Epoque: Practical End-to-End Verifiable Post-Quantum-Secure E-Voting

Xavier Boyen
Queensland University of Technology
Australia
firstname.secondname@qut.edu.au

Thomas Haines
NTNU Trondheim
Norway
firstname.secondname@ntnu.no

Johannes Müller
SnT, University of Luxembourg
Luxembourg
firstname.secondname@uni.lu

***Abstract*—The ultimate goal in modern secure e-voting is to enable everyone to verify whether the final election result correctly reflects the votes chosen by the (human) voters, without exposing how each individual voted. These fundamental security properties are called *end-to-end verifiability* and *voter privacy*. Unfortunately, it turns out to be very challenging to pursue these properties simultaneously, especially when the latter must be future-proofed against the rise of quantum computers. In this work, we show, for the first time, a practical approach to do this.**

We present Epoque, the first end-to-end verifiable, voter-private, post-quantum-secure homomorphic e-voting protocol. It achieves its properties through the combination of practical lattice-based cryptographic primitives only, in a novel way. We formally prove all our security claims under common trust and hardness assumptions. At the core of Epoque lies an efficient identity-based encryption (IBE) scheme with blazingly fast master-key decryption. It is the component that makes the efficient tallying of thousands or millions of ballots a practical possibility. In order to demonstrate its practicality, we fully implemented it and provide detailed benchmarks; we believe this latter contribution is of independent interest beyond the specific e-voting application.

1. Introduction

E-voting systems are now widely used in national, state-wide, and municipal elections all over the world with several hundred million voters so far. The results of these elections make a fundamental impact on the lives of all of us, directly or indirectly. Therefore, it is important to ensure that e-voting systems map the secret inputs provided by the voters to the correct final result, even in the presence of extremely powerful and sophisticated adversaries.

Unfortunately, numerous security studies of real e-voting systems (see, e.g., [10], [35], [42], [63], [66]) demonstrate that there is a high risk that the published election result does not reflect how voters actually voted. Therefore, modern e-voting protocols strive for what is called *end-to-end verifiability* (see, e.g., [2], [8], [17], [20], [46], [47]). This fundamental security property enables voters and external observers to check whether the published election result is correct, even if, for instance, voting devices have programming errors or tallying authorities are outright malicious. Several such systems have already been deployed in real binding elections (see, e.g., [2], [3], [15], [17], [27], [39]). In Switzerland and Norway, for example, e-voting systems for national and local elections and referendums are required to provide verifiability [33], [41].

To date, the security of all practical end-to-end verifiable e-voting protocols relies on "traditional" hardness assumptions, such as factoring integers or computing discrete logarithms. With more and more powerful quantum computers on the horizon (see, e.g., [5]), these voting protocols may be rendered completely insecure. This threat motivates the design of end-to-end verifiable e-voting protocols that are secure against quantum attacks. Unfortunately, it turned out to be very challenging to pursue this objective, and, in fact, it had not been met prior to our work.

The reason behind this state of affairs is that naïvely replacing the "classical" cryptographic primitives of an arbitrary end-to-end verifiable e-voting protocol (e.g., Helios [2]) with known post-quantum primitives can destroy practicality. Despite the fact that post-quantum-secure cryptography has become more efficient and versatile in the past decade or so, there exist only the following two *practical* post-quantum-secure e-voting protocols in the literature. Boyen, Haines, and Müller [13] proposed and implemented a completely lattice-based verifiable decryption mix net which can be used for verifiable post-quantum-secure e-voting but the class of elections it should be used for is limited (see Sec. 8). Del Pino, Lyubashevsky, Neven, and Seiler [31] instantiated the homomorphic e-voting protocol by Cramer, Franklin, Schoenmakers, and Yung [26] with practical lattice-based cryptographic primitives. However, unlike Boyen et al.'s mix net [13], the homomorphic e-voting protocol by Del Pino et al. [31] is not (end-to-end) verifiable: we will elaborate in Sec. 2 that *all* tallying authorities and *all* voters' voting devices in [31] need to be honest in order to (be able to) verify that the final election result is in fact correct. As we will see, it has long been far from obvious how to eliminate these undesirable trust assumptions in the lattice-based setting without undermining practicality.

Altogether, there does not exist a homomorphic e-voting protocol in the literature that can be used in a real practical election to both protect the privacy of votes *and* provide end-to-end verifiability in the presence of quantum attackers.[1]

1. There exist verifiable e-voting protocols in the literature with unconditional privacy and thus post-quantum privacy (see, e.g., [29], [59]) but their verifiability reduces to quantum-insecure hardness assumptions.

2021 IEEE European Symposium on Security and Privacy (EuroS&P) | 978-1-6654-1491-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/EUROSP51992.2021.00027

DOI 10.1109/EuroSP51992.2021.00027

1.1. Our contributions

We propose Epoque, the first provably secure end-to-end verifiable e-voting protocol for post-quantum-secure elections in the real world.

Instead of relying on specific primitives, the security of the generic Epoque e-voting protocol (Section 3) can be guaranteed under certain assumptions these primitives have to satisfy. More precisely, Epoque follows (and generalizes) the homomorphic secret-sharing approach previously taken by [26], [31] but completely eliminates any trust on the tallying authorities for verifiability. To this end, the (generic) Epoque e-voting protocol employs (generic) identity-based encryption (IBE).[2]

We demonstrate that the generic Epoque e-voting protocol can be instantiated with practical and purely lattice-based cryptographic primitives (Section 4). For this purpose, we propose a new version of the prominent lattice-based IBE scheme by Agrawal, Boneh, and Boyen [4] (Section 5). Compared to [4], our IBE scheme provides a blazingly fast additional master-key decryption algorithm, which does not change the IBE security properties but proves particularly useful for our application. We have implemented a prototype of the extended IBE and provide detailed benchmarks illustrating its practicality (Section 6). Though it is only a tweak of the ABB IBE [4], we expect it to be of independent interest and to find good uses beyond the specific e-voting application.

We formally prove that the generic Epoque e-voting protocol achieves vote privacy, verifiability, and even accountability which guarantees that misbehaving parties can also be identified (Section 7). We emphasize that Epoque provides these properties under standard and transparent trust assumptions.

We further show how Epoque can be extended with a lightweight return code scheme (which does not require any further cryptographic primitives) in order to mitigate trust on the voters' voting devices (Appendix A).

1.2. Structure of the paper

We start by describing the general concept of e-voting based on homomorphic secret-sharing (Section 2). In particular, we will elaborate on the security issues of the previous works [26], [31] following this approach, and explain why they cannot be solved easily. After that, the structure of the paper follows the one of the contributions as described above. We discuss the main properties of Epoque and its related work in Section 8.

2. Homomorphic Secret-Sharing E-Voting

We have mentioned in the introduction that the design of our protocol follows the homomorphic secret-sharing approach for e-voting, originally proposed by Cramer et al. [26]. Their e-voting protocol is based on Pedersen commitments and Schnorr-like ZKPs. Since the security of these primitives relies on the hardness of the discrete logarithm problem, the protocol by Cramer et al. could be broken by a (sufficiently strong) quantum computer. More recently, Del Pino et al. [31] demonstrated that [26] can instead be instantiated with practical lattice-based primitives. We will refer to the conceptual design of [31]-plus-[26] as *basic* homomorphic secret-sharing e-voting.

In this section, we will first describe how basic homomorphic secret-sharing e-voting works and the (limited) security it provides (Section 2.1). After that, we will demonstrate that a *single* malicious tallying authority can block any incoming ballot with impunity, and that this issue not only undermines correctness but also vote privacy (Section 2.2). Eventually, we will explain why protecting against malicious voting devices is challenging when usability is taken into account (Section 2.3).

In the remainder of the paper, we will describe and formally analyze how to solve the fundamental security issues at the root of the above dilemma, and do so efficiently. It bears repeating that our solution can be instantiated with efficient lattice-based primitives available today. As a result, we obtain the first practical, wholly lattice-based, e-voting system with end-to-end verifiability.

Notation. Let $(X, +)$ be a finite Abelian group. Throughout this paper, whenever we say that an element $x \in X$ is *secretely shared* among n parties, we mean that $n-1$ elements $x_1, \ldots, x_{n-1}$ have been chosen uniformly at random from X, and $x_n \leftarrow x - \sum_{i=1}^{n-1} x_i$. We write $\langle x \rangle = (x_1, \ldots, x_n)$ to denote a secret sharing of x.

2.1. Overview

Basic homomorphic secret-sharing e-voting works as follows (see also Table 1). We have a number of voters $\mathsf{V}_1, \ldots, \mathsf{V}_{n_\mathsf{V}}$ and trustees $\mathsf{T}_1, \ldots, \mathsf{T}_{n_\mathsf{T}}$. We use a homomorphic commitment scheme and a public-key encryption scheme.

2.1.1. Vote casting. Each voter V_i first chooses her vote $v^i \in \{0, 1\}$ and then secretely shares v^i among the n_T trustees[3]:

$$\langle v^i \rangle = (v^i_1, \ldots, v^i_{n_\mathsf{T}}). \tag{1}$$

After that, V_i commits to each share v^i_k with randomness r^i_k and obtains a commitment c^i_k. Due to the homomorphic property of the commitment scheme, we have that $c^i \leftarrow \sum_k c^i_k$ is a commitment to $v^i = \sum_k v^i_k$ with randomness $r^i \leftarrow \sum_k r^i_k$.

Furthermore, V_i generates a zero-knowledge proof (ZKP) of knowledge of an opening to a valid vote, i.e., that c^i is a commitment to either 0 or 1.

Additionally, for each trustee T_k, voter V_i encrypts (v^i_k, r^i_k) under T_k's public key pk_k and obtains a ciphertext e^i_k. The voter's final ballot consists of all the commitments $(c^i_k)_k$, the ZKP of correctness, and the encrypted openings $(e^i_k)_k$. Eventually, V_i posts her ballot b^i on the bulletin board.

2.1.2. Ballot weeding. First, all ballots with invalid ZKPs are removed.[4] Then, each trustee T_k decrypts the ciphertext e^i_k of each unremoved ballot b^i with its secret key

2. To be clear: Epoque builds on the *cryptographic primitive* of IBE; key-escrow and authority objections to turnkey IBE systems do not apply.

3. Notice that the superscript in v^i is used here to designate a voter; it does not indicate an exponentiation.

4. In what follows, for the sake of simplicity, we assume that all ballots have valid ZKPs.

TABLE 1: Homomorphic Secret-Sharing E-Voting with Honest Participants

Voters				Trustee T_1	$\cdots$	Trustee $\mathsf{T}_{n_\mathsf{T}}$
v^1	$\xrightarrow{\textit{share}}$	$(v_1^1, \ldots, v_{n_\mathsf{T}}^1)$	$\xrightarrow{\textit{commit}}$	$\mathsf{Com}(v_1^1)$	$\cdots$	$\mathsf{Com}(v_{n_\mathsf{T}}^1)$
$\vdots$				$\vdots$		$\vdots$
v^{n_V}	$\xrightarrow{\textit{share}}$	$(v_1^{n_\mathsf{V}}, \ldots, v_{n_\mathsf{T}}^{n_\mathsf{V}})$	$\xrightarrow{\textit{commit}}$	$\mathsf{Com}(v_1^{n_\mathsf{V}})$	$\cdots$	$\mathsf{Com}(v_{n_\mathsf{T}}^{n_\mathsf{V}})$
				$\downarrow$ *sum*		$\downarrow$ *sum*
$\downarrow$ *sum*				$\sum_{i=1}^{n_\mathsf{V}} \mathsf{Com}(v_1^i)$	$\cdots$	$\sum_{i=1}^{n_\mathsf{V}} \mathsf{Com}(v_{n_\mathsf{T}}^i)$
				$\downarrow$ *open*		$\downarrow$ *open*
$\sum_{i=1}^{n_\mathsf{V}} v^i$	$=$	$\sum_{k=1}^{n_\mathsf{T}} \sum_{i=1}^{n_\mathsf{V}} v_k^i$	$\xleftarrow{\textit{sum}}$	$\sum_{i=1}^{n_\mathsf{V}} v_1^i$	$\cdots$	$\sum_{i=1}^{n_\mathsf{V}} v_{n_\mathsf{T}}^i$

Remarks: (1) The upper part of the table illustrates the casting phase and lower part the tallying phase. Recall that the commitment scheme is homomorphic. We note that the sum over all plain votes (left column) is implicit. (See Section 2 for the notation.)
(2) For the sake of simplicity, this overview table assumes that all participants (voters and trustees) are honest. We note that if participants are malicious, then the fundamental disputes (as described in Section 2.2) can occur in the "open" phase.

to obtain $(\tilde{v}_k^i, \tilde{r}_k^i)$. If $(\tilde{v}_k^i, \tilde{r}_k^i)$ is not a valid opening of c_k^i, then T_k publishes a complaint that e_k^i was invalid and V_i's ballot b^i is removed. Let $I \subseteq \{1, \ldots, n_\mathsf{V}\}$ refer to the set of unremoved ballots.

