

Step 4. Helios: Check if $c \stackrel{?}{=} c_0 + c_1$, $a_i \stackrel{?}{=} g^{s_i}/\alpha^{c_i}$ and $b_i \stackrel{?}{=} h^{s_i}/(\beta/g^i)^{c_i}$ for $i = 0, 1$. If yes, accept, otherwise, reject.

Protocol 4 is *complete*, *sound* and *zero-knowledge*. Completeness is maintained from the original Helios protocol as only the challenge generation is different. Thus an honest voter will always be able to convince Helios. We will now prove that the protocol satisfies the special soundness property and is zero-knowledge.

Lemma 7. *Protocol 4 satisfies special soundness.*

Proof. We repeat the argument of Lemma 4: suppose a (potentially dishonest) voter can, given one set of commitments, answer two different challenges. Then he would be able to calculate $\log_g \alpha$. *Thus no dishonest verifier who does not know* $\log_g \alpha$ *has a better than* $1/\log q$ *chance to complete a round successfully.* □

Lemma 8. *Protocol 4 is zero-knowledge under the DDH assumption.*

Proof. We will describe a simulator for the online protocol.

1. *For* $i \in 0, 1$ *choose* $c_i, s_i \in_R Z_q$ *and let* $a_i := g^{s_i}/\alpha^{c_i}$ *and* $b_i := h^{s_i}/(\beta/g^i)^{c_i}$. *Send the commitments to the Verifier.*
2. *If the Verifier replies with* $c = c_0 + c_1$, *output the transcript* (a_0, b_0, a_1, b_1), $c, (c_0, s_0, c_1, s_1)$. *Otherwise, reset the Verifier and return to Step 1.*

Against verifiers who cannot solve the DDH problem, a set of simulated commitments is indistinguishable to a set of random elements, so the verifier's reply, $V(a_0, b_0, a_1, b_1)$ *will be independent of* $c = c_0 + c_1$. *As such, the simulator has*

a $1/\lceil \log q \rceil$ *chance to guess the challenge correctly in one try and thus runs in expected polynomial time. Again, under the DDH assumption the simulated transcripts are indistinguishable from normal ones.* □

We now present a way to divert the above protocol such that blinded ballot copying can take place, while achieving zero-knowledge even against dishonest copiers. As in the offline case, suppose the voter has cast an encrypted vote $(\alpha, \beta) = (g^r, h^r g^v), v \in 0, 1$ with appropriate proof, and the copier is requesting a different proof in order to copy it. We also assume that the copier has blinded the vote by re-encrypting it as $(\alpha', \beta') = (g^z \alpha, h^z \beta)$ and has presented the blinded rather than the original vote to Helios. Again, we follow the notation used in Protocol 2 for the proof commitments.

Protocol 5. Diverted Ballot Verification.

Step 1. Voter: Choose $w \in_R Z_q$ and let $a_v := g^w, b_v := h^w$. Let $\lambda := 1 - v$ and choose $c_\lambda, s_\lambda \in_R Z_q$ and let $a_\lambda := g^{s_\lambda}/\alpha_\lambda^c$ and $b_\lambda := h^{s_\lambda}/(\beta/g^\lambda)^{c_\lambda}$. Send (a_0, b_0, a_1, b_1) as a commitment to the copier.

Step 2. Copier: Choose $\Delta_0, \Delta_1, k_0, k_1 \in_R Z_q$. Let $A_i := a_i g_i^k / \alpha^{\Delta_i}$, $B_i := b_i h_i^k / (\beta/g^i)^{\Delta_i}$ for $i = 0, 1$. Send (A_0, B_0, A_1, B_1) to Helios as a commitment.

Step 3. Helios: Choose $c \in_R Z_{\lceil \log q \rceil}$, send c to copier as a challenge.

Step 4. Copier: Let $C := c - \Delta_0 - \Delta_1$, send C to voter as a challenge.

Step 5. Voter: Let $c_v := C - c_\lambda$ and let $s_v := w + r c_v$, send (c_0, s_0, c_1, s_1) to copier as a reply.

