Auditing for privacy in threshold PKE e-voting

Auditing for Privacy in threshold PKE e-Voting

Abstract

The importance of voter auditing in order to ensure election integrity has been extensively studied in the e-voting literature. On the other hand, the necessity of auditing to protect voter privacy in an e-voting system has been mostly overlooked. This work investigates election privacy issues that appear in the state-of-the-art implementations of e-voting systems that apply *threshold public key encryption* (TPKE) in the client like Helios and use a bulletin board (BB). More specifically, it is shown that without PKI support or -more generally- authenticated BB "append" operations, such systems are vulnerable to attacks where the malicious election server can act as a man-in-the-middle between the election trustees and the voters, hence it can learn how the voters have voted. As countermeasure for this type of man-in-the-middle attacks, this work suggests compulsory trustee auditing which should be executed either (i) immediately after the election setup phase, if the BB is consistent, or (ii) right after the voting phase, if the BB is corrupted.

Finally, this work studies the impact of verifying the cryptographic (zero-knowledge) proofs posted in the BB from a privacy aspect; namely, it is shown that proof auditing implies significantly stronger *provable privacy guarantee* for a TPKE e-voting system against *covert adversaries* (i.e., adversaries that may deviate arbitrarily from the protocol specification, but do not wish to be detected), given that the proof is carried out via standard reduction to the security of the underlying TPKE scheme.

1 Introduction

E-voting systems have emerged as a powerful technology to improve the election process by reducing election cost, making election preparation and tally computation faster and increase voter participation for various underrepresented social groups including voters that face considerable physical barriers and overseas voters. In addition, several e-voting systems (Cramer et al. 1997, Chaum 2001, Adida 2008, Chaum et al. 2005, 2008, Kutylowski & Zagórski 2010, Zagórski et al. 2013, Benaloh et al. 2013, Kiayias et al. 2015*a*,*b*) are *end-to-end verifiable*, i.e., voters and auditors can directly verify the entire election process and be assured that no entities, even the election authorities, have manipulated the election result.

A major class of e-voting systems (Cohen & Fischer 1985, Cramer et al. 1997, *Estonia* n.d., Juels et al. 2005, Clarkson et al. 2008, Kiayias et al. 2006, Adida 2008, Tsoukalas et al. 2013, Gjøsteen 2013, Benaloh et al. 2013, Kiayias et al. 2015*a*) necessitate *client-side cryptography* (CSC); we call those CSC e-voting systems. In a CSC e-voting system, the voter makes use of a *voter supporting device* (VSD), which functionality is to generate an encrypted vote and submit it to the system on behalf of the voter. In addition, the VSD provides the voter with some additional information so that the voter can audit the execution of the election procedure either on her own or with the help of an *auditing supporting device* (ASD). Many CSC e-voting systems have been used in real-world binding procedures such as in the elections of scientific organizations (Adida 2008), academic institutions (Tsoukalas et al. 2013), or even local government (Gjøsteen 2013) and national elections (*Estonia* n.d.). Consequently, analyzing

and challenging the security of CSC e-voting systems has been a prominent theme in e-voting literature (Kremer et al. 2010, Gjøsteen 2010, Bernhard et al. 2011, 2012, Küsters et al. 2012, Springall et al. 2014, Smyth et al. 2015).

In their seminal work (Benaloh & Yung 1986), argue for distributing the administration of an election into multiple authorities in order to enhance its privacy. This paper sheds light on the design and implementation details that are necessary for the security of *multi-authority* CSC e-voting systems. In particular, this paper studies the election privacy of an important category of multi-authority CSC e-voting systems that include (Cramer et al. 1997, Kiayias et al. 2006, Adida 2008, Clarkson et al. 2008, Tsoukalas et al. 2013, Benaloh et al. 2013, Kiayias et al. 2015*a*) which construction utilizes a *threshold public key encryption (TPKE) scheme* -e.g. the threshold El Gamal cryptosystem (Pedersen 1991)- as follows: there is a set of *trustees* so that each trustee generates a pair of partial decryption and public keys and provide their keys into an *election authority (EA)*, along with *zero-knowledge (ZK) proofs* (Goldwasser et al. 1985) of correct key pair generation. The EA is responsible for posting the partial public keys and the corresponding ZK proofs on a publicly accessible *bulletin board* (BB). Then, the voters, using their VSD can encrypt their votes using the election public key that derives from the partial public keys and submit them to the EA. After the voting period ends, the EA processes the encrypted votes, either using the *additive homomorphic*[1] property of the TPKE scheme or via mixnets (Chaum 1981) to provide anonymity. Then, it sends the product of this process to every trustee which responds with their share of partial decryption of the tally and ZK proofs of correct decryption. Finally, the EA posts the information it receives from the trustees on the BB, so that the voters or any auditor can verify the election result.

The most widely used representative of TPKE-based e-voting systems is Helios (Adida 2008). This paper brings to surface a flaw in the current implementation of Helios that sounds the alarm for the preservation of voter privacy. Specifically, we observe that as yet, it is not required from the trustees to verify the integrity of the election public key posted in the BB before election starts. Due to this lack of precaution, there is nothing that prevents a malicious EA from launching a *man-in-the-middle (MitM)* attack by replacing the trustees' public keys with ones of its choice without becoming detected. Obviously, if this attack happens, then all the cast votes of the honest voters are directly exposed to the EA since they have been encrypted under the adversarial public key. In addition, the EA can engage with the trustees in a seemingly valid tally phase, by providing the trustees with a set of votes encrypted under their election public key.

In detail, this paper's contributions comprise:

1. Presenting the design and implementation defects that make any TPKE-based e-voting system susceptible to MitM attacks where an adversary that controls the EA can break voter privacy simply by posting public keys of its choice on the BB.

2. Proposing effective countermeasures to deal with this type of attacks against TPKE-based e-voting systems without PKI support (Kiayias et al. 2006, Adida 2008, Tsoukalas et al. 2013). Namely, each trustee should audit the e-voting process by verifying the correct record of its partial public key on the BB. It is argued that given a consistent BB, the proposed enhancement eliminates the possibility of such MitM attacks. Moreover, in case of an adversarially controlled BB, our countermeasure makes the system private against an election authority that acts as a *covert adversary* (Aumann & Lindell 2010), i.e. an

[1]The additive homomorphic property of an encryption scheme suggests that multiplying the encryptions of two messages m_1 and m_2 under some public key, results in an encryption of $m_1 + m_2$ under the same public key.

adversary that may deviate arbitrarily from the protocol specification, but does not wish to be detected cheating.

3. Analyzing the impact of verifying the ZK proofs posted in the BB from a privacy aspect, as further evidence on the importance of auditing to preserve election privacy. It is shown that ZK verification implies significantly stronger *provable privacy guarantee* for a TPKE e-voting system against covert adversaries, given that the proof is carried out via standard reduction to the security of the underlying TPKE scheme. Namely, for typical setup parameters (number of candidates, number of voters, TPKE scheme group size) lack of auditing results in major loosening of the privacy error upper bound, whereas enforcing auditing normally implies an acceptable level of privacy.

Comparison with preliminary version.

The preliminary version of this work (Kiayias et al. 2015*c*) was presented in the 6th e-Democracy International Conference. Compared to (Kiayias et al. 2015*c*), this work focuses exclusively on the study of auditing with respect to election privacy by including the analysis of the importance of ZK verification from the privacy aspect (cf. Section 5) in place of the generic guidelines for the secure implementation of TPKE e-voting systems discussed in (Kiayias et al. 2015*c*, Section 5). The new additional Section 5 of this work extends the core contribution of the conference version, that is, the importance of auditing for preserving election privacy.

Furthermore, this work captures a major revision of the editorial part of (Kiayias et al. 2015*c*). Among a list of edits, the MitM attack is now illustrated with the help of a sequence of detailed figures (cf. Figures 1, 2, 3, 4, and 5), whereas a separate subsection (cf. Subsection 4.3) is devoted to provide an explicit view of the effectiveness of the attack against several well-known TPKE-based e-voting systems.