2.1.3. Tallying. Each trustee T_k publishes

$$v_k \leftarrow \sum_{i \in I} v_k^i \text{ and } r_k \leftarrow \sum_{i \in I} r_k^i. \tag{2}$$

The final result is then

$$\mathsf{res} \leftarrow \sum_k v_k. \tag{3}$$

2.1.4. Security. If the final election result res does not correspond to the votes contained in the *unremoved* ballots $(b^i)_{i \in I}$, then this can be detected. This is due to the binding and homomorphic properties of the commitment scheme which guarantee that

$$\mathsf{res} = \sum_k v_k = \sum_k \sum_{i \in I} v_k^i = \sum_{i \in I} \sum_k v_k^i = \sum_{i \in I} v^i. \tag{4}$$

Therefore, the tallying of the *unremoved* ballots $(b^i)_{i \in I}$ is a verifiable operation.

Furthermore, the IND-CCA security of the encryption scheme and the hiding property of the commitment scheme guarantee that the tallying does not reveal more information about the single votes $(v^i)_{i \in I}$ inside the *unremoved* ballots $(b^i)_{i \in I}$ than what can be derived from the final result res.

However, when we regard the *complete* voting protocol, verifiability is no longer guaranteed. More precisely, as we will demonstrate in what follows, it is not possible to verify whether a (single!) malicious trustee (Section 2.2) or some malicious voting devices (Section 2.3) have tampered with the voters' votes. This undermines not just verifiability, but also vote privacy.

2.2. Malicious tallying authorities

In this section, we focus on the threat of malicious trustees in basic homomorphic secret-sharing e-voting as well as the challenge of protecting against it in a lattice-based setting.

2.2.1. Attacks. Observe that if a trustee T_k claims that some ciphertext e_k^i was invalid, then b^i is removed—regardless of whether T_k's complaint is correct! In particular, if T_k is dishonest, then T_k could effectively "block" any ballot b^i without having to provide any evidence. Hence, basic homomorphic secret-sharing e-voting does not guarantee verifiability against a single malicious trustee.

This verifiability issue also affects vote privacy, since if T_k were to block incoming ballots selectively, perhaps based on metadata, then the remaining ballots would be left in a smaller "anonymity set" and with possibly much decreased privacy if the attacker chose a set with high correlations or from other a priori information. We refer to Cortier and Lallemand [23] for more details on the relation between verifiability and vote privacy, but even with a single corrupted trustee, the adversary has a significant advantage in breaking vote privacy.

2.2.2. Defenses. Del Pino et al. [31] acknowledge the aforementioned issue and propose to let each voter V_i store the randomness used for each encryption e_k^i so that, in case V_i's ballot is incorrectly claimed invalid, V_i could reveal the respective randomness. Although this approach may be appealing at a theoretical level, its implicit assumptions are problematic in a real-world election because the trustees in [31] need to use their secret tallying keys in order to verify whether a ballot is valid. In a real election, however, these tallying keys should be encapsulated outside the actual tallying phase to ensure that they are physically inaccessible by a potentially malicious environment.[5] Because incoming ballots could no longer be verified immediately after being submitted, as proposed by [31], in a real election, each voter would have to store all the critical information of her ballot (plain vote, random coins, etc.) until her ballot was verified at a later point. This would be problematic, first, because the voters would have to store sensitive information beyond the casting phase, and second, because the voters' voting/verification ceremony would become so complex that it is questionable whether

5. For example, in the e-voting system used for national elections in Estonia, it is specified that the "processing of votes is carried out in an off-line environment" and that the private key "cannot be used before the process of counting of votes" [38].

enough voters would even attempt, much less complete, the required checks.

In Section 3, we propose a novel solution based on identity-based encryption (IBE) to overcome the verifiability gap without introducing any restrictions. The resulting e-voting protocol, named Epoque, is practical and can completely be instantiated with lattice-based cryptographic primitives (Section 4). We formally prove that Epoque provides verifiability and vote privacy in the presence of fully malicious tallying authorities under common trust assumptions (Section 7.2 and 7.3).

Due to the latest developments of more efficient ZKPs of correct decryption (most notably [36]), there exist alternative practically efficient solutions to the one presented in this paper.[6] If we require that a trustee T_k, who claims that some ciphertext e_k^i decrypts to an invalid opening of the respective commitment, proves this claim in ZK, then a malicious trustee can no longer block incoming valid ballots, while, at the same time, an honest trustee can still correctly reject invalid ballots without revealing information on her secret key sk_k. Both approaches, the one using IBE and the one using ZKPs of correct decryption, are practical solutions for the previously open verifiability and privacy problems of [31] (see Sec. 8).

2.3. Malicious voting devices

In this section, we focus on the threat of malicious voting devices in basic homomorphic secret-sharing e-voting, and the challenge of defeating them in a usable way.

2.3.1. Impact. It is obvious that basic homomorphic secret-sharing e-voting does not protect against malicious voting devices, neither in terms of verifiability nor privacy. To see that verifiability is broken, assume that a corrupted voting device replaces the candidate entered by the human voter by a different one: the human voter would not have any means to detect this manipulation. To see that vote privacy cannot be guaranteed, recall that a voting device always receives the human voter's chosen candidate in clear.

2.3.2. Mitigations. Ensuring vote privacy against a malicious voting device typically involves using some kind of verifiable re-encryption mix net for which, to date, there does not exist sufficiently practical lattice-based instantiations (see Section 8). Therefore, in what follows, we restrict our attention to *verifying* possibly malicious voting devices, further focusing on correctness of in-band transmissions (i.e., of the ballots), since preventing out-of-band data exfiltration from a malicious system is really a hardware problem.

One approach for verifying voting devices is the *challenge-or-cast* technique by Benaloh [9]. It is, for instance, employed in the Helios e-voting system [2]. From a technical perspective, it would be straightforward to employ the challenge-or-cast technique in basic secret-sharing e-voting as well. However, several usability studies [1], [11], [44], [57], [58], [60] indicate that only few human voters successfully execute the challenge-or-cast gambit, even within the IACR's[7] own Helios elections. We have therefore decided not to follow this approach for Epoque.

A different technique for verifying the correct behaviour of voting devices is based on *return codes*. This solution was, for example, used for binding political elections in Switzerland [39] and Norway [6], [41]. We will follow this approach. In Appendix A, we propose a lightweight return code scheme for Epoque (Section 3) that can be instantiated without any additional cryptographic primitives.

3. Description of Epoque

In this section, we introduce the Epoque e-voting protocol. In Section 3.1, we explain the general idea of how Epoque extends the basic homomorphic secret-sharing e-voting protocol in order to protect against malicious tallying authorities (recall Section 2.2). In Section 3.2, we describe Epoque in full technical detail.

3.1. Idea

Recall from Section 2.2 that we are essentially facing the following problem when protecting against malicious trustees: How can T_k publicly prove that e_k^i encrypts (or fails to encrypt) certain message(s) m under pk_k without revealing any information on the remaining messages encrypted under pk_k? How can this be realized with practical lattice-based cryptographic primitives?

We propose to use the following idea [30] to solve this problem effectively. Instead of using a PKE scheme, we employ an IBE scheme (see Appendix B) with chosen-plaintext security (IND-ID-CPA) and the following property: Given a master public key mpk, an identity i, and an individual secret key msk^i, it can be efficiently decided whether msk^i is correct, i.e., corresponds to i w.r.t. mpk.

Now in Epoque, we assume that each trustee T_k holds a master public key mpk_k instead of an "ordinary" public key pk_k as in the basic secret-sharing e-voting protocol. In order to secretly send her k-th opening values (v_k^i, r_k^i) to T_k, voter V_i uses T_k's master public key mpk_k together with her identity i (instead of pk_k as in the basic protocol) to encrypt (v_k^i, r_k^i). We denote the resulting ciphertext by e_k^i as before.

Why does this technique enable a public observer/third party to verify the complaint of a (possibly) malicious trustee? To see this, consider the following two relevant cases (either T_k or V_i is honest):

- *Case 1*: Assume that T_k is honest and that V_i is dishonest. Assume that V_i creates an invalid ciphertext e_k^i. In this case, T_k uses its master secret key msk_k to derive V_i's individual secret key msk_k^i. After that, T_k publishes msk_k^i so that everyone can first verify the correctness of msk_k^i, and then decrypt e_k^i to verify T_k's complaint. Due to the IND-ID-CPA security of the IBE scheme, revealing msk_k^i does not leak any information about the vote shares of the remaining voters.

6. The paper by Esgin, Nguyen, and Seiler [36] has been presented at Asiacrypt after we had submitted our paper to EuroS&P. We thank the anonymous referees for pointing us to the work by Esgin et al. [36].

7. The International Association for Cryptologic Research.

- *Case 2*: Conversely, assume that T_k is dishonest and that V_i is honest. Since V_i is honest, her ballot b^i, including all encrypted opening values e^i_k, is valid. Assume that T_k wants to block V_i's ballot by incorrectly claiming that e^i_k was invalid. In order to "convince" a public observer of its false statement, T_k would have to publish an element, say x, such that (i) x is a correct individual secret key corresponding to V_i, and (ii) using x, e^i_k decrypts to a message different from the message that V_i encrypted under her identity i w.r.t. mpk_k. Due to the correctness of the IBE scheme, this is impossible. Therefore, a dishonest trustee can no longer "block" honestly generated ballots.

This argument demonstrates that our technique solves the verifiability (and, hence, vote privacy) issues of the basic homomorphic secret-sharing e-voting in the presence of malicious trustees. Importantly—and this is the crucial point!—, there exist highly practical lattice-based instantiations of IBE with the required features (Section 4).

3.2. Protocol

We now present the Epoque e-voting protocol in full detail. As mentioned in Section 1, instead of relying on specific primitives, the security of Epoque can be proven from generic assumptions these primitives have to satisfy. In Section 4, we show how to instantiate Epoque from practical lattice-based cryptographic primitives.

3.2.1. Cryptographic primitives. We require the following:

- A computationally hiding and computationally binding *homomorphic commitment scheme* $(\mathsf{KeyGen}_{\mathsf{com}}, \mathsf{Com}, \mathsf{Open})$.
- A NIZKPoK[8] scheme for creating proofs π_{V} of knowledge of a correct shared committed vote, i.e., whose sum over all committed shares opens to either 0 or 1. This can be described by the following relation $\mathcal{R}_{\mathsf{V}}$. Let $\mathsf{prm}_{\mathsf{com}}$ be an arbitrary output of $\mathsf{KeyGen}_{\mathsf{com}}(1^\ell)$. Then, a tuple $(\mathsf{prm}_{\mathsf{com}}, c^j_k, (r^j_k, m^j_k))_{k\in K, j\in J}$ is in $\mathcal{R}_{\mathsf{V}}$ if and only if
  - $\sum_{k\in K} m^j_k \in \{0,1\}$ for all $j \in J$, and
  - $\sum_{j\in J}\sum_{k\in K} m^j_k \in \{0,1\}$, and
  - $\mathsf{Open}(\mathsf{prm}_{\mathsf{com}}, m^j_k, c^j_k, r^j_k) = 1$ for all $k \in K$ and all $j \in J$.
- An *identity-based encryption (IBE) scheme* $(\mathsf{KeyGen}_{\mathsf{ibe}}, \mathsf{Extr}, \mathsf{Enc}, \mathsf{Dec})$ which is IND-ID-CPA-secure (Appendix B). We require that the correctness of an individual secret key given the public parameters can efficiently be decided. More precisely, this correctness property can be described by the following relation $\mathcal{R}_{\mathsf{ibe}}$. Let $\mathsf{prm}_{\mathsf{ibe}}$ be an arbitrary output of $\mathsf{KeyGen}_{\mathsf{ibe}}(1^\ell)$, and id be a valid identity. Then, a tuple $(\mathsf{prm}_{\mathsf{ibe}}, \mathsf{msk}^{\mathsf{id}}, \mathsf{id})$ is in $\mathcal{R}_{\mathsf{ibe}}$ if and only if there exist random coins r such that $(\mathsf{prm}_{\mathsf{ibe}}, \mathsf{msk}) = \mathsf{KeyGen}_{\mathsf{ibe}}(r)$ and $\mathsf{msk}^{\mathsf{id}} = \mathsf{Extr}(\mathsf{prm}_{\mathsf{ibe}}, \mathsf{msk}, \mathsf{id})$.

8. Non-interactive zero-knowledge proof of knowledge.

3.2.2. Protocol participants. Epoque is run among the following participants:

- voting authority AT,
- human voters $\mathsf{V}_1, \ldots, \mathsf{V}_{n_\mathsf{V}}$,
- voters' supporting devices $\mathsf{VSD}_1, \ldots, \mathsf{VSD}_{n_\mathsf{V}}$,
- trustees $\mathsf{T}_1, \ldots, \mathsf{T}_{n_\mathsf{T}}$, and
- a public, append-only bulletin board B.

We assume that for each party there exists a mutually authenticated channel to B.[9]

3.2.3. Protocol overview. A protocol run of Epoque consists of the following phases: setup, ballot creation, ballot submission, ballot weeding, voter verification, tallying, and public verification. We now explain each phase in more details.