Step 6. Copier: Check if $C \stackrel{?}{=} c_0 + c_1$, $a_i \stackrel{?}{=} g^{s_i}/\alpha_i^c$ and $b_i \stackrel{?}{=} h^{s_i}/(\beta/g^i)^{c_i}$ for $i = 0, 1$. If yes, accept and let $C_i := c_i + \Delta_i$ and $S_i := s_i + k_i$. Let $S_i' := S_i + z C_i$ Send (C_0, S_0', C_1, S_1') to Helios. Otherwise, reject.

Theorem 2. *The diverted ballot verification protocol is a divertible interactive zero knowledge proof of validity for helios ballots.*

Proof. (sketch) First we prove that both interactions are zero-knowledge. Completeness transfers over from the offline blinded copying protocol since the only difference from the offline protocol is the challenge generation and vote blinding (point to Lemma for VB). Special soundness holds for both interactions (see Lemma 7). Furthermore, the simulator of Lemma 8 can be used to prove that both interactions are zero knowledge.

It is easy to see that neither the copier or Helios can calculate a witness directly as that would solve the discrete logarithm problem.

Finally, indistinguishability transfers over from the offline version. □

As the diverted ballot verification protocol is provably zero-knowledge, the transcripts cannot be used as signatures: they might be signatures for invalid ballots produced by a simulator operated by a dishonest voter and a dishonest Helios. As such, the universal verifiability property of Helios no longer holds. Such a modification would thus require trust to be placed on the bulletin board administrator, something that diverges significantly from the original design of Helios but is necessary if we want to achieve zero-knowledge and avoid using the random oracle model.

5 Conclusion and Further Work

We have described a protocol which enables voters to allow people who trust them to copy their vote without revealing it in the process. This can be used as an alternative to public endorsements. In settings where one person's expertise or judgement is well regarded our protocol offers the ability for others to trust his judgement without forcing him to reveal his opinion –this can be especially important in small, local elections where revealing one's vote can lead to rivalries (of course in a small or close election the tally [20] or even the result might reveal information). We also include an online variant of the protocol which offers greater security to the voter but requires a trusted server to accept ballots.

Blinded vote copying would also reduce the power of traditional voting blocs. A voting bloc is a club or special interest group that coordinates its voting. They achieve stronger [38] representation compared to individual voters by not diffusing their votes. The trust requirements for blinded vote copying are more relaxed than in a typical voting bloc since the "leader" does not need to make his vote public. By making the creation of voting blocs easier we thus create a more even voting field without needing to change the electoral system.

Since our main contribution is honest-verifier zero-knowledge a natural continuation would be implementing a trusted warden that facilitates the copying.

It would also be interesting to replace the original voter with a coalition of voters, essentially providing a framework (thus avoiding the complexity of secure multi-party computation) for holding a primary election amongst the members of the coalition. This can lower the barrier for creating a voting bloc further since there is no need for a single person to be singled out as the decision maker.

Another avenue for future research would be using a witness-hiding protocol such as Schnorr-Okamoto [36] instead of Schnorr as a proof of knowledge for discrete logs and integrating it with a homomorphic encryption scheme. The result would then be witness-hiding (which is adequate for security since the witness in our case includes the vote) rather than honest-verifier zero-knowledge.

In the context of internet voting, the issues of untrusted platforms and the lack of a private voting booth (generalised under "physical assumptions") have been known and well described (see eg. [35,13]). We argue that the potential for voter cooperation is a third characteristic, unique to internet voting. It is therefore natural to state a more general open problem: What other differences exist between e-booth and internet voting?

Acknowledgments. The authors would like to thank Dr. Niel Ferguson for suggesting the topic of encrypted copying during CRYPTO 2007. We also thank the referees for suggesting that code-voting prohibits blinded copying and for pointing out [7].