Related work.

Security analysis of CSC e-voting systems:

(Kremer et al. 2010) proved the verifiability of Helios 2.0 in a symbolic framework. Similarly, (Smyth et al. 2015), introduce a computational framework for defining verifiability as a set of properties and perform an analysis of Helios and the JCJ (Juels et al. 2005) e-voting system in this framework. The ballot privacy of Helios is studied in (Bernhard et al. 2011) and (Bernhard et al. 2012). The security of the Estonian and the Norwegian e-voting system has been analyzed in (Springall et al. 2014) and (Gjøsteen 2010) respectively. (Jefferson et al. 2004), discuss the impact of man-in-the-middle attacks on e-voting security, expecially by spoofing the voting server.

Attacks on the security of Helios:

Helios is among the most widely used e-voting systems. As a result, several works studied and challenged the security of Helios. (Estehghari & Desmedt 2010) described an attack based on vulnerabilities of the voter's browser that compromised the integrity of Helios 2.0. A fix to prevent launching this attack was considered in Helios's upgrade to version 3.0. (Heiderich et al. 2011), presented an XSS attack against Helios and exposed serious security threats (these vulnerabilities were fixed by the developers of Helios). (Cortier & Smyth 2011) discovered an important attack on Helios 2.0 and 3.0, where the adversary can break the ballot secrecy of an honest voter by replaying his encrypted vote for a corrupted user that he controls. The attack can be generalized by exploiting the malleability of the Helios ciphertexts. They proposed a solution to this attack and proved the privacy that their solution achieves using applied pi calculus.

(Küsters et al. 2012) introduced a new type of attacks that they name clash attacks, which compromise the integrity of Helios, for variants where the ballots are not linked with the identities of the voters. Bernhard, Pereira and Warinschi (Bernhard et al. 2012) showed that Fiat-Shamir (FS) transformation (Fiat & Shamir 1986) applied in Helios must be properly implemented to prevent the system from integrity attacks.

2 Preliminaries

2.1 Notation

Throughout this paper, we use λ as the security parameter. A shorthand $x \xleftarrow{\$} \mathcal{X}$ denotes that x is drawn uniformly at random from a set $\mathcal{X}$. For algorithms and distributions, the notation $x \leftarrow \mathsf{Alg}(I)$ means that the element x is sampled according to the output distribution of Alg on input I. If Alg is a probabilistic algorithm, then we write $x \in \mathsf{Alg}(I)$ to denote that x is a possible output of Alg on input I. By $\langle x \rangle$, we denote the encoding of x as an element of some message space $\mathcal{M}$. The length of string x is denoted as $|x|$. By $\mathsf{negl}(\cdot)$, we denote that some function is *negligible*, i.e. it is asymptotically smaller than the inverse of any polynomial.

We say that two random variable ensembles $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are *(computationally) indistinguishable* if for every (PPT) algorithm $\mathcal{D}$ it holds that

$$\left| \Pr\left[s \leftarrow X_n : \mathcal{D}(s) = 1\right] - \Pr\left[s \leftarrow Y_n : \mathcal{D}(s) = 1\right] \right| = \mathsf{negl}(n) \ .$$

2.2 TPKE Schemes

Let $\mathsf{Ser}_1, \ldots \mathsf{Ser}_k$ be a set of k decryption servers. A (t,k)-*TPKE scheme* TPKE is a quintuple of algorithms ($\mathsf{TPKE.Gen}$, $\mathsf{TPKE.Combine}$, $\mathsf{TPKE.Enc}$, $\mathsf{TPKE.Dec}$, $\mathsf{TPKE.Recon}$) defined as follows:

- The *partial key generation* algorithm $\mathsf{TPKE.Gen}$ that on input 1^λ outputs the partial public key and secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ for Ser_i.
- The *public key construction* algorithm $\mathsf{TPKE.Combine}$ that on input $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$ computes the public key pk.
- The *encryption* algorithm $\mathsf{TPKE.Enc}$ that on input pk and a message M in some message space $\mathcal{M}$ outputs a ciphertext C.
- The *partial decryption* algorithm $\mathsf{TPKE.Dec}$ that on input sk_i and a ciphertext C either outputs a message share M_i or aborts.
- The *plaintext reconstruction algorithm* $\mathsf{TPKE.Recon}$ that on input a set of t out-of k message shares $M_{i_1}, \ldots, M_{i_t}$ outputs the message M or aborts.

The most common instantiation of a TPKE, utilized in (Cramer et al. 1997, Kiayias et al. 2006, Adida 2008, Clarkson et al. 2008, Tsoukalas et al. 2013, Benaloh et al. 2013, Kiayias et al. 2015*a*) is the (t,k)-*threshold El Gamal cryptosystem* (Pedersen 1991) that is *IND-CPA secure* under the *Decisional Diffie-Hellman* (DDH) assumption.

2.3 Zero-Knowledge Proofs of Knowledge

Let $\mathcal{L}$ be an **NP** language. A *zero-knowledge proof of knowlegde (ZK-PoK)* (Goldwasser et al. 1985) $\Gamma = (\mathsf{P}, \mathsf{V})$ for some language $\mathcal{L} \in \boldsymbol{\mathcal{NP}}$ with witness relation $R_\mathcal{L}$ is an interactive pair of an (potentially unbounded) *prover* P and a probabilistic polynomial time (PPT) *verifier* V that satisfies the following properties (informally stated):

1. *(Perfect) Completeness*: for every statement $x \in \mathcal{L}$ and $w \in R_{\mathcal{L}}(x)$, V always accepts the proof of $\mathsf{P}(w)$.

2. *Proof of knowledge*: for every (potentially malicious) prover P^* and $y \in \{0,1\}^*$ such that $\mathsf{P}^*(y)$ convinces V of the validity of some statement x with probability non-negligible in x, there exists a PPT knowledge extractor K that on input x and given access to the code of $\mathsf{P}^*(y)$, outputs a witness w for x with probability non-negligible in x.

3. *(Computational) Zero-Knowledge:* for every (PPT) verifier V^*, there exists a PPT simulator algorithm Sim such that for every statement $x \in \mathcal{L}$ and $w \in R_{\mathcal{L}}(x)$, Sim outputs transcripts that are (computationally) indistinguishable from transcripts generated by the interaction between $\mathsf{P}(w)$ and V^*.

3 A typical TPKE-based e-voting system

Let $\boldsymbol{\mathcal{VS}}$ be a TPKE e-voting system. We consider four parameters: a security parameter λ that determines the security level of the underlying cryptographic primitives, the number of voters n, the number of candidates m and the number of trustees k, which comprise the election committee that guarantees the privacy of the election. We use the notation $\mathcal{V} = \{V_1, ..., V_n\}$ for the set of voters, $\mathcal{P} = \{P_1, ..., P_m\}$ for the set of candidates and $\mathcal{T} = \{T_1, \ldots, T_k\}$ for the set of trustees. For simplicity, we consider the case of *1-out-of-m elections*, where the set of allowed selections is the collection of singletons, $\{\{P_1\}, \ldots, \{P_m\}\}$, from the set of candidates $\mathcal{P}$.

3.1 Entities in a TPKE e-voting system

In an abstract form, the entities involved in $\boldsymbol{\mathcal{VS}}$ are:

- The *Election Authority (EA)* that sets up the election and communicates initialization data to all other components. Additionally, the EA is responsible for posting the election results and possibly some election audit information on the BB.
- The *Bulletin Board (BB)*, where the election result and all necessary audit information is posted. The BB allows only an "append" operation and recognizes at minimum one entity (the EA).
- The *Voters* who are equipped with a *voter supporting device (VSD)* (that can be a tablet, PC or other network enabled equipment) as well as an *auditing supporting device (ASD)* that can be the same as the VSD or a different network enabled device.
- The *Trustees* that constitute a subsystem responsible for the election tally. Each trustee is equipped with its own pair of *trustee supporting device (TSD)* and ASD.