3.2.4. Setup. In this phase, all election parameters are fixed and posted on the bulletin board.

The voting authority AT determines and publishes the security parameter ℓ, the number of candidates n_{cand}, the list $\{1, \ldots, n_\mathsf{V}\}$ of eligible voters, opening and closing times, the election identity $\mathsf{id}_{\mathsf{election}}$, etc. Afterwards, the voting authority runs the key generation algorithm of the commitment scheme and publishes $\mathsf{prm}_{\mathsf{com}} \leftarrow \mathsf{KeyGen}_{\mathsf{com}}(1^\ell)$.

Each trustee T_k runs the key generation algorithm $\mathsf{KeyGen}_{\mathsf{ibe}}$ of the IBE scheme to generate its public parameters $\mathsf{prm}_k = \mathsf{prm}_{\mathsf{ibe},k}$ and its master secret key msk_k. After that, T_k publishes prm_k on the bulletin board.

3.2.5. Ballot creation. In this phase, every voter V_i can vote for some candidate $j' \in \{1, \ldots, n_{\mathsf{cand}}\}$. The voter V_i enters her chosen candidate to her supporting device VSD_i. On input $j' \in \{1, \ldots, n_{\mathsf{cand}}\}$, VSD_i encodes j' as a binary vector

$$v^i = (v^{i,j})_{j=1}^{n_{\mathsf{cand}}} \in \{0,1\}^{n_{\mathsf{cand}}} \subseteq M_{\mathsf{com}}^{n_{\mathsf{cand}}}, \tag{5}$$

where $v^{i,j'} = 1$. Then for all candidates $j \in \{1, \ldots, n_{\mathsf{cand}}\}$, VSD_i secretely shares $v^{i,j}$ among the trustees

$$\langle v^{i,j} \rangle = (v^{i,j}_1, \ldots, v^{i,j}_{n_\mathsf{T}}). \tag{6}$$

Afterwards, for all $j \in \{1, \ldots, n_{\mathsf{cand}}\}$ and all trustees T_k, VSD_i runs the commitment algorithm Com with input $v^{i,j}_k$:

$$(c^{i,j}_k, r^{i,j}_k) \leftarrow \mathsf{Com}(\mathsf{prm}_{\mathsf{com}}, v^{i,j}_k). \tag{7}$$

Since the commitment scheme is homomorphic, the commitment $c^{i,j} \leftarrow \sum_{k=1}^{n_\mathsf{T}} c^{i,j}_k$ decommits to $v^{i,j}$ using opening value $r^{i,j} \leftarrow \sum_{k=1}^{n_\mathsf{T}} r^{i,j}_k$, i.e.,

$$\mathsf{Open}(\mathsf{prm}_{\mathsf{com}}, v^{i,j}, c^{i,j}, r^{i,j}) = 1. \tag{8}$$

In order to guarantee the well-formedness and knowledge of all committed shares $c^{i,j}_k$, VSD_i creates a NIZKP of knowledge π^i_{V} for proving that VSD_i knows messages $v^{i,j}_k \in M_{\mathsf{com}}$ and opening values $r^{i,j}_k \in R$ such that

- for all $j \in \{1, \ldots, n_{\mathsf{cand}}\}$, it holds that $\sum_{k=1}^{n_\mathsf{T}} v^{i,j}_k \in \{0,1\}$, and

9. In practice, as in Helios-C [20] or Belenios [22], one could establish a PKI among the voters so that they can sign their ballots.

- $\sum_{j=1}^{n_{\mathsf{cand}}} \sum_{k=1}^{n_{\mathsf{T}}} v_k^{i,j} \in \{0,1\}$, and
- for all $j \in \{1,\ldots,n_{\mathsf{cand}}\}$ and all $k \in \{1,\ldots,n_{\mathsf{T}}\}$, it holds that $\mathsf{Open}(\mathsf{prm}_{\mathsf{com}}, v_k^{i,j}, c_k^{i,j}, r_k^{i,j}) = 1$.

For each trustee T_k, VSD_i reads T_k's public parameters prm_k from the bulletin board B. Then, VSD_i encrypts the tuple $(v_k^{i,j}, r_k^{i,j})_{j=1}^{n_{\mathsf{cand}}}$ using T_k's public parameters prm_k and V_i's identity i, i.e.,

$$e_k^i \leftarrow \mathsf{Enc}(\mathsf{prm}_k, i; (v_k^{i,j}, r_k^{i,j})_{j=1}^{n_{\mathsf{cand}}}). \tag{9}$$

Eventually, VSD_i returns a message to V_i stating that the ballot is ready for submission.

3.2.6. Ballot submission. VSD_i submits the ballot

$$b^i \leftarrow (i, (c_k^{i,j})_{j,k}, (e_k^i)_k, \pi_{\mathsf{V}}^i) \tag{10}$$

to the bulletin board B. For each incoming ballot b^i, B checks whether

- V_i has not yet submitted a ballot before, and
- b^i does not contain a duplicate entry of another ballot $b^j \in \mathsf{B}$, and
- π_{V}^i is valid.

If all checks are positive, then B adds b^i to the (initially empty) list of ballots $\vec{b}$ and publicly updates $\vec{b}$.

3.2.7. Ballot weeding. Each trustee T_k reads $\vec{b}$ from the bulletin board B. For each ballot $b^i \in \vec{b}$, trustee T_k uses msk_k to decrypt e_k^i.

If decryption fails, then T_k runs the IBE private-key extraction algorithm Extr to derive V_i's individual secret key $\mathsf{msk}_k^i \leftarrow \mathsf{Extr}(\mathsf{prm}_k, \mathsf{msk}_k, i)$, and publishes (i, msk_k^i) as proof that e_k^i is malformed.

If decryption succeeds, then T_k parses the resulting plaintext as $(v_k^{i,j}, r_k^{i,j})_{j=1}^{n_{\mathsf{cand}}}$, and verifies whether

$$\mathsf{Open}(\mathsf{prm}_{\mathsf{com}}, v_k^{i,j}, c_k^{i,j}, r_k^{i,j}) = 1 \tag{11}$$

holds true for all $j \in \{1,\ldots,n_{\mathsf{cand}}\}$. In other words, T_k verifies whether the resulting plaintext contains opening values for all commitments $c_k^{i,1}, \ldots, c_k^{i,n_{\mathsf{cand}}}$ assigned by voter V_i to trustee T_k. If this check fails, then T_k extracts msk_k^i as described above and publishes (i, msk_k^i) as proof that e_k^i is invalid.

Eventually, after T_k has verified all ballots b^i in $\vec{b}$ as described, T_k sends a respective message to the bulletin board.

If, in this phase, a trustee T_k publishes (i, x) for some voter V_i, then everyone can efficiently verify whether $(\mathsf{prm}_{\mathsf{ibe},k}, x, i) \in \mathcal{R}_{\mathsf{ibe}}$, i.e., whether x is indeed the (or, at least, a) valid individual secret key for V_i w.r.t. T_k's public IBE parameters $\mathsf{prm}_{\mathsf{ibe},k}$. Furthermore, everyone can use x to verify b^i in the same (deterministic) way as T_k has done internally before.

3.2.8. Tallying. Those ballots which have not been proven invalid by any of the trustees are processed as follows.[10] Each trustee T_k publishes

$$v_k^j \leftarrow \sum_{i=1}^{n_{\mathsf{V}}} v_k^{i,j} \text{ and } r_k^j \leftarrow \sum_{i=1}^{n_{\mathsf{V}}} r_k^{i,j} \tag{12}$$

for all $j \in \{1,\ldots,n_{\mathsf{cand}}\}$. Everyone can verify the correctness of v_k^j, r_k^j by checking

$$1 \stackrel{?}{\leftarrow} \mathsf{Open}(\mathsf{prm}_{\mathsf{com}}, v_k^j, c_k^j, r_k^j), \tag{13}$$

where $c_k^j \leftarrow \sum_{i=1}^{n_{\mathsf{V}}} c_k^{i,j}$.

The final election result can publicly be computed as

$$\mathsf{res} \leftarrow \left(\sum_{k=1}^{n_{\mathsf{T}}} v_k^1, \ldots, \sum_{k=1}^{n_{\mathsf{T}}} v_k^{n_{\mathsf{cand}}} \right). \tag{14}$$

3.2.9. Public verification. In this phase, every participant, including the voters themselves and external auditors, can verify correctness of both the ballot weeding phase and the tallying phase.

4. Lattice-Based Instantiation

In this section, we demonstrate how to instantiate the generic Epoque protocol (Section 3) with lattice-based cryptographic primitives only. We describe different options to efficiently instantiate the generic IBE scheme which differ in terms of security and efficiency. Before that, following Del Pino et al. [31], we briefly describe concrete lattice-based instantiations of the generic commitment scheme and the NIZKPs of well-formedness.

4.1. Commitment scheme

We use the lattice-based commitment scheme proposed by Del Pino et al. [31] which is computationally binding under the Module Short Integer Solution (M-SIS) problem and computationally hiding under the Module Learning With Errors (M-LWE) problem [54]. We refer to [31] for details of their construction.

4.2. NIZKP of well-formedness

We follow Del Pino et al. [31] by realizing π^{V} by the conjunction of the statements proved by two other NIZKPs. This strategically places most of the computational effort on the authorities, who are able to use amortized proofs for greater efficiency. We note that this NIZKP is also a proof of knowledge (PoK) which is important for our optimisation of the IBE decryption (see below).

We illustrate this approach for an election with two candidates. In this case, each voter first commits either to 0 or to 1 (for candidate A or B, respectively), and then generates a NIZKP π^{V} for proving that c^i is a commitment either to 0 or to 1. However, such an *exact* NIZKP is computationally complex for lattice-based commitments which would be in conflict with the requirement of a lightweight casting procedure. In order to solve this issue, Del Pino et al. [31] proposed the following solution. Instead of proving the exact relation above, each voter V_i proves an *approximate* variant of it. Then, each trustee T_k proves that the commitment c_k^i assigned by V_i has small randomness. These two proofs collectively imply the exact relation (i.e., that c^i is a commitment either to 0 or to 1). Importantly, each trustee can prove that the randomness is small for many commitments at once so that its cost

10. For the sake of readability, we now suppose all submitted ballots valid.

can be amortized over many voters. We refer the reader to [31] for much of the details, in particular for why these two proofs collectively imply a ZKP as required for π_{V}.

We note that there exist further subsequent innovations in the literature (e.g., [12], [36], [37], [67]) that open up other interesting alternatives for realizing this primitive.

4.3. IBE scheme

We elaborate on three different options to instantiate the generic IBE scheme efficiently in the lattice-based setting.

4.3.1. Based on RLWE in the ROM. The most straightforward solution is to instantiate the generic IBE scheme with a lattice-based one whose security reduces to the *ring* learning-with-errors (RLWE) hardness assumption in the random oracle model (ROM). The lattice-based IBE scheme by Ducas, Lyubashevsky, and Prest [34] would be a good candidate for this purpose.[11] However, applying such an IBE scheme comes along with a possible security weakening. Firstly, while LWE is known to be at least as hard as standard (worst-case) problems on euclidean lattices, RLWE is only known to be as hard as their restrictions to special classes of ideal lattices, corresponding to ideals of some polynomial rings. Secondly, if the encryption scheme is secure only in the ROM, then the long-term privacy of ballots is threatened by unforeseen future cryptanalyses of the hash functions employed in the election.[12]

4.3.2. Based on LWE in the standard model. In order to avoid the aforementioned possible security issues, we prefer to employ an IBE scheme in the standard model, such as the prominent one by Agrawal, Boneh, and Boyen (ABB) [4], whose security reduces to the *plain* LWE hardness assumption. However, if we applied the ABB scheme directly to our scenario, then Epoque would be rendered inefficient because for each ballot, the voter's respective individual secret key would need to be extracted. Because the extraction algorithm is in the order of 1 min (see Table 2), this approach would be too slow for processing possibly millions of ballots. In Sec. 5, we propose a modified version of the ABB IBE scheme with fast master decryption which solves this efficiency problem, as explained next. Unlike in the original ABB IBE scheme, the master secret key holder (the trustee in Epoque) can use a shortcut decryption algorithm to decrypt each ciphertext independently of the individual public key the ciphertext had been created with. Now, the trustees in Epoque can use this shortcut decryption algorithm to decrypt each voter's ciphertext, without having to (slowly) extract the respective individual secret key. In Sec. 6 (Table 2), we provide detailed benchmarks of our implementation of the modified ABB IBE scheme from Sec. 5 which demonstrate that, due to this modification, decryption can be done blazingly fast. In short, our shortcut yields a 4-order-of-magnitude speed improvement over the original ABB IBE.

Altogether, following the approach taken in this paper, we obtain a practically efficient instantiation of the generic IBE scheme in Epoque which is more secure than the ones based on RLWE in the ROM.