References

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

2. Adida, B., de Marneffe, O., Pereira, O., Quisquater, J.J.: Electing a University President using Open-Audit Voting: Analysis of real-world use of Helios. In: 2009 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 2009) (Online proceedings) (2009)
3. Arrow, K.J.: Social Choice and Individual Values, 2nd edn. Yale University Press, New Haven (1963)
4. Balzarotti, D., Banks, G., Cova, M., Felmetsger, V., Kemmerer, R., Robertson, W., Valeur, F., Vigna, G.: Are your votes really counted?: testing the security of real-world electronic voting systems. In: Proceedings of the 2008 International Symposium on Software Testing and Analysis, pp. 237–248. ACM (2008)
5. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM (1993)
6. Boneh, D.: The Decision Diffie-Hellman Problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
7. Bulens, P., Giry, D., Pereira, O.: Running mixnet-based elections with Helios. In: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections. Usenix (2011)
8. Burmester, M., Desmedt, Y., Beth, T.: Efficient zero-knowledge identification schemes for smart cards. The Computer Journal 35(1), 21 (1992)
9. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. Journal of the ACM (JACM) 51(4), 557–594 (2004)
10. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology: Proceedings of Crypto., vol. 82, pp. 199–203 (1983)
11. Chaum, D., Pedersen, T.: Wallet Databases with Observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
12. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–90 (1981)
13. Clark, J., Hengartner, U.: Selections: Internet Voting with Over-the-Shoulder Coercion-Resistance. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 47–61. Springer, Heidelberg (2012)
14. Computing Research Association: Four grand challenges in trustworthy computing (2003)
15. Cortier, V., Smyth, B.: Attacking and fixing Helios: An analysis of ballot secrecy. In: Proceedings of the 24th Computer Security Foundations Symposium, CSF 2011 (2011)
16. Cramer, R., Damgård, I., Schoenmakers, B.: Proof of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
17. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. European Transactions on Telecommunications 8(5), 481–490 (1997)
18. Crockford, D.: Javascript object notation (July 2006), `http://www.ietf.org/rfc/rfc4627.txt`
19. Desmedt, Y.G., Goutier, C., Bengio, S.: Special Uses and Abuses of the Fiat Shamir Passport Protocol. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 21–39. Springer, Heidelberg (1988)
20. Desmedt, Y., Kurosawa, K.: Electronic voting: Starting over? In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 329–343. Springer, Heidelberg (2005)

21. Desmedt, Y., Chaidos, P.: Blinding ballot copying in Helios: from Condorcet to IACR. In: CRYPTO 2011 Rump Session (2011), http://rump2011.cr.yp.to/
22. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
23. Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
24. Estehghari, S., Desmedt, Y.: Exploiting the client vulnerabilities in internet e-voting systems: hacking Helios 2.0 as an example. In: Proceedings of the 2010 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections. EVT/WOTE 2010, pp. 1–9. USENIX Association, Berkeley (2010)
25. Estonian National Electoral Committee: E-voting system -general overview (2010), http://www.vvk.ee/public/dok/General_Description_E-Voting_2010.pdf
26. Feige, U., Fiat, A., Shamir, A.: Zero-knowledge proofs of identity. Journal of Cryptology 1(2), 77–94 (1988)
27. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, pp. 416–426. ACM (1990)
28. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
29. Gerlach, J., Gasser, U.: Three case studies from switzerland: E-voting. Berkman Center Research Publication No. 3 (2009)
30. Gjøsteen, K.: Analysis of an internet voting protocol. Cryptology ePrint Archive, Report 2010/380 (2010)
31. Goldwasser, S., Kalai, Y.: On the (in) security of the Fiat-Shamir paradigm. In: Proceedings of 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 102–113. IEEE (2003)
32. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, pp. 291–304. ACM (1985)
33. Helios Voting: Helios v3 verification specs (August 2011), http://documentation.heliosvoting.org/verification-specs/helios-v3-verification-specs
34. Jacobs, B., Pieters, W.: Electronic voting in the Netherlands: from early adoption to early abolishment. In: Foundations of Security Analysis and Design V, pp. 121–144 (2009)
35. Mote Jr., C.: Report of the national workshop on internet voting: issues and research agenda. In: Proceedings of the 2000 Annual National Conference on Digital Government Research, pp. 1–59. Digital Government Society of North America (2000)
36. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
37. Okamoto, T., Ohta, K.: Divertible Zero Knowledge Interactive Proofs and Commutative Random Self-reducibility. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 134–149. Springer, Heidelberg (1990)
38. Penrose, L.: The elementary statistics of majority voting. Journal of the Royal Statistical Society 109(1), 53–57 (1946)
39. Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting Scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
40. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)