3.2 Description

Let $\mathsf{TPKE} = (\mathsf{TPKE.Gen}, \mathsf{TPKE.Combine}, \mathsf{TPKE.Enc}, \mathsf{TPKE.Dec}, \mathsf{TPKE.Recon})$ be an IND-CPA TPKE scheme as described in Section 2.2. For every security parameter λ, partial key pair $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{TPKE.Gen}(1^\lambda)$, $i = 1, \ldots, k$ and public key $\mathsf{pk} \leftarrow \mathsf{TPKE.Combine}(\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$ we define the following **NP** languages:

$$\mathcal{L}_\lambda = \big\{\mathsf{pk}_i \mid \text{there is an } \mathsf{sk}_i^* : (\mathsf{pk}_i, \mathsf{sk}_i^*) \in \mathsf{TPKE.Gen}(1^\lambda)\big\},$$
$$\mathcal{L}_{\lambda,\mathsf{pk}} = \big\{C \mid \text{there is a } P \in \mathcal{P} : C \in \mathsf{TPKE.Enc}(\mathsf{pk}, \langle P \rangle)\big\} \quad \text{and}$$
$$\mathcal{L}_{\lambda,\mathsf{pk},sk_i} = \big\{(C, M_i) \mid M_i \leftarrow \mathsf{TPKE.Dec}(\mathsf{sk}_i, C)\big\}.$$

Informally, $\mathcal{L}_\lambda$ is the language of valid partial public keys, $\mathcal{L}_{\lambda,\mathsf{pk}}$ is the language of the well-formed encrypted votes under pk and $\mathcal{L}_{\lambda,\mathsf{pk},sk_i}$ is the language of valid pairs of encryptions under pk and partial decryptions for trustee T_i.

Following the syntax introduced in (Kiayias et al. 2015*b*), $\boldsymbol{\mathcal{VS}}$ is a quintuple of algorithms and protocols $\langle$**Setup**, **Cast**, **Tally**, **Result**, **Verify**$\rangle$ as follows:

The protocol **Setup**$(1^\lambda, \mathcal{P}, \mathcal{V}, \mathcal{T})$:

Each trustee T_i uses its TSD to run $\mathsf{TPKE.Gen}(1^\lambda)$ and receive the partial key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$. It sends pk_i to the EA along with a ZK-PoK of $\mathsf{pk}_i \in \mathcal{L}_\lambda$ by *proving knowledge of* sk_i. If there is a proof that EA does not verify, then EA aborts the protocol. Upon receiving all $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$, EA computes the election public key $\mathsf{pk} \leftarrow \mathsf{TPKE.Combine}(\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$. Then, it posts the public parameters, Pub, which include the election information Info, pk, the partial public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$ as well as the ZK-PoK of $\mathsf{sk}_1, \ldots, \mathsf{sk}_k$ in the BB. Namely, the election transcript τ of the BB is initialized as Pub. Finally, EA generates the voter credentials $\mathsf{cr}_1, \ldots, \mathsf{cr}_n$, where cr_ℓ contains a voter identifier ID_ℓ, and issues them to the voters $V_1, \ldots, V_n$ respectively.

The protocol **Cast**:

At the voting phase, each voter V_ℓ chooses a candidate selection $\{P_{j_\ell}\}$ and sends $(\mathrm{ID}_\ell, \{P_{j_\ell}\})$ to her VSD. The VSD reads pk from the BB and creates a ciphertext $C_\ell \leftarrow \mathsf{TPKE.Enc}(pk, \langle P_{j_\ell}\rangle)$, where $\langle P_{j_\ell}\rangle$ is the encoding of candidate P_{j_ℓ}. In addition, it attaches a ZK-PoK of *ballot correctness* π_ℓ showing that $C_\ell \in \mathcal{L}_{\lambda,\mathsf{pk}}$. The *encrypted ballot* generated is $\mathbf{B}_\ell = (C_\ell, \pi_\ell)$. Upon the generation of $\mathbf{B}_\ell$, the VSD provides V_ℓ with a *ballot tracker* (e.g. a hash of $\mathbf{B}_\ell$), Tr_ℓ, so that V_ℓ can locate $\mathbf{B}_\ell$ in the BB after election ends. Next, V_ℓ must choose either one of the following options by flipping a coin with some bias $p \in [0, 1]$[2].

A. ***Cast*** her vote. In this case, V_ℓ provides VSD with her credential cr_ℓ. Subsequently, VSD submits $\mathbf{B}_\ell$ to the EA using cr_ℓ.

B. ***Audit*** the ballot $\mathbf{B}_\ell$ via its ASD, by requiring the VSD to provide all the randomness used to generate $\mathbf{B}_\ell$. In this case, the ballot $\mathbf{B}_\ell$ is spoiled and when the audit is finished, V_ℓ requests a new ballot $\mathbf{B}'_\ell$ (possibly under a different candidate selection $\{P'_{j_\ell}\}$). After V_ℓ has received the new ballot tracker Tr'_ℓ, she will be prompted to audit-or-cast $\mathbf{B}'_\ell$.

When EA receives a submitted ballot vote $\mathbf{B}_\ell = (C_\ell, \pi_\ell)$, it checks that it is well-formed by verifying the ZK-PoK π_ℓ. If the check fails, then EA aborts the protocol. After voting ends, EA updates its state with the pairs $\{(\mathrm{ID}_\ell, \mathbf{B}_\ell)\}_{V_\ell \in \mathcal{V}_{\mathsf{succ}}}$ of cast votes and the associated identifiers, where $\mathcal{V}_{\mathsf{succ}}$ is the set of voters that voted successfully.

The protocol **Tally**:

The EA processes $\{(\mathrm{ID}_\ell, \mathbf{B}_\ell)\}_{V_\ell \in \mathcal{V}_{\mathsf{succ}}}$ [3] to provide anonymity, either using the additive homomorphic property of the threshold ElGamal scheme or via mixnets (Chaum 1981). Then, the EA sends the processed votes to all trustees. Every trustee T_i, $i = 1, \ldots, k$, uses their TSD to perform the following computation: it uses sk_i and all ciphertexts $\{C_\ell\}_{V_\ell \in \mathcal{V}_{\mathsf{succ}}}$ to compute T_i's partial decryption of the tally denoted by R_i. Then, it sends R_i to the EA along with a ZK-PoK *correct partial decryption*. If there is a proof that EA does not verify, then it aborts

[2] If such a feature is not available in some specific instantiation, as in. (Cramer et al. 1997, Kiayias et al. 2006), then we trivially set $p = 0$.

[3] For instance (Cramer et al. 1997, Kiayias et al. 2006, v4 n.d., Benaloh et al. 2013) apply the homomorphic operation on the encrypted votes while (Clarkson et al. 2008, Tsoukalas et al. 2013) use mixnets in order to provide anonymity.

the protocol. After all trustees finish their computation, EA updates the BB's transcript τ with $\{(\mathrm{ID}_\ell, \mathbf{B}_\ell)\}_{V_\ell \in \mathcal{V}_{\mathsf{succ}}}$ along with all the partial decryptions and the respective ZK-PoK sent by the trustees.

The algorithm **Result**:

The election result R is the output of $\mathsf{TPKE.Recon}(R_{i_1}, \ldots, R_{i_t})$, where $R_{i_1}, \ldots, R_{i_t}$ is any subset of t out-of-the k partial decryptions $R_1, \ldots, R_k$.