4.3.3. Combination of IBE and PKE. Alternatively to our modification of the ABB scheme to avoid slow decryption on the trustees' side, we now describe how to combine an arbitrary IBE scheme, e.g., original ABB IBE [4], with an arbitrary efficient lattice-based PKE scheme.[13]

Let $\mathsf{Enc}_{\mathsf{ibe}}$ denote the encryption algorithm of an IND-ID-CPA-secure IBE scheme, and $\mathsf{Enc}_{\mathsf{pke}}$ the encryption algorithm of an IND-CPA-secure PKE scheme. In the setup phase, each trustee T_k runs the key generation algorithm of the IBE scheme to obtain public parameters prm_k and a master secret key msk_k. Now, additionally, each trustee T_k runs the key generation algorithm of the PKE scheme to obtain a public/private key pair $(\mathsf{pk}_k, \mathsf{sk}_k)$. In the ballot creation phase, voter V_i encrypts the opening values $(v_k^{i,j}, r_k^{i,j})_{j=1}^{n_{\mathsf{cand}}}$ under T_k's public key pk_k

$$e_k^i \leftarrow \mathsf{Enc}_{\mathsf{pke}}(\mathsf{pk}_k; (v_k^{i,j}, r_k^{i,j})_{j=1}^{n_{\mathsf{cand}}}), \tag{15}$$

and then encrypts the randomness $\hat{r}_k^i$ used to create e_k^i under V_i's individual public key, i.e., the combination of T_k's public parameters prm_k and V_i's identity i:

$$\hat{e}_k^i \leftarrow \mathsf{Enc}_{\mathsf{ibe}}(\mathsf{prm}_k, i; \hat{r}_k^i). \tag{16}$$

Note that the difference between this technique and the abstract one described in the generic Epoque protocol (Sec. 3.2) is that the voter does not encrypt the opening values under its individual public key (prm_k, i) but the randomness used to encrypt these opening values.

In the ballot weeding phase, T_k first decrypts the PKE ciphertext e_k^i and verifies whether the resulting plaintext is a valid opening for the respective commitments. If this is the case, T_k simply discards the IBE ciphertext $\hat{e}_k^i$. If and only if e_k^i does not decrypt to a valid opening, then T_k runs the extraction algorithm of the IBE scheme to derive and publish V_i's individual secret key msk_k^i so that everyone can decrypt the IBE ciphertext $\hat{e}_k^i$. Everyone can then use the resulting plaintext in combination with the trustee's public key pk_k to decrypt the PKE ciphertext e_k^i and verify that e_k^i does not decrypt to a valid opening of the respective commitments.

Note that, using the above technique, the trustees need to run the expensive extraction algorithm only if a ciphertext decrypts to an invalid opening. Because this will rarely if ever need to be done in an election, this technique could also be realized efficiently with the original, i.e., unmodified, ABB IBE scheme [4]. On the contrary, the technique proposed in Sec. 4.3.2 is more compact in that it requires only one ciphertext per ballot, unlike two in the latter technique.

11. Ducas et al. implemented their IBE scheme on a standard computer and they report that generating an individual secret key takes less than 33ms, while both encryption and decryption take less than 2ms each (for a security level of 192 bits). Applied to our application, a trustee can decrypt the ciphertexts of roughly 30 ballots per second (not taking into account the trustee's further actions).

12. Practitioners may push back on any criticism of random oracles, but the argument is strongly supported, both on theoretical and empirical grounds: (1) being keyless, hash functions are the trickiest to design and essentially the most vulnerable of all symmetric-key constructions, a fact that has been eloquently demonstrated in (2) very-high-profile breaks of universally used hash functions theretofore believed secure (e.g., SHA-1 [55] or MD5 [64]).

13. We thank the anonymous EuroS&P referee who proposed this alternative approach.

5. Lattice-Based IBE with Fast Master Decryption

Our IND-ID-CPA-secure IBE instantiation is based on the Agrawal-Boneh-Boyen (ABB) IBE scheme in the standard model [4] whose security reduces to the plain learning-with-errors (LWE) hardness assumption. This IBE scheme was selected primarily because we can transform the general master key msk normally used for identity-based private-key extraction, into a special master key that can decrypt all well-formed ciphertexts regardless of the ciphertext recipient, extremely efficiently.

In what follows, we briefly recall some basic details of the ABB IBE [4] first. After that, we explain how to construct an efficient master decryption key without affecting the security of the original construction.

5.1. Agrawal-Boneh-Boyen IBE scheme

Let n be the security parameter. We choose q, m, σ, α such that

$$m = 6n^{1+\delta} \qquad q = m^{2.5} \cdot \omega(\sqrt{\log n}) \tag{17}$$

$$\sigma = m \cdot \omega(\sqrt{\log n}) \qquad \alpha = [m^2 \cdot \omega(\sqrt{\log n})]^{-1} \tag{18}$$

hold true, where we round up m to the nearest larger integer and q to the nearest larger prime. We assume that δ is such that $n^\delta > \lceil \log q \rceil = \mathcal{O}(\log n)$.

We use a function $H\colon \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ to map identities in $\mathbb{Z}_q^n$ to matrices in $\mathbb{Z}_q^{n\times n}$. The function H needs to satisfy the following notion of injectivity:

$$\forall \mathsf{id}_1, \mathsf{id}_2 \in \mathbb{Z}_q^n\colon (\mathsf{id}_1 \neq \mathsf{id}_2 \Rightarrow \det(H(\mathsf{id}_1) - H(\mathsf{id}_2)) \neq 0). \tag{19}$$

We refer to [4] for an explicit construction of H.

The *public parameters*

$$\mathsf{prm} = (A_0, A_1, B, u) \tag{20}$$

consist of a three random matrices $A_0, A_1, B \in \mathbb{Z}_q^{n\times m}$ plus a vector $u \in \mathbb{Z}_q^n$.

The *master secret*

$$\mathsf{msk} = (T_{A_0}) \tag{21}$$

consists of a basis with low Gram-Schmidt norm ($\leq \mathcal{O}(\sqrt{n \log q})$) for the lattice $\Lambda_q^\perp(A_0)$. For the non-specialist, this is to say that T_{A_0} is an $m \times m$ integer matrix of full rank but low norm, whose column vectors mod q are all orthogonal to A_0, i.e.,

$$A_0 \cdot T_{A_0} = 0 \pmod q \tag{22}$$

It is relatively easy to generate T_{A_0} at the same time as a random A_0 is created, but this is believed to be infeasibly hard after the fact, even quantumly.

The *extraction algorithm* Extr on input $\mathsf{prm}, \mathsf{msk}, \mathsf{id}$ runs a certain sampling algorithm (see [4] for details) that returns a short (low-norm) integer $2m$-vector d_{id} such that

$$[A_0|A_1 + H(\mathsf{id}) \cdot B] \cdot d_{\mathsf{id}} = u, \tag{23}$$

where $F_{\mathsf{id}} \leftarrow [A_0|A_1 + H(\mathsf{id}) \cdot B]$ is a publicly constructable $n \times 2m$ matrix which varies with id.

The *encryption algorithm* Enc takes $\mathsf{prm}, \mathsf{id}$ and a message $b \in \{0, 1\}$ as input. It first chooses uniformly random $s \in \mathbb{Z}_q^n$ and $R \in \{-1, 1\}^{m\times m}$. After that, noise vectors $x \in \mathbb{Z}_q$ and $y \in \mathbb{Z}_q^m$ are generated according to some distribution χ for which the LWE problem is hard (e.g., as hard as the worst-case SIVP and GapSVP under a quantum reduction; see [61] for details). The resulting ciphertext

$$(c_0, c_1) \in \mathbb{Z}_q \times \mathbb{Z}_q^{2m} \tag{24}$$

consists of

$$c_0 \leftarrow u^T s + x + b \cdot \lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q, \tag{25}$$

$$c_1 \leftarrow F_{\mathsf{id}}^T s + \begin{bmatrix} y \\ R^T y \end{bmatrix} \in \mathbb{Z}_q^{2m}. \tag{26}$$

The *decryption algorithm* Dec on input $\mathsf{prm}, \mathsf{msk}^{\mathsf{id}} = d_{\mathsf{id}}$ and ciphertext (c_0, c_1) returns 1 if

$$\left| c_0 - d_{\mathsf{id}}^T c_1 - \lfloor \frac{q}{2} \rfloor \right| < \lfloor \frac{q}{4} \rfloor \tag{27}$$

in $\mathbb{Z}$, and otherwise 0.

For completeness, whereas [4] describes using a ciphertext $(c_0, c_1) \in \mathbb{Z}_q \times \mathbb{Z}_q^{2m}$ to encrypt a single bit of message, it is noted that multi-bit encryption is possible by adding instances of $c_0, c_0', c_0'', ... \in \mathbb{Z}_q$, all constructed using the same encryption randomness s so that they can all share the same ciphertext vector $c_1 \in \mathbb{Z}_q^{2m}$. This requires preparing a matching number of independent random vectors $u, u', u'', ... \in \mathbb{Z}_q^n$ in the IBE public key. We shall use this for efficiency.

5.2. Our construction

In the original ABB scheme, to decrypt a ciphertext encrypted under an individual public key F_{id}, it is first necessary to extract the respective individual secret key d_{id}. Unfortunately, the performance of the extraction algorithm Extr is comparatively slow (see Table 2 for benchmarks of Extr), because it involves a somewhat costly Gaussian integer sampling, required to prevent the extracted private key from leaking details of the master key. If however the master-key holder is doing the decryption, one possibility to speed things up is to degrade the Gaussian sampler, as long as the resulting "unsafe" private key is functional and not disseminated—but we can do even better.

To make master-key-based decryption really fast we designed an additional "master shortcut decryption" mechanism, able to decrypt any recipient's ciphertext immediately from the master secret key msk, without having to extract an individual private key first, and without changing (what the outside world sees of) the rest of the scheme. Conceptually, this adds no functionality to the IBE scheme, nor does it remove any security from it; but the benefit is that such master shortcut decryption is blazingly fast, as demonstrated by our performance benchmarks in Section 6 (Table 2).

Speedy master decryption is particularly well motivated for Epoque, where each tallier T_k is required to perform one decryption for each voter's ballot, all using different ids, but only infrequently (if at all) does it need to publish a properly extracted private key for any voter. Without shortcut decryption, each tallier T_k would have to extract a use-once-and-discard private key for every single voter. Master shortcut decryption is much more practical in large-scale elections.

5.2.1. Master shortcut decryption. In ABB IBE, the only two properties for a private key d_{id} to work are that $\|d_{\mathsf{id}}\|_2 < \beta$ for some β, and that $F(\mathsf{id}) \cdot d_{\mathsf{id}} = u$ per Eq. 23. Additional properties on the *distribution* of d_{id} are only necessary to prevent d_{id} from leaking information about msk, either upon extraction of d_{id}, or, in a chosen-ciphertext attack, upon decryption of specially crafted ciphertexts using d_{id} (more on that below). Absent those two circumstances, the master-key holder is able safely to use any d_{msk} for decryption, provided that $\|d_{\mathsf{msk}}\|_2 < \beta$ and $F(\mathsf{id}) \cdot d_{\mathsf{msk}} = u \pmod q$. We show how to construct such d_{msk} that work for all id.

First, in the master-key generation phase, we construct the random matrix B with a trapdoor T_B, i.e., s.t.,

$$A_2 \cdot T_B = 0 \pmod q \quad , \quad \|T_B\| < \beta' \tag{28}$$

just like we did for A_0 and T_{A_0}. Note that this can be done while A_2 remains vanishingly close to a uniform distribution over $\mathbb{Z}_q^{n \times m}$.

Then, we obtain $d_{\mathsf{msk}} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$ by sampling, in order:

1) Using T_{A_2}, a low-norm vector $d_2 \in \Lambda_q^{\perp}(A_2)$, which is to say that $A_2 \cdot d_2 = 0$;
2) Using T_{A_0}, a low-norm vector $d_1 \in \Lambda_q^{-A_1 d_2}(A_0)$, i.e., such that $A_0 \cdot d_1 = -A_1 \cdot d_2$.

Notice that both equations above are vector equalities in $\mathbb{Z}_q^n$, infeasible to solve for low-norm solutions except with knowledge of the respective trapdoors T_{A_2} and T_{A_0}, e.g., using the SampleLeft() algorithm from [4].

The *Master shortcut decryption algorithm* is then exactly the same as regular decryption, substituting d_{msk} for d_{id}. Note that master shortcut decryption works regardless of id, since

$$\forall \mathsf{id}, \quad [A_0 | A_1 + H(\mathsf{id}) \cdot B] \cdot d_{\mathsf{msk}} = u \tag{29}$$

Using d_{msk} in the same way as a normal user would use d_{id}, a tallyer is able to decrypt all (well-formed) ciphertexts extremely quickly.

5.2.2. Security. The addition of a master decryption algorithm does not alter the *chosen-plaintext* (IND-ID-CPA) semantic security of the IBE scheme, since all user data (including public and private keys) have the same distribution as in the original IND-ID-CPA secure construction [4], and for the same reason the modified scheme is also correct.[14] More precisely, Game 2 of the original security reduction (see Sec. 6.4 of [4]) contains the same modification of original ABB that we use here for master shortcut decryption. This means that the original security reduction applies to our modification immediately.

14. Technically, the basic ABB IBE and our variant only satisfy a weaker notion of security against *selective-identity attacks*, denoted IND-sID-CPA, wherein the attacker has to announce in advance the identity it is going to target. Since in Epoque the assigned voter IDs actually live in a "cryptographically tiny" domain of mere thousands or millions, the distinction between ID and sID security is inconsequential. Specifically, any sID-secure IBE is also fullID-secure with an ϵ advantage divided by a factor no greater than the size of the domain of IDs, here quite small. This distinction is also orthogonal to CPA versus CCA security.

5.3. Secure employment in Epoque

In what follows, we argue that the IBE with fast master shortcut decryption can be used securely in the Epoque protocol. We first explain that publishing an individual secret key d_{id} of a dishonest voter can leak some information on the master shortcut key d_{msk} but that this leakage is practically negligible (as already proved in [4]).

5.3.1. Leakage of individual secret keys. Privacy of Epoque relies on the assumption that an honest trustee never reveals any individual secret key of an honest voter. This assumption could be violated indirectly through the use of master (shortcut) decryption. To see this, observe that when an honest tallier reveals the key of some voter V, it is *also* indicating that V is *dishonest* in the sense that her ciphertext is either invalid or malformed. Ciphertexts which are only slightly malformed (perhaps intentionally made so), will decrypt correctly with some non-trivial probability (neither 0 or 1), depending on how the malformation "aligns" with the decryption key d_{id} or d_{msk} (recall that d is a vector in an Euclidean space, and thus has a direction). Through this mechanism, a crafty attacker is theoretically able to obtain information about the decryption key. This is completely inconsequential when d_{id} is used for decryption, since d_{id} itself is being revealed on decryption failures, but when d_{msk} is used, the disclosure of d_{id} is actually leaking a minute amount of information about d_{msk}. Since d_{msk} is a valid key for every user, including honest ones, the assumption is being violated.

5.3.2. Amount of leakage. The intuition behind the existing formal proof that this "attack" does not work at all, is that merely gathering some information about d_{msk}, of which there are exponentially many, is useless unless d_{msk} can be reconstructed *exactly*. (Most lattice-based cryptosystems, including ABB IBE, tolerate and in fact require noise on the ciphertext, but not on the private keys.) But, since the ciphertext-validity-testing oracle above is *non-interactive*, an exponential number of malformed voter ballots would have to be submitted before the information gained will allow accurate reconstruction of d_{msk}.[15]

6. Implementation and Benchmarks

We provide benchmarks of our instantiation of Epoque presented in Section 4. We first present detailed benchmarks of the new IBE scheme which show that our instantiation of the IBE scheme is very efficient in practice. After that, we recall some experimental results by Del Pino et al. [31] who implemented the lattice-based commitment scheme and ZKPs, leaving the encryption scheme unspecified. These *partial* performance results of the voting scheme from [31] apply immediately to the *complete* Epoque system, because the overhead we introduce with the IBE is practically negligible (see Table 2). Altogether, Epoque is, at worst, of nearly identical speed, but is significantly more secure than [31].

15. This is the same mechanics whereby interactive binary search can isolate an element in $O(\log N)$ comparisons, while non-interactive search on the same domain would need $O(N)$ of the same comparisons (or $O(\sqrt{N})$ if quantum). Here N is the size of the domain, so $N = 2^{\ell}$, indeed exponential in the bit-length ℓ of the data being searched.

6.1. IBE scheme

In Table 2, we provide benchmarks of our implementation of the IBE scheme presented in Section 5. Our implementation is optimised first for decryption which is the main bottleneck, then encryption, key extraction, and finally key generation, with the latter two currently using a generic Gaussian sampler which has room for improvement.[16]

Our experiments show that while master-key generation starts to take significant time at the higher security levels, using such IBE system as part of Epoque, both to encrypt ballot shares by the voters, and to decrypt them by the authority, are still blazingly fast, respectively taking 11 ms and 4 μs per bit of IBE ciphertext at the "medium-high" security level indicated in Table 2, which by current estimates for lattice cryptography would correspond to a security parameter λ somewhere between 128 and 192 bits.

Encryption and decryption times increase linearly with ciphertext size at first, then plateau at λ bits and beyond, since for long messages one would use hybrid IBE+Symmetric encryption scheme, where the IBE is used to encrypt a λ-bit ephemeral key for a suitable symmetric cipher mode of operation (an unauthenticated mode, since we only require chosen-plaintext security).

Individual private key extraction at this security level is a bit lengthier at several seconds per identity, but recall that this will rarely if ever need to be done in an election. The mere existence of this functionality acts as a deterrent against voters intentionally crafting incorrect ballot shares.

In terms of ciphertext and public-key *sizes*, accommodating λ-bit plaintexts (or more with hybrid encryption) is not much different than 1-bit plaintexts. The bulk of the ciphertext consists of the n-vector s (where, e.g., $n \approx 512$), to which one would add some n' additional elements $y \in \mathbb{Z}_q$, where $\lambda/\log_2 q < n' \leq \lambda$ (e.g., for $\lambda = 192$, take $n' = 48$ or 64). In summary, the total ciphertext overhead, excluding the size of the message itself, will be slightly in excess of the "smaller" lattice dimension n times $\log_2 q$ bits.

6.2. Commitment scheme and NIZKP

For the commitment scheme and the NIZKP π^{V}, Del Pino et al. [31] give results of an experiment conducted with 11000 voters and 4 trustees that we briefly recall here.[17] They report that each voter device takes about 8.5 ms for creating the commitments and the (approximate) NIZKP per trustee, and that its total size is roughly 15 KB per trustee. They further report that it takes each trustee 0.15 sec per voter, in total for creating the amortized exact NIZKP and for tallying the ballots.

16. Our test machine was an 8th-generation Intel i7 laptop, with 3.3 GHz single-thread measured clockspeed. All mitigations against speculative-execution attacks on Intel have been applied. The entire implementation is in plain C from the ground up, linking only to the standard C math library and only for the Gaussian sampler.

17. They used a laptop with an Intel Skylake i7 CPU running at 2.6 GHz.

7. Security Analysis

In this section, we formally analyze the security of the generic Epoque protocol in terms of verifiability and privacy.

7.1. Computational Model

We start by formally modeling Epoque using a general and established computational framework (see, e.g., [21], [51], [52]) that we can use both for analyzing verifiability and privacy of Epoque.

The computational model introduces the notion of a process which can be used to model protocols (we recall some details in Appendix C). Essentially, a process $\hat{\pi}_{\mathsf{P}}$ modeling some protocol P is a set of interacting ppt Turing machines which capture the honest behavior of protocol participants. The protocol P runs alongside an adversary A, modeled via another process π_{A}, which controls the network and may corrupt protocol participants; here we assume static corruption. We write $\pi{=}(\hat{\pi}_{\mathsf{P}} \| \pi_{\mathsf{A}})$ for the combined process.

7.1.1. Modeling of Epoque. The Epoque voting protocol can be modeled in a straightforward way as a protocol $\mathsf{P}_{\mathsf{Epoque}}(n_{\mathsf{V}}, n_{\mathsf{T}}, n_{\mathsf{cand}}, \mu)$ in the above sense, as detailed next. By n_{V} we denote the number of voters V_i and by n_{T} the number of trustees T_k. By n_{cand} we denote the number of candidates, and by μ we denote a probability distribution on the set of choices $\mathsf{C} = \{v \in \{0,1\}^{n_{\mathsf{cand}}} : \sum_{j=1}^{n_{\mathsf{cand}}} v[j] = 1\}$. An honest voter makes her choice according to this distribution.[18] This choice is called the *actual choice* of the voter.

In our model of Epoque, the voting authority AT is part of an additional agent, the scheduler S. Besides playing the role of the authority, S schedules all other agents in a run according to the protocol phases. We assume that S and the bulletin board B are honest, i.e., they are never corrupted. While S is merely a virtual entity, in reality, B should be implemented in a distributed way (see, e.g., [28], [43], [45]).

7.2. Verifiability

In this section, we establish the level of verifiability provided by Epoque. To this end, we use the generic and widely used verifiability definition proposed in [51] which we briefly recall first.

7.2.1. Verifiability framework. The verifiability definition [51] assumes a "virtual" entity, called the *judge* J, whose role is to either accept or reject a protocol run. In a real election, the program of the judge can be executed by any party, including external observers and even voters themselves. The judge takes as input solely public information (e.g., the zero-knowledge proofs in Epoque published on the bulletin board) to perform certain checks. In the context of e-voting, for verifiability to hold, the judge should only accept a run if "the announced election

18. This in particular models that adversaries know this distribution. In reality, the adversary might not know this distribution precisely. This, however, makes our security results only stronger.

TABLE 2: IBE Benchmarks

IBE Algorithm/Metric	Complexity	Low Security	Medium Security	Med-High Security	Higher Security
Dimensions $n \times 2m$		100×2400	200×4800	300×7200	400×9600
Max Ctx. Overhead	$O((n+\lambda) \cdot \log q)$	0.4 kB	0.6 kB	0.8 kB	1 kB
KeyGen	$O(m^3 \cdot \log^2 q)$	405 sec	3711 sec	11589 sec	35443 sec
Extr	$O(m \cdot n^2 \cdot \log^2 q)$	1 to 3 sec	6 to 8 sec	22 sec	53 sec
Enc (1 bit)	$O(m \cdot n \cdot \log^2 q)$	3 ms	7 ms	11 ms	17 ms
Dec (1 bit)	$O(m \cdot \log^2 q)$	1 μs	2 μs	3 to 4 μs	4 μs
Dec (192 bits)	$O(m \cdot \log^2 q)$	0.2 ms	0.4 ms	0.6 ms	0.8 ms

Remarks: (1) For all security levels in the experiments, we used the prime modulus $q = 4093$. (2) Private-key extraction timing has higher variance than the other operations, due to a combination of rejection sampling without the need for too many samples. (3) Encryption and especially decryption times are barely measurable, scaling only linearly and with very small constant factors. For 1-bit messages, we measured them using respectively 1000 and 1000000 iterations with precautions against the compiler possibly optimising some of the work away. (4) The $\log q$ and $\log^2 q$ factors in the complexity column respectively capture the asymptotic size and time factors of working in q-bit arithmetic, and can be ignored when q (here 12 bits) fits in a CPU register.

result corresponds to the actual choices of the voters". This statement is formalized via the notion of a *goal* γ of a protocol P. A goal γ is simply a set of protocol runs for which the mentioned statement is true, where the description of a run contains the description of the protocol, the adversary with which the protocol runs, and the random coins used by these entities.

Following [51], we say that a goal γ is *verifiable* by the judge J in a protocol P, if the probability that J accepts a run r of P even though the goal γ is violated (i.e., $r \notin \gamma$) is negligible in the security parameter. In the formal definition of verifiability (which is a bit shortened for brevity of presentation), we denote by $\Pr[(\hat{\pi}_\mathsf{P} \| \pi_\mathsf{A})^{(\ell)} \mapsto \neg\gamma, (\mathsf{J}\colon \mathsf{accept})]$ the probability that a run of the protocol along with an adversary π_A (and a security parameter ℓ) produces a run which is not in γ but in which J (nevertheless) returns accept. This probability should be negligible.

Definition 1 (Verifiability [51]). We say that *a goal* γ *is verifiable by the judge* J *in a protocol* P if for all adversaries π_A, the probability

$$\Pr[(\hat{\pi}_\mathsf{P} \| \pi_\mathsf{A})^{(\ell)} \mapsto \neg\gamma, (\mathsf{J}\colon \mathsf{accept})]$$

is negligible as a function of ℓ.

For our subsequent verifiability analysis of Epoque, we instantiate the verifiability definition with the goal $\gamma(\varphi)$ proposed in [21]. This goal captures the intuition of γ given before. The parameter φ is a Boolean formula to describe which protocol participants are assumed honest. The goal $\gamma(\varphi)$ is defined formally as described next (we simplified the definition for brevity of presentation).

Definition 2 (Goal $\gamma(\varphi)$ [21]). Let P be a voting protocol. Let I_h and I_d denote the set of honest and dishonest voters, respectively, in a given protocol run. Then, $\gamma(\varphi)$ consists of all those runs of the voting protocol P where either

- φ is false (e.g., the adversary corrupted a voter that is assumed to be honest), or
- φ holds true and there exist (valid) dishonest choices $(m_i)_{i \in I_d}$ such that the election result equals $(m_i)_{i \in I_h \cup I_d}$, where $(m_i)_{i \in I_h}$ are the honest voters' choices.

7.2.2. Analysis. We prove the verifiability result for Epoque under the following assumptions:

- (V1) The IBE scheme is correct (for verifiability, IND-ID-CPA-security is not needed), the commitment scheme is computationally binding, and π_V is a NIZKP.
- (V2) The scheduler S, the bulletin board B, the judge J, and all voter supporting devices VSD_i are honest:

$$\varphi = \mathsf{hon}(\mathsf{S}) \wedge \mathsf{hon}(\mathsf{B}) \wedge \mathsf{hon}(\mathsf{J}) \bigwedge_{i=1}^{n_\mathsf{V}} \mathsf{hon}(\mathsf{VSD}_i)$$

Note that an arbitrary number of voters and trustees may be controlled by the adversary. In Appendix A, we show how to mitigate trust on the voter supporting devices as well.

The verification procedure J of Epoque essentially involves checking individual secret keys revealed by the trustees (if any) and NIZKPs (see Appendix D for details). If one of these checks fails, the protocol run and hence the result are rejected. Now, essentially, the following theorem states that the probability that in a run of Epoque an honest voter's vote has been dropped or manipulated if φ holds true (i.e., $\gamma(\varphi)$ is broken) but the protocol run is nevertheless accepted by J is negligible.