The algorithm **Verify**(τ, Tr_ℓ):

The verification algorithm is accepting if and only if the following conditions hold:

1. The structure of the information posted on the BB and all election information is correct (using Info).
2. There exists a ballot in the BB that matches the ballot tracker Tr_ℓ.
3. The ZK-PoK proofs for the correctness of all ballots on the BB verify.
4. The ZK-PoK proofs for the correctness of all trustees' partial decryptions verify.

4 A MitM attack against voter privacy

Note that according to our description of a general TPKE-based e-voting system in Section 3, there is no way for the BB to authenticate the trustees' data (this typically requires a user-side PKI which is hard to deploy in practice) and the trustees are not required to audit the information posted in the BB e.g., as in Helios (Adida 2008).

Due this oversight, subtle privacy problems emerge in the case of a malicious EA. In order to achieve voter privacy, it is necessary to ensure that at least one of the trustees participates in the election audit. For instance, this is consistent with claims made in the Helios web server material where it is argued that voter privacy is guaranteed unless all the trustees are corrupted, see (Helios n.d.). Nevertheless, the trustee auditing step is optional and there are no proper instructions regarding the necessity of the trustee verification process. Moreover, the current Helios implementation (v4 n.d.) makes it very difficult for someone without technical knowledge and understanding of the code to do so.

In Section 4.1, we introduce a generic MitM attack against the voter privacy of a TPKE-based e-voting system when the trustee auditing step is not performed. Next, in Section 4.2, we demonstrate our attack against Helios as a specific instance of the attack methodology. We summarize the effectiveness of our attack in Section 4.3. In Section 4.4, we propose simple countermeasures that deal with this type of attacks.

4.1 The system vulnerability and our MitM attack

Recall that a typical TPKE-based e-voting system mainly consists of EA, BB, and VSD, and it forms a star network as depicted in Figure 1. Namely, the BB, all the trustees and all the VSDs are connected through the central node, the EA. Such a network topology is sensible and is followed by systems used in practice (including Helios) since it avoids additional pair-wise communication between the entities participating in the election; this has a number of advantages. First of all, it significantly reduces the development and deployment complexity, as the entire e-voting system can be realized by a single server. Secondly, it is consistent with a reasonable level of usability, as the administrators only need to keep the election server online

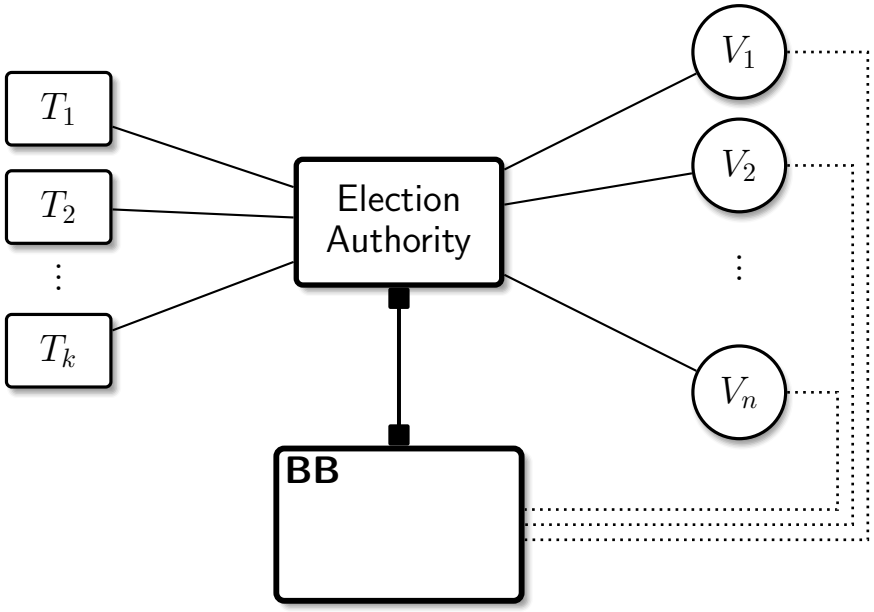

Figure 1: The star network topology in the architecture of a typical TPKE e-voting system. The dotted lines denote read-only access to the BB.

and all the other election parties are able to participate in the election asynchronously without any coordination.

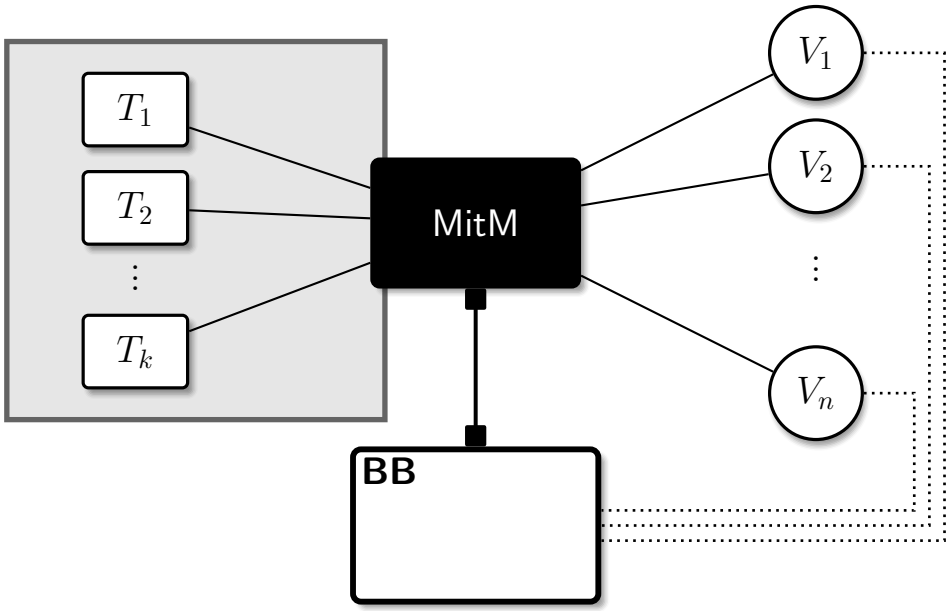

Figure 2: A malicious EA acting as MitM against the privacy of a TPKE e-voting system. The trustees $T_1, T_2, \ldots, T_k$ communicate only with the malicious EA, isolated in a fake "gray" environment.

Unfortunately, the implementation efficiency enjoyed by the aforementioned star network topology does not come without a price; the specific architecture makes the system vulnerable

to a class of MitM attacks when the central node (i.e. the election server) is compromised, as depicted in Figure 2. Furthermore, the lack of PKI support makes it impossible for a third-party auditor to identify the actual sources of messages that appear in the BB. This problem is recognized in terms of election integrity, and the concept of *individual verifiability* is widely adopted to mitigate this problem by preventing the malicious election server from tampering the submitted ballots. Nevertheless, little attention is given to the contributions of the election trustees even though it is equally important to ensure the integrity of trustees' messages (i.e. the election parameters) in the BB.

Description of the MitM attack:

Our attack assumes that only the EA is controlled by the adversary, whereas the rest of the TPKE-based e-voting system entities and all the supporting devices remain honest. The attack consists of three steps 1,2 and 3 that are illustrated via the corresponding Figures 3, 4, and 5.

STEP 1: During the election setup phase, the malicious EA follows the **Setup** protocol description and interacts with the real trustees $T_1, \ldots, T_k$ to jointly generate the real election public parameters Pub and the real voters' credentials $\mathsf{cr}_1, \ldots, \mathsf{cr}_n$. Meanwhile, the malicious EA (conceptually) creates another set of fake trustees, $T_1^*, \ldots, T_k^*$ and generates fake pairs of keys $(\mathsf{sk}_1^*, \mathsf{pk}_1^*), \ldots, (\mathsf{sk}_k^*, \mathsf{pk}_k^*)$ fake election public parameters Pub^* and the fake voters' credentials $\mathsf{cr}_1^*, \ldots, \mathsf{cr}_n^*$ by running the **Setup** protocol with the fake trustees "in its mind", obtaining $\mathsf{sk}_1^*, \ldots, \mathsf{sk}_k^*$. The malicious EA then publishes Pub* (that include $\mathsf{pk}_1^*, \ldots, \mathsf{pk}_k^*$ and $\mathsf{pk}^* \leftarrow \mathsf{TPKE.Combine}(\mathsf{pk}_1, \ldots, \mathsf{pk}_k^*)$) on the BB and thus all the voters will use the fake election parameters during the **Cast** protocol.

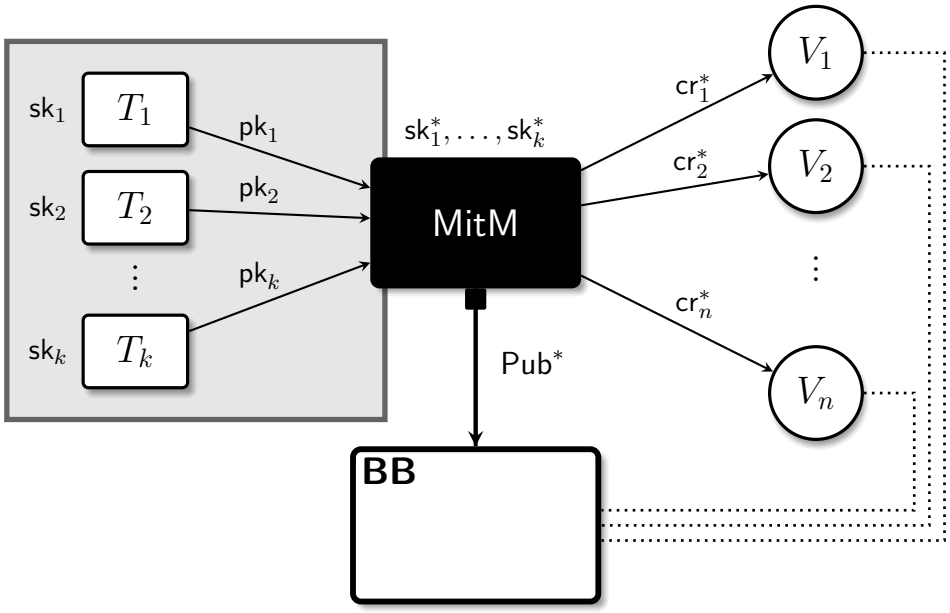

Figure 3: Replacement of the trustees' election parameters and setting up of a "fake" election (STEP 1).

STEP 2: The voters $V_1 \ldots, V_n$ read the BB and obtain the fake public key pk^*. They encrypt their candidate selections $P_{j_1}, \ldots, P_{j_n}$ under pk^* and submit their ballots $\mathbf{B}_1^*, \ldots, \mathbf{B}_n^*$

to the malicious EA which, clearly, is able to learn every voter's selection by decrypting the ballots using all fake partial decryption keys $\mathsf{sk}_1^*, \ldots, \mathsf{sk}_k^*$.

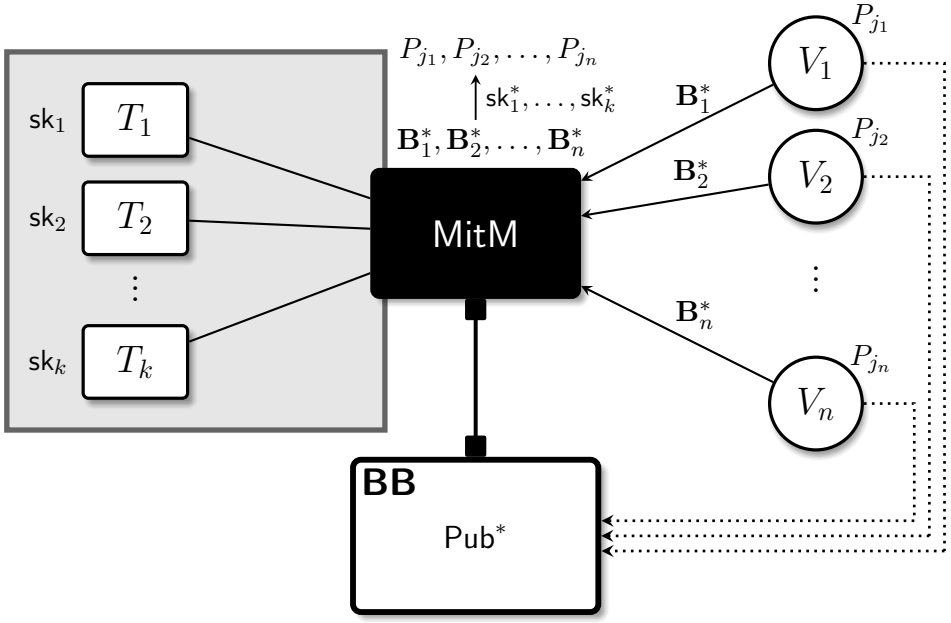

Figure 4: Voting under fake election public key and breaching the voters' privacy (STEP 2).

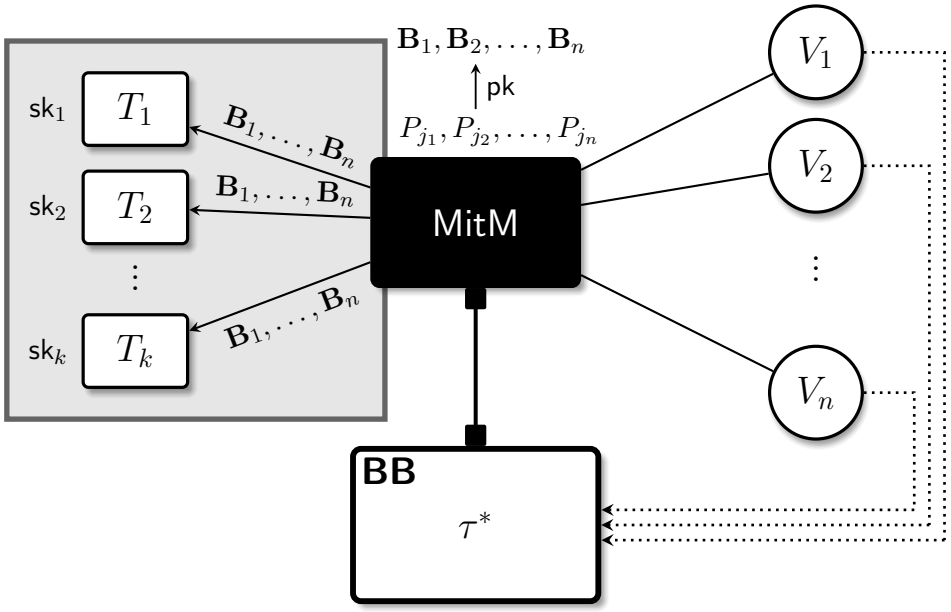

Figure 5: Completion of a consistent "real" tally phase under the trustees' election public key (STEP 3).

STEP 3: The malicious EA can simulate the voting and tally phase of a presumably "real" election run under the real trustees' keys engaging with them in the **Tally** protocol. The purpose of this step is to make the real trustees believe the election tally result is produced by them, whereas EA can simply perform the actual election tally itself and publish the corresponding election result in the BB. It is easy to see that all information in the BB is consistent in the sense that the produced fake election transcript τ^* is publicly verifiable.

4.2 Instantiation of our MitM attack against Helios

For concreteness, we demonstrate our MitM attack against Helios. There are many variants of Helios in the literature (supporting use of aliases, tally via homomorphic addition or mix-net tally, etc.). However, our attack can apply to all the variants with respect to their latest implementations. Our attack does not tamper the JavaScript code, therefore it is impossible to detect our attack by checking the integrity of the source code as observed at the client side.