Theorem 1 (Verifiability). Under the assumptions (V1) and (V2) stated above, the goal $\gamma(\varphi)$ is verifiable in the protocol $\mathsf{P}_{\mathsf{Epoque}}(n_\mathsf{V}, n_\mathsf{T}, n_{\mathsf{cand}}, \mu)$ by the judge J.

The correctness of Theorem 1 follows immediately from a stronger result. In fact, Epoque even provides *accountability* which implies verifiability as demonstrated in [51]. For verifiability, one requires only that, if the election outcome does not correspond to how the voters actually voted, then such a protocol run is not accepted. Verifiability however does not require the blame of misbehaving parties. On the contrary, accountability requires that misbehaving parties be (publicly) blamed, an important property in practice as misbehavior should be identifiable and have consequences: accountability serves as a deterrent. We state the accountability result of Epoque in Appendix E and formally prove it in [14].

7.3. Privacy

In this section, we carry out a rigorous analysis of the vote privacy of Epoque and show that the privacy level of Epoque is ideal. For this purpose, we use the privacy definition for e-voting protocols proposed in [52] which has already been used to analyze a number of further voting protocols and mix nets [48]–[50], [52], [53].

7.3.1. Definition. The definition proposed in [52] formalizes privacy of an e-voting protocol as the inability of an adversary to distinguish whether some voter $\mathsf{V}_{\mathsf{obs}}$ (the *voter under observation*), who runs her honest program, voted for choice ch_0 or choice ch_1.

To define this notion formally, we first introduce the following notation for an arbitrary e-voting protocol P. Given a voter $\mathsf{V}_{\mathsf{obs}}$ and $\mathsf{ch} \in \mathsf{C}$, we consider instances of P of the form $(\hat{\pi}_{\mathsf{V}_{\mathsf{obs}}}(\mathsf{ch}) \| \pi^* \| \pi_{\mathsf{A}})$ where $\hat{\pi}_{\mathsf{V}_{\mathsf{obs}}}(\mathsf{ch})$ is the honest program of the voter $\mathsf{V}_{\mathsf{obs}}$ under observation who takes ch as her choice, π^* is the composition of programs of the remaining parties in P, and π_{A} is the program of the adversary. In the case of Epoque, π^* includes the scheduler, the bulletin board, all other voters, and all trustees.

Let $\Pr[(\hat{\pi}_{\mathsf{V}_{\mathsf{obs}}}(\mathsf{ch}) \| \pi^* \| \pi_{\mathsf{A}})^{(\ell)} \mapsto 1]$ denote the probability that the adversary writes the output 1 on some dedicated tape in a run of $(\hat{\pi}_{\mathsf{V}_{\mathsf{obs}}}(\mathsf{ch}) \| \pi^* \| \pi_{\mathsf{A}})$ with security parameter ℓ and some $\mathsf{ch} \in \mathsf{C}$, where the probability is taken over the random coins used by the parties in $(\hat{\pi}_{\mathsf{V}_{\mathsf{obs}}}(\mathsf{ch}) \| \pi^* \| \pi_{\mathsf{A}})$.

Now, vote privacy is defined as follows, where for Epoque we quantify over all adversaries π_{A} which neither corrupt the bulletin nor the scheduler S.

Definition 3 (Privacy). Let P be a voting protocol, $\mathsf{V}_{\mathsf{obs}}$ be the voter under observation, and $\delta \in [0,1]$. Then, P *achieves* δ*-privacy*, if for all choices $\mathsf{ch}_0, \mathsf{ch}_1 \in \mathsf{C}$ and all adversaries π_{A} the difference

$$\Pr[(\hat{\pi}_{\mathsf{V}_{\mathsf{obs}}}(\mathsf{ch}_0) \| \pi^* \| \pi_{\mathsf{A}})^{(\ell)} \mapsto 1] - \Pr[(\hat{\pi}_{\mathsf{V}_{\mathsf{obs}}}(\mathsf{ch}_1) \| \pi^* \| \pi_{\mathsf{A}})^{(\ell)} \mapsto 1]$$

is δ-bounded[19] as a function of the security parameter 1^ℓ.

In other words, the level δ is an upper bound of an arbitrary adversary's advantage to "break" vote privacy. Therefore, δ should be as small as possible. Note, however, that even for an ideal e-voting protocol with a completely passive adversary, δ might not be 0: for example, there might be a non-negligible chance that all honest voters, including the voter under observation, voted for the same candidate, in which case the adversary can easily derive from the final election result how the voter under observation voted.

7.3.2. Analysis. We now state that Epoque provides ideal vote privacy in the case that at most $n_{\mathsf{T}}-1$ trustees are dishonest, where n_{T} is the number of trustees: clearly, if all trustees were dishonest, privacy could not be guaranteed because an adversary could simply open all commitments in the list of ballots.

19. A function f is δ-*bounded* if, for every $c > 0$, there exists ℓ_0 such that $|f(\ell)| \leq \delta + \ell^{-c}$ for all $\ell > \ell_0$.

More specifically, the formal privacy result for Epoque is formulated w.r.t. an ideal voting protocol $\mathcal{I}_{\mathsf{voting}}(n_{\mathsf{V}}, n_{\mathsf{cand}}, \mu)$. In this protocol, all n_{V} voters pick their candidates according to the distribution μ. The ideal protocol outputs the total number of votes per candidate. The privacy level $\delta^{\mathsf{ideal}}_{(n_{\mathsf{V}}, n_{\mathsf{cand}}, \mu)}$ this ideal protocol has depending on the given parameters was derived in [52].

To prove that the privacy level of Epoque is ideal, we make the following assumptions about the primitives we use (see also Section 3):

- (P1) The IBE scheme is IND-ID-CPA-secure, the commitment scheme is computationally binding and hiding, and π_{V} is a NIZKP of knowledge.
- (P2) An adversary π_{A} does neither corrupt the scheduler S, nor the bulletin B, nor all trustees, and at least $n_{\mathsf{V}}{}^{\mathsf{honest}}$ voters including their voting devices are honest.

Now, the following privacy theorem says that the privacy level of Epoque is ideal under the previous assumptions.

Theorem 2 (Privacy). Under the assumptions (P1) and (P2) stated above, the voting protocol $\mathsf{P}_{\mathsf{Epoque}}(n_{\mathsf{V}}, n_{\mathsf{T}}, n_{\mathsf{cand}}, \mu)$ achieves a privacy level of $\delta^{\mathsf{ideal}}_{(n_{\mathsf{V}}{}^{\mathsf{honest}}, n_{\mathsf{cand}}, \mu)}$.

The proof is provided in [14], where we use a sequence of games to reduce the privacy game for Epoque to the privacy game for the ideal voting protocol.

8. Discussion

In this section, we first discuss the main properties of Epoque, and then compare Epoque with the few related e-voting protocols that are designed to protect against quantum computers.

8.1. Lattice-based security

We have formally proven that the generic Epoque e-voting protocol guarantees vote privacy, as well as verifiability and accountability in the presence of malicious tallying authorities (Section 3 to 7.3). Furthermore, we have demonstrated how Epoque can be extended with a lightweight return code scheme to also protect against malicious voters' supporting devices (Appendix A). Since Epoque and its return code extension can be instantiated with purely lattice-based cryptographic primitives (Section 4), we obtained a completely lattice-based e-voting protocol with end-to-end verifiability and vote privacy.

8.2. Practicality

Our lattice-based instantiation of the generic Epoque e-voting protocol consists of three cryptographic primitives, namely our new IBE scheme, the commitment scheme and the ZKP of well-formedness. Del Pino et al. [31] demonstrated that the lattice-based instantiations of the commitment scheme and the ZKP of well-formedness are practical. In this work, we have fully implemented our lattice-based IBE scheme and provide detailed benchmarks to show its practicality (Section 6).

Altogether, in combination, this demonstrates that our lattice-based instantiation of Epoque is practical.

The core components scale linearly in the number of voters and trustees, and only logarithmically in the number of candidates (for simple voting rules). For the whole system, complexity is $\mathcal{O}(n \log n)$ in the number of ballots, since sorting will be needed, and $\mathcal{O}(n)$ in the number of candidates, if only to print them somewhere.

We note that the trustees can safely provide their service even if many malicious voters submit incorrect ciphertexts. Proving that a ciphertext decrypts to an invalid opening is not an urgent matter. If a ciphertext decrypts to an invalid opening, then the trustees can set aside the respective ballot and tally the remaining (valid) ballots to announce the election result without delay. Extracting and publishing the individual secret key of an incorrect ciphertext can be done afterwards. Also note that a voter who submits an invalid ballot effectively wastes her vote and can be identified (and thus held accountable) as well. Altogether, both the risk and the effect of such "DDoS" attacks is very small.

8.3. Coercion-resistance

We have formally proven that Epoque provides the most fundamental security properties each secure e-voting system must provide: privacy and verifiability (and even accountability). Beyond these fundamental properties, it is sometimes desirable that voters cannot be coerced to vote for a given candidate, or that they cannot sell their votes. Therefore, some e-voting systems were designed to resist against or to mitigate the possibility of coercing voters and selling votes (e.g., [16], [19], [62]). Typically, these e-voting systems employ very specific cryptographic primitives (e.g., re-randomizable signature schemes [16]). Instantiating such techniques with practical lattice-based primitives is an open and interesting question that we leave for future work.

8.4. ZKPs of correct decryption

As mentioned in Sec. 2.2, an alternative approach to the one followed in this paper is to employ a ZKP of correct decryption. Constructing such a ZKP for lattice-based encryption schemes has attracted much interest in the literature. Until very recently, existing results relied on classical hardness assumptions (e.g., [32]) or came with parameters indicating that they are not really practical (e.g., [7]) compared to the results we achieve here using the IBE "trick". As to the best of our knowledge, the first comparably efficient and completely lattice-based solution to this problem can be based on the ZKP system by Esgin, Nguyen, and Seiler [36].[20] Both Esgin et al.'s ZKP and our IBE technique do not introduce any significant overhead on the voters' or on the trustees' side. Only if a voter does not encrypt a valid opening, Esgin et al.'s ZKP enables a trustee to prove malformedness faster (but with larger proof size). Because this will rarely if ever need to be done in an election and can also be executed after the election result was announced (see above), both Esgin et al.'s ZKP as well as our IBE technique are similarly practical solutions to the same problem. Compared to Esgin et al.'s ZKP, the ABB IBE scheme [4] we extended is not only well-established but also conceptually simpler and thus easier to implement correctly—an aspect which must not be underestimated (see, e.g., [42]). Ultimately, it will be up to the implementer which option to choose.

20. Lyubashevsky, Nguyen, and Seiler [56] demonstrate how to further reduce the proof size of [36], assuming a new hardness assumption ("extended LWE") which is not yet established.

8.5. Related work

There are several e-voting protocols in the literature that aim to achieve *vote privacy* in the presence of quantum (see [40]) or even computationally unbounded adversaries (see, e.g., [29]). But only few (techniques for) e-voting protocols have been published so far that were designed to guarantee both vote privacy *and* verifiability against quantum attackers. In what follows, we elaborate on all of these works and their relationship to Epoque.

8.5.1. Del Pino et al. (CCS 2017). The e-voting protocol by Del Pino, Lyubashevsky, Neven, and Seiler [31] is a practical lattice-based instantiation of the basic homomorphic secret-sharing e-voting protocol by Cramer, Franklin, Schoenmakers, and Yung [26]. In particular, [31] inherits the (limited) security of [26] (see Section 2).

8.5.2. Chillotti et al. (PQCrypto 2016). The voting protocol by Chillotti, Gama, Georgieva, and Izabachène [18] is built upon a fully homomorphic lattice-based PKE scheme. Chillotti et al. do not provide any benchmarks but the employment of fully homomorphic encryption indicates that their voting protocol is not practical.

8.5.3. Boyen et al. (Esorics 2020). Boyen, Haines, and Müller [13] proposed and employed a generic technique, named *trip wires*, to make any decryption mix net verifiable without requiring any further cryptographic primitives (in particular, no ZKPs). Boyen et al. applied this technique to a completely lattice-based decryption mix net; the resulting protocol can directly be used for verifiable and highly practical lattice-based e-voting. In contrast to Epoque, [13] does not aim for a "perfect" but "high" verifiability level (which guarantees that the probability of being caught cheating increases exponentially in the number of manipulated votes), and [13] requires a set of auditors one of which needs to be trusted (temporarily). At the same time, in contrast to homomorphic protocols like Epoque, decryption mix nets such as [13] have the advantage to easily handle very complex ballots (e.g., for IRV or STV elections). Therefore, both protocols, Epoque and [13], are useful options for practical lattice-based e-voting, each of them with its own balance between security and the set of manageable voting methods.

8.5.4. Proofs of correct shuffle. None of the published lattice-based proofs of correct shuffling for re-encryption mix nets [24], [25], [65] has been implemented so far.

9. Conclusion

We proposed Epoque, the first end-to-end verifiable e-voting protocol that can completely be instantiated with

practical lattice-based cryptographic primitives. Epoque supports arbitrarily many voters and tallying authorities.

For this purpose, we proposed a new highly efficient version of the lattice-based Agrawal-Boneh-Boyen IBE scheme. We fully implemented this IBE scheme and provided detailed benchmarks for demonstrating the practicality of the IBE scheme and thus the one of Epoque.

We formally proved the security of Epoque in terms of verifiability, accountability, and vote privacy, each one under standard and transparent trust assumptions.

We also demonstrated how Epoque can be extended with a lightweight return code scheme in order to mitigate trust on the voters' voting devices.

Acknowledgements

We thank the anonymous EuroS&P referees for their helpful feedback, in particular for pointing us to related work on lattice-based ZKPs of correct decryption that we had not been aware of before, and for proposing an interesting alternative approach (see Sec. 4.3.3) to the one followed in this paper.

All authors acknowledge support from the Luxembourg National Research Fund (FNR) and the Research Council of Norway for the joint INTER project SURCVS (Number 11747298). The research of Johannes Müller was in part funded by FNR, grant reference number 11299247.

References

[1] Claudia Z. Acemyan, Philip T. Kortum, Michael D. Byrne, and Dan S. Wallach. Users' Mental Models for Three End-to-End Voting Systems: Helios, Prêt à Voter, and Scantegrity II. In *Human Aspects of Information Security, Privacy, and Trust - Third International Conference, HAS 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015. Proceedings*, pages 463–474, 2015.

[2] Ben Adida. Helios: Web-based Open-Audit Voting. In *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 335–348, 2008.

[3] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jaques Quisquater. Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In *USENIX/ACCURATE Electronic Voting Technology (EVT 2009)*, 2009.

[4] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient Lattice (H)IBE in the Standard Model. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, pages 553–572, 2010.

[5] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A. Buell, et al. Quantum Supremacy using a Programmable Superconducting Processor. *Nature*, 574(7779):505–510, 2019.

[6] Jordi Barrat, Michel Chevalier, Ben Goldsmith, David Jandura, John Turner, and Rakesh Sharma. Internet Voting and Individual Verifiability: The Norwegian Return Codes. In *5th International Conference on Electronic Voting 2012, (EVOTE 2012), Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC, July 11-14, 2012, Castle Hofen, Bregenz, Austria*, pages 35–45, 2012.

[7] Carsten Baum, Ivan Damgård, Kasper Green Larsen, and Michael Nielsen. How to Prove Knowledge of Small Secrets. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 478–498, 2016.

[8] Susan Bell, Josh Benaloh, Mike Byrne, Dana DeBeauvoir, Bryce Eakin, Gail Fischer, Philip Kortum, Neal McBurnett, Julian Montoya, Michelle Parker, Olivier Pereira, Philip Stark, Dan Wallach, and Michael Winn. STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System. *USENIX Journal of Election Technology and Systems (JETS)*, 1:18–37, August 2013.

[9] Josh Benaloh. Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In *2007 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'07, Boston, MA, USA, August 6, 2007*, 2007.

[10] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 626–643. Springer, 2012.

[11] David Bismark, James Heather, Roger M. A. Peel, Steve Schneider, Zhe Xia, and Peter Y. A. Ryan. Experiences Gained from the first Prêt à Voter Implementation. In *First International Workshop on Requirements Engineering for e-Voting Systems, RE-VOTE 2009, Atlanta, Georgia, USA, August 31, 2009*, pages 19–28, 2009.

[12] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic Techniques for Short(er) Exact Lattice-Based Zero-Knowledge Proofs. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, pages 176–202, 2019.

[13] Xavier Boyen, Thomas Haines, and Johannes Müller. A Verifiable and Practical Lattice-Based Decryption Mix Net with External Auditing. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 336–356. Springer, 2020.

[14] Xavier Boyen, Thomas Haines, and Johannes Müller. Epoque: Practical End-to-End Verifiable Post-Quantum-Secure E-Voting. *IACR Cryptol. ePrint Arch.*, 2021:304, 2021.

[15] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Vanessa Teague, Roland Wen, Zhe Xia, and Sriramkrishnan Srinivasan. Using Prêt à Voter in Victoria State Elections. In J. Alex Halderman and Olivier Pereira, editors, *2012 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '12, Bellevue, WA, USA, August 6-7, 2012*. USENIX Association, 2012.

[16] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1614–1625, 2016.

[17] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. In *2008 USENIX/ACCURATE Electronic Voting Workshop, EVT 2008, July 28-29, 2008, San Jose, CA, USA, Proceedings*, 2008.

[18] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. A Homomorphic LWE Based E-voting Scheme. In *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, pages 245–265, 2016.

[19] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a Secure Voting System. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 354–368, 2008.

[20] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election Verifiability for Helios under Weaker Trust Assumptions. In *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II*, pages 327–344, 2014.

[21] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. SoK: Verifiability Notions for E-Voting Protocols. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 779–798, 2016.

[22] Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. Belenios: A Simple Private and Verifiable Electronic Voting System. In Joshua D. Guttman, Carl E. Landwehr, José Meseguer, and Dusko Pavlovic, editors, *Foundations of Security, Protocols, and Equational Reasoning - Essays Dedicated to Catherine A. Meadows*, volume 11565 of *Lecture Notes in Computer Science*, pages 214–238. Springer, 2019.

[23] Véronique Cortier and Joseph Lallemand. Voting: You Can't Have Privacy without Individual Verifiability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018*, pages 53–66, 2018.

[24] Núria Costa, Ramiro Martínez, and Paz Morillo. Proof of a Shuffle for Lattice-Based Cryptography. In *Secure IT Systems - 22nd Nordic Conference, NordSec 2017, Tartu, Estonia, November 8-10, 2017, Proceedings*, pages 280–296, 2017.

[25] Núria Costa, Ramiro Martínez, and Paz Morillo. Lattice-Based Proof of a Shuffle. *IACR Cryptology ePrint Archive*, 2019:357, 2019.

[26] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-Autority Secret-Ballot Elections with Linear Work. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 72–83, 1996.

[27] Chris Culnane, Peter Y. A. Ryan, Steve A. Schneider, and Vanessa Teague. vVote: A Verifiable Voting System. *ACM Trans. Inf. Syst. Secur.*, 18(1):3, 2015.

[28] Chris Culnane and Steve A. Schneider. A Peered Bulletin Board for Robust Use in Verifiable Voting Systems. In *IEEE 27th Computer Security Foundations Symposium, CSF, 2014*, pages 169–183, 2014.

[29] Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, volume 8134 of *Lecture Notes in Computer Science*, pages 481–498. Springer, 2013.

[30] Ivan Damgård, Dennis Hofheinz, Eike Kiltz, and Rune Thorbek. Public-Key Encryption with Non-interactive Opening. In *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008. Proceedings*, volume 4964 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 2008.

[31] Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical Quantum-Safe Voting from Lattices. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1565–1581, 2017.

[32] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Short Discrete Log Proofs for FHE and Ring-LWE Ciphertexts. In *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part I*, pages 344–373, 2019.

[33] Der Bundesrat. Das Portal der Schweizer Regierung. Federal Chancellery Ordinance on Electronic Voting, 2013. https://www.admin.ch/opc/en/classified-compilation/20132343/index.html (accessed 02.11.2020).

[34] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient Identity-Based Encryption over NTRU Lattices. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2014.

[35] Jeremy Epstein. Weakness in Depth: A Voting Machine's Demise. *IEEE Security & Privacy*, 13(3):55–58, 2015.

[36] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical Exact Proofs from Lattices: New Techniques to Exploit Fully-Splitting Rings. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 259–288. Springer, 2020.

[37] Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-Based Zero-Knowledge Proofs: New Techniques for Shorter and Faster Constructions and Applications. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, pages 115–146, 2019.

[38] State Electoral Officeof Estonia. General Framework of Electronic Voting and Implementation thereof at National Elections in Estonia. https://www.valimised.ee/sites/default/files/uploads/eng/IVXV-UK-1.0-eng.pdf (accessed 17.02.2021).

[39] David Galindo, Sandra Guasch, and Jordi Puiggali. 2015 Neuchâtel's Cast-as-Intended Verification Mechanism. In *E-Voting and Identity - 5th International Conference, VoteID 2015, Bern, Switzerland, September 2-4, 2015, Proceedings*, pages 3–18, 2015.

[40] Huangyi Ge, Sze Yiu Chau, Victor E. Gonsalves, Huian Li, Tianhao Wang, Xukai Zou, and Ninghui Li. Koinonia: Verifiable E-Voting with Long-Term Privacy. In David Balenson, editor, *Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019*, pages 270–285. ACM, 2019.

[41] Kristian Gjøsteen. The Norwegian Internet Voting Protocol. In *E-Voting and Identity - Third International Conference, VoteID 2011, Tallinn, Estonia, September 28-30, 2011, Revised Selected Papers*, pages 1–18, 2011.

[42] Thomas Haines, Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. How not to prove your election outcome. In *IEEE Symposium on Security and Privacy*, pages 644–660. IEEE, 2020.

[43] Lucca Hirschi, Lara Schmid, and David A. Basin. Fixing the Achilles Heel of E-Voting: The Bulletin Board. *IACR Cryptol. ePrint Arch.*, 2020:109, 2020.

[44] Fatih Karayumak, Maina M. Olembo, Michaela Kauer, and Melanie Volkamer. Usability Analysis of Helios - An Open Source Verifiable Remote Electronic Voting System. In *2011 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '11, San Francisco, CA, USA, August 8-9, 2011*, 2011.

[45] Aggelos Kiayias, Annabell Kuldmaa, Helger Lipmaa, Janno Siim, and Thomas Zacharias. On the Security Properties of e-Voting Bulletin Boards. In *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Proceedings*, pages 505–523, 2018.

[46] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. DEMOS-2: Scalable E2E Verifiable Elections without Random Oracles. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 352–363. ACM, 2015.

[47] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-End Verifiable Elections in the Standard Model. In *Advances in Cryptology - EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 468–498. Springer, 2015.

[48] Ralf Küsters, Julian Liedtke, Johannes Müller, Daniel Rausch, and Andreas Vogt. Ordinos: A Verifiable Tally-Hiding E-Voting System. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020*. IEEE, 2020 (in press).

[49] Ralf Küsters, Johannes Müller, Enrico Scapin, and Tomasz Truderung. sElect: A Lightweight Verifiable Remote Voting System. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 341–354, 2016.

[50] Ralf Küsters and Tomasz Truderung. Security Analysis of Re-Encryption RPC Mix Nets. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 227–242. IEEE, 2016.

[51] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and Relationship to Verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 526–535, 2010.

[52] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*, pages 538–553, 2011.

[53] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Formal Analysis of Chaumian Mix Nets with Randomized Partial Checking. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 343–358. IEEE Computer Society, 2014.

[54] Adeline Langlois and Damien Stehlé. Worst-Case to Average-Case Reductions for Module Lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.

[55] Gaëtan Leurent and Thomas Peyrin. SHA-1 is a Shambles: First Chosen-Prefix Collision on SHA-1 and Application to the PGP Web of Trust. In Srdjan Capkun and Franziska Roesner, editors, *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 1839–1856. USENIX Association, 2020.

[56] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter Lattice-Based Zero-Knowledge Proofs via One-Time Commitments. *IACR Cryptol. ePrint Arch.*, 2020:1448, 2020.

[57] Karola Marky, Oksana Kulyk, Karen Renaud, and Melanie Volkamer. What Did I Really Vote For? In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, page 176, 2018.

[58] Karola Marky, Oksana Kulyk, and Melanie Volkamer. Comparative Usability Evaluation of Cast-as-Intended Verification Approaches in Internet Voting. In *Sicherheit 2018, Beiträge der 9. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), 25.-27.4.2018, Konstanz.*, pages 197–208, 2018.

[59] Tal Moran and Moni Naor. Split-ballot voting: everlasting privacy with distributed trust. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 246–255. ACM, 2007.

[60] Maina M. Olembo, Steffen Bartsch, and Melanie Volkamer. Mental Models of Verifiability in Voting. In *E-Voting and Identify - 4th International Conference, Vote-ID 2013, Guildford, UK, July 17-19, 2013. Proceedings*, pages 142–155, 2013.

[61] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[62] Peter Y. A. Ryan, Peter B. Rønne, and Vincenzo Iovino. Selene: Voting with Transparent Verifiability and Coercion-Mitigation. In *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, pages 176–192, 2016.

[63] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. Security Analysis of the Estonian Internet Voting System. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 703–715, 2014.

[64] Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2007.

[65] Martin Strand. A Verifiable Shuffle for the GSW Cryptosystem. In *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*, pages 165–180, 2018.

[66] Scott Wolchok, Eric Wustrow, J. Alex Halderman, Hari K. Prasad, Arun Kankipati, Sai Krishna Sakhamuri, Vasavya Yagati, and Rop Gonggrijp. Security Analysis of India's Electronic Voting Machines. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 1–14, 2010.

[67] Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient Lattice-Based Zero-Knowledge Arguments with Standard Soundness: Construction and Applications. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, pages 147–175, 2019.

Appendix A. Epoque with Return Codes

In Section 3, we have presented the Epoque e-voting protocol which protects against malicious tallying authorities. In this section, we describe how Epoque can be extended so that it is also verifiable in the presence of corrupted voting devices (recall Section 2.3). More precisely, our extension enables each human voter to verify whether or not her voting device submitted a ballot that contains her actual vote. For this purpose, we extend Epoque with a return code scheme so that the resulting voting protocol is *end-to-end* verifiable.