Recall that the latest version of Helios v4 uses k out-of k threshold (lifted) ElGamal encryption. During the election setup phase, each trustee T_i, $i = 1, \ldots, k$ locally generates a pair of ElGamal partial keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ and sends pk_i to the EA. At this step, a fingerprint (SHA256 hash digest) of the partial public key pk_i is computed and provided to the trustee. It is suggested that the trustees keep the fingerprints of their partial public key and confirm that they are properly stored on the election server. However, there is no further instruction to indicate to the trustees where and how to verify the consistency of this information. Notice that there is no interface for the trustees to verify whether their partial public keys are correctly used to produce the (combined) election public key during the **Setup** protocol. Besides, each trustee does not know the other trustees' actual partial public keys. In fact, only the election public key is used to generate the election fingerprint (i.e. a SHA256 hash digest of the JSON format of the election definition) after an election is fixed (or "frozen" in Helios terminology). Moreover, the voters are only given the election public key at the *voting booth* page in the **Cast** protocol, so it is impossible to check the validity of the partial public keys even if the trustee is also an eligible voter. During the tally phase, the trustees are given their partial public key fingerprints to prevent them from using incorrect partial secret keys. The information displayed on the tally page can never be used for auditing purposes, because every trustee should first identify himself by submitting a unique token to the EA before receiving the content. Hence, the malicious EA can specifically tailor a (inconsistent) view of the tally page for each trustee. Finally, the partial public key information is not even displayed on the bulletin board, which only contains the submitted voters' ciphertexts.

In our MitM attack, the malicious EA receives pk_i from the trustee T_i, $i = 1, \ldots, k$ during the election setup phase. Then, it generates another set of fake partial ElGamal key pairs $(\mathsf{pk}_i^*, \mathsf{sk}_i^*)$ and computes the fake election public key $\mathsf{pk}^* = \prod_{i=1}^{k} \mathsf{pk}_i^*$. When the election is frozen, the malicious EA switches the real election public key with pk^*, so pk^* is used to generate the election fingerprint. In the voting booth, the voters are given pk^* to encrypt their choices, and thus it is consistent with the election fingerprint. In the tally phase, the malicious EA sends the real trustee T_i his partial public key pk_i; therefore, the hash of pk_i matches the fingerprint stored by T_i and the trustee should perform tallying as usual. Once T_i submits the decryption values, the malicious EA mimics the same process with T_i^* (in its "head") and posts the fake decryption factors instead. Clearly, all the information on the BB is publicly verifiable.

Remark. If a trustee carefully checks the public verification page, it is possible to find out that its partial public key is missing. However, the user is never instructed to perform such a step; besides this, the highlighted "Verified" indication hints to the user that everything is okay. We

stress that the voters' privacy is violated already even in the case where the trustees check the public verification page *after* voting phase ends (as the voters will have already encrypted the votes under the adversarial public key).

4.3 Effectiveness of our MitM attack

Our MitM attack can be launched against any TPKE-based e-voting system which, like Helios, (i) does not urge that the trustees directly verify that their partial public keys were correctly published and (ii) does not allow for any third party to verify the source of the partial public keys, as in (Kiayias et al. 2006, Tsoukalas et al. 2013). Note that having the EA sign BB data like in (Benaloh et al. 2013) does not prevent the attack; the adversarial public key is posted on the BB by the EA itself, hence BB consistency is not violated from the EA's point of view. On the contrary, our MitM attack can be prevented when the trustees collaboratively verify their partial public keys either via a verifiable secret sharing scheme (Cramer et al. 1997) or with PKI support (Clarkson et al. 2008). Finally, we note that the architecure in (Kiayias et al. 2015*a*) deviates from the typical star topology in Figure 1, thus, although (Kiayias et al. 2015*a*) can be classified as a TPKE-based e-voting system, it remains out of the scope of this work.

The effectiveness of the attack against various TPKE-based e-voting systems is summarised in Table 1.

System	Resistance	Vulnerability
(Cramer et al. 1997)	✓	
(Kiayias et al. 2006)		✓
(Clarkson et al. 2008)	✓	
(Adida 2008)		✓
(Tsoukalas et al. 2013)		✓
(Benaloh et al. 2013)		✓

Table 1: Effectiveness of the MitM attack against various TPKE-based e-voting systems.

4.4 Countermeasures

In this section, we propose two countermeasures for the MitM attack described in Section 4.1 and for TPKE-based e-voting systems without PKI support (Kiayias et al. 2006, Adida 2008, Tsoukalas et al. 2013). Each countermeasure suits a specific threat model for the BB.

Given a consistent BB:

As commonly used in the e-voting literature (Cramer et al. 1997, Kiayias et al. 2006, Adida 2008, Clarkson et al. 2008, Tsoukalas et al. 2013, Benaloh et al. 2013) and shown in Figure 1, the BB is considered to be passive and robust in the sense that posting on the BB is done in an append-only way. In this model, the trustee auditing step should be performed *immediately after* the election setup phase and *before* the voting phase starts. Since the adversary cannot modify the election public key in the BB, it cannot decrypt the encrypted votes without knowing all the partial secret keys. Hence, the voters' privacy is preserved, if at least one of the trustees remains honest.

Under BB corruption:

In an alternative threat model, we consider a *covert* adversary (Aumann & Lindell 2010) (i.e. an adversary that may deviate arbitrarily from the protocol specification, but does not

wish to be detected cheating) that may also fully corrupt the BB but cannot link the identity of the auditing party (including both voters and trustees) with the ASD that is used. In this model,

(i.) the trustee auditing step should be performed *after* the end of the voting phase and

(ii.) the interaction between the BB and a trustee's ASD should be *indistinguishable* from an interaction between the BB and any voter's ASD.

Observe that the election public key that has been used for vote encryption is determined w.r.t. a specific voter's ballot tracker. This is because the said public key can be deducted from the statement of any ZK-PoK of ballot correctness. In order to pass the trustee auditing, the adversary has to post the real public key on the BB whereas in order to pass the voter auditing, it has to post the fake public key. Since the adversary cannot tell whether an auditing party is a trustee or a voter, it will either (i) be discouraged from launching the MitM attack or (ii) eventually be detected.

5 Auditing for Optimal Provable Security

In this section, we study an additional case where election auditing is a crucial parameter for the preservation of voter privacy. In particular, we point out how the lack of ZK proof verification affects the level of privacy that can be provably guaranteed in a typical TPKE e-voting system against a covert adversary. In our argumentation, we avoid technical details that could harm the readability of the paper, without making discounts on the accuracy of our findings.

Formal modeling of election secrecy (privacy, receipt-freeness, coercion resistance) has been a prominent goal in the literature, equipping e-voting research with a variety of symbolic and cryptographic definitions (cf. (Kiayias et al. 2015*c*) for a relevant list of related work). In all these definitions, indistinguishability among votes that sum to the same result is the minimum requirement. Without loss of generality, this privacy property can be narrowed down to the following case:

> *If all but two voters are corrupted, then the adversary should not be able to distinguish whether the two honest voters voted for* ('YES','NO') *or* ('NO','YES').

In a typical TPKE e-voting system $\boldsymbol{\mathcal{VS}}$ as described in Section 3, the said privacy property, that we denote by $\mathbf{P}$ for brevity, can be proven via a *standard cryptographic reduction* to the security of the underlying TPKE scheme, normally a threshold ElGamal cryptosystem. Specifically, we can show that if there is an adversary $\mathcal{A}$ that breaks $\mathbf{P}$, then we can construct an adversary $\mathcal{B}$ that breaks the IND-CPA security of the TPKE scheme.

Intuitively, IND-CPA security suggests that if the adversary obtains the TPKE partial public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$ and performs encryptions of its own selection, then it can not distinguish an encryption of 'YES' from an encryption 'NO', if the encrypted message is chosen at random. For syntax consistency, the options 'YES' and 'NO' are encoded as the messages 1 and 0 respectively. The quantity

$$\left| \Pr[\mathcal{B}(\mathsf{pk}_1, \ldots, \mathsf{pk}_k) \text{ outputs } 1] - \Pr[\mathcal{B}(\mathsf{pk}_1, \ldots, \mathsf{pk}_k) \text{ outputs } 0] \right|$$

is called the *distinguishinh advantage* of the IND-CPA adversary and if the TPKE scheme is secure, then it is a negligible value. The adversary must allow at least one decryption server to be honest, otherwise it could decrypt all ciphertexts and security would be trivially broken.

At a high level, the IND-CPA adversary $\mathcal{B}$ receives $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$ and executes the following steps:

1. It receives a *challenge ciphertext* C^* that is an encryption of either 1 ('YES') or 0 ('NO').
2. It invokes $\mathcal{A}$ and simulates an election execution of $\boldsymbol{\mathcal{VS}}$ where $\mathcal{A}$ and $\mathcal{B}$ play the roles of the corrupted and honest parties respectively. Every corrupted trustee by $\mathcal{A}$ corresponds to a decryption server that $\mathcal{B}$ can corrupt.
3. During the simulated execution, $\mathcal{A}$ is required to output a verdict about $\mathbf{P}$, i.e. whether two specific voters voted for ('YES','NO') or ('NO','YES').
4. If the simulation finishes successfully, then $\mathcal{B}$ outputs 1 if $\mathcal{A}$ returns ('YES','NO'), or 0 if $\mathcal{A}$ returns ('NO','YES'). If the simulation is terminated incomplete or $\mathcal{A}$ returns no output, then $\mathcal{B}$ outputs a random response.

By the above construction, the following claim can be shown:

> *If $\mathcal{A}$ breaks $\mathbf{P}$ with probability α and the election simulation is completed successfully, then the distinguishing advantage of $\mathcal{B}$ is at least $\alpha/2$.*

As a result, it remains to compute the probability of the event that $\mathcal{B}$ runs a successful simulation. For this event to happen, the following two conditions are crucial.

(A). The TPKE scheme must satisfy some algebraic property (homomorphic addition), enabling $\mathcal{B}$ to manipulate the tally accordingly.

(B). $\mathcal{B}$ must be able to use the partial tally derived by the adversarial votes controlled by $\mathcal{A}$.

Without proceeding to details, we note that this property is satisfied by the threshold ElGamal cryptosystem, thus we may assume condition (A) holds. In order for condition (B) to hold, $\mathcal{B}$ must either (i) *extract* the adversarial tally if the underlying cryptography supports it, or (ii) *guess* the adversarial tally, out-of all possible outcomes.

In most standard implementations such as Helios, extraction is not supported[4]. Therefore, provable security relies on the guessing probability. In this case, and based on the previous claim, election privacy can be informally stated as follows:

> *If $\mathcal{A}$ breaks $\mathbf{P}$ with probability α and the probability that $\mathcal{B}$ guesses the tally is p, then the distinguishing advantage of $\mathcal{B}$ is at least $p\alpha/2$.*

[4]The ZK proofs are non-interactive and/or the Fiat-Shamir transformations are of the weak type.

The importance of auditing the ZK proofs for provable election privacy. By the above privacy statement, the larger the "reduction factor" p is, the stronger privacy we can guarantee for $\boldsymbol{\mathcal{VS}}$. Since $\mathcal{B}$ can not do more than a random guess, if the number of all possible outcomes is X, then this probability p is $1/X$. We now make a crucial observation; if no party verifies the ZK proofs, then the strategy of $\mathcal{A}$ may include a meaningless adversarial tally that lies arbitrarily in the underlying group of the TPKE scheme. If the number of candidates is m and the size the of the group is q, then there are q^m possible outcomes. The latter holds even if $\mathcal{A}$ is a covert adversary, since there is no entity that will detect its fraud.

On the other hand, if ZK proofs are verified by some auditor (voter, trustee, or an external party) and $\mathcal{A}$ is covert, then it is forced to submit a meaningful set of votes that results in a valid tally, otherwise it will be detected with high probability by the soundness of the ZK proofs. A valid tally can be expressed as one of the non-negative integer solutions of the equation $x_1 + \ldots + x_m = n$, where n is the number of voters. Thus, the possible valid outcomes now upper bounded by $\binom{n+m-1}{n} \leq n^{m-1}$.

Consequently, lack of auditing leads to reduction factor of $1/q^m$, where as ZK verification implies a factor $\geq 1/n^{m-1}$. To get an estimation of this gap, note that even the value 2^{160} is considered relatively small for q (elliptic curve groups), whereas n is maximized at $\sim 2^{30}$ (billions of voters). If, for instance, we set $m = 2$, then even assuming 384-bit IND-CPA security for the TPKE scheme, it must hold that

$$p\alpha/2 \leq 2^{-384} \Leftrightarrow \alpha \leq 2^{-383}/p \; .$$

If $p = 1/q^m \leq 2^{-160 \cdot 2}$, then $\alpha \leq 2^{-63}$, which could be barely acceptable election privacy guarantee. However, if $p \geq 1/n^{m-1} \geq 2^{-30}$, then $\alpha \leq 2^{-353}$ which implies a strong 353-bit security level.

6 Conclusion

We provided a formal description of a typical TPKE e-voting system with a star network topology, where the EA plays the role of the central node for all other involved entities. We showed that this architecture is susceptible to MitM attacks against privacy by having a malicious EA replacing the trustees' public keys, when BB data are non-authenticated. We instantiated the attack against Helios and discussed the possibility that these attacks jeopardize the security of other well-known TPKE-based e-voting systems. We suggested that imposing trustee auditing as part of the election guidelines could remove vulnerability under such attacks when (i) the BB is consistent or (ii) the BB is inconsistent, but the adversary will refrain from attacking, if it is aware that it will be detected. Finally, we presented an additional case where auditing has a beneficial effect on election privacy, by pointing out the significant gap in the level of provable privacy in TPKE e-voting systems against covert adversaries, from the state of (a) no party to (b) at least one party performing verification of the ZK proofs posted in the BB. We suggest that BB auditing by at least one entity should be enforced, so that election privacy can be supported persuasively, given current proof techniques.

We believe that this paper promotes subsequent research for the study of auditing with respect to election privacy. In addition, it gives emphasis to the necessity for meticulous guidelines regarding the imperatives of any entity involved in an e-voting execution (see (Kiayias et al. 2015*c*) for a list of such guidelines). As future work, auditing for privacy could be treated formally as part of a security study that focuses on human behavior in e-voting.

Acknowledgements. This research was partly supported by ERC project #259152 (CODAMODA), Horizon 2020 project #653497 (PANORAMIX), and project FINER, Greek Secretariat of Research and Technology, funded under action ARISTEIA 1.

References

2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013 (2013), USENIX Association.

Adida, B. (2008), Helios: Web-based open-audit voting, *in* P. C. van Oorschot, ed., 'Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA', USENIX Association, pp. 335–348.

Aumann, Y. & Lindell, Y. (2010), 'Security against covert adversaries: Efficient protocols for realistic adversaries', *J. Cryptology* **23**(2), 281–343.

Benaloh, J., Byrne, M. D., Eakin, B., Kortum, P. T., McBurnett, N., Pereira, O., Stark, P. B., Wallach, D. S., Fisher, G., Montoya, J., Parker, M. & Winn, M. (2013), Star-vote: A secure, transparent, auditable, and reliable voting system, in *2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013 2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013* (2013).

Benaloh, J. C. & Yung, M. (1986), Distributing the power of a government to enhance the privacy of voters (extended abstract), *in* J. Y. Halpern, ed., 'Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing, Calgary, Alberta, Canada, August 11-13, 1986', ACM, pp. 52–62.

Bernhard, D., Cortier, V., Pereira, O., Smyth, B. & Warinschi, B. (2011), Adapting Helios for provable ballot privacy, *in* V. Atluri & C. Díaz, eds, 'Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings', Vol. 6879 of *Lecture Notes in Computer Science*, Springer, pp. 335–354.

Bernhard, D., Pereira, O. & Warinschi, B. (2012), How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios, *in* X. Wang & K. Sako, eds, 'Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings', Vol. 7658 of *Lecture Notes in Computer Science*, Springer, pp. 626–643.

Chaum, D. (1981), 'Untraceable electronic mail, return addresses, and digital pseudonyms', *Commun. ACM* **24**(2), 84–88.

Chaum, D. (2001), Surevote: Technical overview, *in* 'Proceedings of the Workshop on Trustworthy Elections', WOTE.

Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A. T. & Vora, P. L. (2008), 'Scantegrity: End-to-end voter-verifiable optical-scan voting', *IEEE Security & Privacy* **6**(3), 40–46.

Chaum, D., Ryan, P. Y. A. & Schneider, S. A. (2005), A practical voter-verifiable election scheme, *in* S. D. C. di Vimercati, P. F. Syverson & D. Gollmann, eds, 'Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings', Vol. 3679 of *Lecture Notes in Computer Science*, Springer, pp. 118–139.

Clarkson, M. R., Chong, S. & Myers, A. C. (2008), Civitas: Toward a secure voting system, *in* '2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA', IEEE Computer Society, pp. 354–368.

Cohen, J. D. & Fischer, M. J. (1985), A robust and verifiable cryptographically secure election scheme (extended abstract), *in* '26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985', IEEE Computer Society, pp. 372–382.

Cortier, V. & Smyth, B. (2011), Attacking and fixing Helios: An analysis of ballot secrecy, *in* 'Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011', IEEE Computer Society, pp. 297–311.

Cramer, R., Gennaro, R. & Schoenmakers, B. (1997), A secure and optimally efficient multi-authority election scheme, *in* W. Fumy, ed., 'Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding', Vol. 1233 of *Lecture Notes in Computer Science*, Springer, pp. 103–118.

Estehghari, S. & Desmedt, Y. (2010), Exploiting the client vulnerabilities in internet e-voting systems: Hacking Helios 2.0 as an example, *in* D. W. Jones, J. Quisquater & E. Rescorla, eds, '2010 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '10, Washington, D.C., USA, August 9-10, 2010', USENIX Association.

Estonia (n.d.).
URL: *https://www.valimised.ee/eng/*

Fiat, A. & Shamir, A. (1986), How to prove yourself: Practical solutions to identification and signature problems, *in* A. M. Odlyzko, ed., 'Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings', Vol. 263 of *Lecture Notes in Computer Science*, Springer, pp. 186–194.

Gjøsteen, K. (2010), 'Analysis of an internet voting protocol', *IACR Cryptology ePrint Archive* **2010**, 380.

Gjøsteen, K. (2013), 'The Norwegian internet voting protocol', *IACR Cryptology ePrint Archive* **2013**, 473.

Goldwasser, S., Micali, S. & Rackoff, C. (1985), The knowledge complexity of interactive proof-systems (extended abstract), *in* R. Sedgewick, ed., 'Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA', ACM, pp. 291–304.

Heiderich, M., Frosch, T., Niemietz, M. & Schwenk, J. (2011), The bug that made me president: a browser- and web-security case study on Helios voting, *in* A. Kiayias & H. Lipmaa, eds, 'E-Voting and Identity - Third International Conference, VoteID 2011, Tallinn, Estonia, September 28-30, 2011, Revised Selected Papers', Vol. 7187 of *Lecture Notes in Computer Science*, Springer, pp. 89–103.

Helios (n.d.), 'Helios privacy claims'. Last accessed: 2014-07-31.
URL: *https://vote.heliosvoting.org/privacy*

Jefferson, D. R., Rubin, A. D., Simons, B. & Wagner, D. (2004), 'Analyzing internet voting security', *Commun. ACM* **47**(10), 59–64.

Juels, A., Catalano, D. & Jakobsson, M. (2005), Coercion-resistant electronic elections, *in* V. Atluri, S. D. C. di Vimercati & R. Dingledine, eds, 'Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, Alexandria, VA, USA, November 7, 2005', ACM, pp. 61–70.

Kiayias, A., Korman, M. & Walluck, D. (2006), An internet voting system supporting user privacy, *in* '22nd Annual Computer Security Applications Conference (ACSAC 2006), 11-15 December 2006, Miami Beach, Florida, USA', IEEE Computer Society, pp. 165–174.

Kiayias, A., Zacharias, T. & Zhang, B. (2015*a*), DEMOS-2: scalable E2E verifiable elections without random oracles, *in* I. Ray, N. Li & C. Kruegel, eds, 'Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015', ACM, pp. 352–363.
URL: *http://dl.acm.org/citation.cfm?id=2810103*

Kiayias, A., Zacharias, T. & Zhang, B. (2015*b*), End-to-end verifiable elections in the standard model, *in* E. Oswald & M. Fischlin, eds, 'Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II', Vol. 9057 of *Lecture Notes in Computer Science*, Springer, pp. 468–498.

Kiayias, A., Zacharias, T. & Zhang, B. (2015*c*), On the necessity of auditing for election privacy in e-voting systems, *in* S. K. Katsikas & A. B. Sideridis, eds, 'E-Democracy - Citizen Rights in the World of the New Computing Paradigms - 6th International Conference, E-Democracy 2015, Athens, Greece, December 10-11, 2015, Proceedings', Vol. 570 of *Communications in Computer and Information Science*, Springer, pp. 3–17.
URL: *http://dx.doi.org/10.1007/978-3-319-27164-4*

Kremer, S., Ryan, M. & Smyth, B. (2010), Election verifiability in electronic voting protocols, *in* D. Gritzalis, B. Preneel & M. Theoharidou, eds, 'Computer Security - ESORICS 2010, 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings', Vol. 6345 of *Lecture Notes in Computer Science*, Springer, pp. 389–404.

Küsters, R., Truderung, T. & Vogt, A. (2012), Clash attacks on the verifiability of e-voting systems, *in* 'IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA', IEEE Computer Society, pp. 395–409.

Kutylowski, M. & Zagórski, F. (2010), Scratch, Click & Vote: E2E voting over the internet, *in* D. Chaum, M. Jakobsson, R. L. Rivest, P. Y. A. Ryan, J. Benaloh, M. Kutylowski & B. Adida, eds, 'Towards Trustworthy Elections, New Directions in Electronic Voting', Vol. 6000 of *Lecture Notes in Computer Science*, Springer, pp. 343–356.

Pedersen, T. P. (1991), A threshold cryptosystem without a trusted party (extended abstract), *in* D. W. Davies, ed., 'Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings', Vol. 547 of *Lecture Notes in Computer Science*, Springer, pp. 522–526.

Smyth, B., Frink, S. & Clarkson, M. R. (2015), 'Computational election verifiability: Definitions and an analysis of Helios and JCJ', *IACR Cryptology ePrint Archive* **2015**, 233.

Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M. & Halderman, J. A. (2014), Security analysis of the Estonian internet voting system, *in* G. Ahn, M. Yung & N. Li, eds, 'Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014', ACM, pp. 703–715.

Tsoukalas, G., Papadimitriou, K., Louridas, P. & Tsanakas, P. (2013), From Helios to Zeus, in *2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013 2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013* (2013).

v4, H. (n.d.), 'Helios github repository'. Last accessed: 2014-07-31.
URL: *https://github.com/benadida/helios-server*

Zagórski, F., Carback, R., Chaum, D., Clark, J., Essex, A. & Vora, P. L. (2013), Remotegrity: Design and use of an end-to-end verifiable remote voting system, *in* M. J. J. Jr., M. E. Locasto, P. Mohassel & R. Safavi-Naini, eds, 'Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings', Vol. 7954 of *Lecture Notes in Computer Science*, Springer, pp. 441–457.