In what follows, we will first recall the general concept of return codes in Section A.1. After that, in Section A.2, we will explain the idea of our return code scheme for Epoque. In Section A.3, we will elaborate on the properties of Epoque with return codes, including its security and its (negligible) overhead.

A.1. General concept of return codes

On a high level, a return code scheme works as follows. In addition to the existing protocol participants, we require a *return code authority* (RCA). The original voting protocol is extended as follows:

A.1.1. Generating return codes. For each human voter V_i and each possible candidate $j \in \{1, \ldots, n_{\mathsf{cand}}\}$, the return code authority generates a unique return code $rc^{i,j}$. Then, the return code authority sends the *return code sheet*

$$(rc^{i,1}, \ldots, rc^{i,n_{\mathsf{cand}}}) \tag{30}$$

to voter V_i on a channel different from the one that the voter uses to cast her ballot. Furthermore, RCA posts V_i's return code sheet in a "secret" (e.g., encrypted) form on the bulletin board so that only the trustees can jointly open it (a threshold scheme may be used here to avoid introducing single points of failure, and likewise the RCA may be distributed).

A.1.2. Reconstructing return codes. As before, voter V_i enters her chosen candidate j to her voting device. Then, the voting device generates and submits V_i's ballot to the bulletin board. Now, the trustees take as input V_i's "secret" return code sheet and the voting device's "secret" candidate j', and securely reconstruct V_i's return code

$rc^{i,j'}$ that belongs to candidate j' chosen by the voting device. The trustees post $rc^{i,j'}$ on the bulletin board.[21] The voter can now verify whether or not $rc^{i,j} = rc^{i,j'}$ holds true, i.e., whether her voting device submitted her actual candidate j.

A.1.3. Security. The main idea is that if the return code authority and the voter's voting device do not collude, then undetectably manipulating the voter's choice is as hard as correctly guessing the respective return code. Since each return code is chosen uniformly at random, vote privacy is not affected.

A.2. Idea of Epoque with return codes

We now present the idea of our return code technique. For the sake of simplicity, we focus on the case of two candidates, i.e., where each voter can either vote for "0" or "1".

In addition to the cryptographic primitives of Epoque (Section 3.2), we require a cryptographic hash function that maps tuples of the message space of the commitment scheme M_{com} to $\{0,1\}^l$. The bit length $l \geq 1$ should be chosen such that (i) human voters can successfully check with high probability whether or not two random elements from $\{0,1\}^l$ are equal, and (ii) the probability of collisions under h is bounded by some small δ.

A.2.1. Generating return codes. For each voter, the return code authority RCA chooses a blinding element β uniformly at random from the message space of the commitment scheme M_{com}. Then, the voter's return codes are $h(\beta)$ for candidate 0 and $h(\beta+1)$ for candidate 1. The voter's return code sheet is

$$\vec{rc} = (h(\beta), h(\beta+1)). \tag{31}$$

The return code authority RCA sends $\vec{rc}$ to the voter who verifies whether or not it contains any duplicates. If yes, the voter contacts the voting authority and reveals the malformed return code sheet. If not, the voter accepts the return code sheet.

Now, analogously to the ballot casting procedure of the voters, the return code authority

1) shares β among the trustees:
$$\langle \beta \rangle = (\beta_1, \ldots, \beta_{n_\mathsf{T}}), \tag{32}$$
2) commits to all of these shares, i.e., for each T_k
$$(\gamma_k, \rho_k) \leftarrow \mathsf{Com}(\mathsf{prm}_{\mathsf{com}}, \beta_k), \tag{33}$$
3) encrypts each tuple (β_k, ρ_k) using T_k's public parameters prm_k and RCA's identity, i.e., for each T_k
$$\epsilon_k \leftarrow \mathsf{Enc}(\mathsf{prm}_k, \mathsf{RCA}; (\beta_k, \rho_k)). \tag{34}$$

Eventually, the return code authority publishes $(\gamma_k, \epsilon_k)_{k=1}^{n_\mathsf{T}}$ on the bulletin board.

A.2.2. Verification of committed blinded return code sheets. Analogously to the ballot verification, each trustee T_k verifies whether ϵ_k decrypts to a valid opening of γ_k. If this is not the case, then T_k publishes the individual secret key of RCA w.r.t. its master public key mpk_k.

21. It could also be returned to the voter on any channel not controlled by the voting device.

A.2.3. Reconstruction of return code. For each trustee T_k, let (v'_k, r'_k) be the result of decrypting e_k (Section 3.2). In other words, $v' = \sum_k v'_k \in \{0,1\}$ is the choice submitted by the voting device. Now, each trustee T_k publishes the tuple

$$(\tilde{v}_k, \tilde{r}_k) \leftarrow (v'_k + \beta_k, r'_k + \rho_k) \tag{35}$$

which can be publicly verified by $\mathsf{Open}(\mathsf{prm}_{\mathsf{com}}, \tilde{v}_k, c_k + \gamma_k, \tilde{r}_k)$.

After all trustees have published their opening values, then

$$\tilde{v} \leftarrow \sum_{k=1}^{n_\mathsf{T}} \tilde{v}_k \tag{36}$$

is publicly computed and the reconstructed return code $h(\tilde{v})$ is sent to the voter, by a channel other than her voting device.

A.2.4. Extended voter verification. Observe that, due to the homomorphic and binding property of the commitment scheme,

$$\tilde{v} = \sum_{k=1}^{n_\mathsf{T}} \tilde{v}_k = \left(\sum_{k=1}^{n_\mathsf{T}} v'_k\right) + \left(\sum_{k=1}^{n_\mathsf{T}} \beta_k\right) = v' + \beta \tag{37}$$

holds true (with overwhelming probability), where $v' \in \{0,1\}$ is the vote cast by the voter's voting device. Therefore, $\tilde{v}$ equals to β if the voting device has cast a "0-vote" and to $\beta+1$ if it has cast a "1-vote".

The voter verifies whether $h(\tilde{v})$ equals to the return code associated to the candidate $v \in \{0,1\}$ that she has chosen. If this is not the case, the voter can publish a complaint.

A.3. Properties

We elaborate on the properties of the return code scheme.

A.3.1. Additional channels. In the description above, we have deliberately abstracted away from the channel that the human voter V_i uses to receive her return code sheet $\vec{rc}$ from the return code authority RCA, and from the one that she uses to read the reconstructed return code from the bulletin board B.

It is obvious that the channel from RCA to V_i is supposed to be authenticated as otherwise the adversary could choose the return code sheet on RCA's behalf. Furthermore, in order to protect V_i's vote privacy against a public observer, we also require that this channel is untappable. In practice, RCA could send $\vec{rc}$ to V_i via postal mail or a second (trusted) device.

We also require that the channel used by V_i to read $h(\tilde{v})$ from B is authenticated. To see why, assume that V_i votes for the first candidate but the corrupted voting device VSD_i submits a ballot for the second one. Now, the trustees output $\beta+1$ so that the reconstructed return code equals to $h(\beta+1)$. However, if the channel from V_i to B was not authenticated, the adversary (who knows $\beta+1$, hence β) could simply show $h(\beta)$ to V_i who would not complain even though her ballot was manipulated. In practice, one could send the reconstructed return code to

V_i via a second (trusted) device. For example, if VSD_i is V_i's personal computer, then the second device could be her mobile phone to which $h(\tilde{v})$ is sent as SMS.

A.3.2. Verifiability. As described in Section A.2, if the return code authority RCA is honest, then the reconstructed return code $h(\tilde{v})$ equals to the return code on $\vec{rc}$ for the candidate that the (possibly corrupted) voting device voted for. Hence, under the above assumptions on V_i's channels, a corrupted voting device VSD_i cannot tamper with V_i's vote without this manipulation being detected by V_i (with a certain probability).

We note that the return code scheme is not dispute-free. In fact, if a voter complains that her reconstructed return code was not correct, then it is not clear who is dishonest: the voter or her voting device? This is a general issue that, as far as we know, applies to virtually all return-code schemes in the literature. Then again, recall that the idea of return codes is not to completely remove trust from the voting devices but to mitigate it. Therefore, in case several voters (independently from each other) complain that their return code check was not successful, then this provides some evidence that something went wrong. Subsequent investigations can then help to single what caused the issue (e.g., a bug of the voters' voting software).

A.3.3. Privacy. Observe that if the return code authority is honest, then β perfectly blinds V_i's vote. Hence, vote privacy is not affected by the return code scheme.

A.3.4. Overhead. The return code scheme does not add any overhead to creating and submitting ballots. For the trustees, the overhead is minimal, too. In addition to the steps in Epoque, each trustee T_k merely has to decrypt n_V IBE ciphertexts and to do n_V plaintext additions.

Appendix B. Identity-Based Encryption (IBE)

B.1. Definition

An *identity-based encryption (IBE) scheme* is a tuple of algorithms $(\mathsf{KeyGen}_\mathsf{ibe}, \mathsf{Extr}, \mathsf{Enc}, \mathsf{Dec})$ where:

- $\mathsf{KeyGen}_\mathsf{ibe}$ is a ppt algorithm which takes 1^ℓ and outputs the public parameters $\mathsf{prm}_\mathsf{ibe} \in \{0,1\}^{\mathrm{poly}(\ell)}$ and a *master secret key* msk. The public parameters prm contain a definition of the *message space* $M_\mathsf{ibe} = M_\mathsf{ibe}^\ell$, the *ciphertext space* $C_\mathsf{ibe} = C_\mathsf{ibe}^\ell$, the *set of identities* ID, and the *master public key* mpk.
- Extr is a deterministic polynomial-time algorithm which takes $\mathsf{prm}_\mathsf{ibe}, \mathsf{msk}, \mathsf{id} \in \mathsf{ID}$ and outputs the individual secret key msk^id corresponding to identity id.
- Enc is a ppt algorithm which takes $\mathsf{prm}_\mathsf{ibe}$, $\mathsf{id} \in \mathsf{ID}$, $m \in M_\mathsf{ibe}$ and outputs a ciphertext $c \in C_\mathsf{ibe}$.
- Dec is a deterministic polynomial-time algorithm which takes $\mathsf{prm}_\mathsf{ibe}, \mathsf{msk}^\mathsf{id}, c \in C_\mathsf{ibe}$ and outputs either $m \in M_\mathsf{ibe}$ or $\perp$.

These algorithms must satisfy that for all $\mathsf{id} \in \mathsf{ID}$, $m \in M_\mathsf{ibe}$, we have $m = \mathsf{Dec}(\mathsf{prm}_\mathsf{ibe}, \mathsf{msk}^\mathsf{id}, c)$, where $c \leftarrow \mathsf{Enc}(\mathsf{prm}_\mathsf{ibe}, \mathsf{id}, m)$ and $\mathsf{msk}^\mathsf{id} \leftarrow \mathsf{Extr}(\mathsf{prm}_\mathsf{ibe}, \mathsf{msk}, \mathsf{id})$.

B.2. IND-ID-CPA security

Let $(\mathsf{KeyGen}_\mathsf{ibe}, \mathsf{Extr}, \mathsf{Enc}, \mathsf{Dec})$ be an IBE scheme. The *(IND-ID-CPA) challenger* Ch is a ppt algorithm that takes as input a bit b, public parameters $\mathsf{prm}_\mathsf{ibe}$, a master secret key msk and that serves the following types of queries:

1) For $\mathsf{id} \in \mathsf{ID}$, the challenger returns the individual secret key corresponding to id, i.e., $\mathsf{Extr}(\mathsf{prm}_\mathsf{ibe}, \mathsf{msk}, \mathsf{id})$, if the challenger has not yet returned a ciphertext for identity id (second query type).
2) For two messages $m_0, m_1 \in M_\mathsf{ibe}$ of the same size and a *challenge identity* $\mathsf{id} \in \mathsf{ID}$, the challenger returns the *challenge ciphertext* $c \leftarrow \mathsf{Enc}(\mathsf{prm}_\mathsf{ibe}, \mathsf{id}, m_b)$ if the challenger has not yet returned the individual secret key corresponding to the challenge identity id (first query type) and if this query type has not yet been called before.

Let $(\mathsf{KeyGen}_\mathsf{ibe}, \mathsf{Extr}, \mathsf{Enc}, \mathsf{Dec})$ be an IBE scheme with security parameter ℓ and let Ch be the challenger (as defined above). Then the IBE scheme is *IND-ID-CPA-secure*, if for every ppt adversary A, we have that

$$\begin{aligned} &|\mathsf{Pr}[(\mathsf{prm}_\mathsf{ibe}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\ell); \\ &\quad b' \leftarrow \mathsf{A}^{\mathsf{Ch}(1,\mathsf{prm}_\mathsf{ibe},\mathsf{msk})}(1^\ell, \mathsf{prm}_\mathsf{ibe}); b' = 1] \\ - \;&\mathsf{Pr}[(\mathsf{prm}_\mathsf{ibe}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\ell); \\ &\quad b' \leftarrow \mathsf{A}^{\mathsf{Ch}(0,\mathsf{prm}_\mathsf{ibe},\mathsf{msk})}(1^\ell, \mathsf{prm}_\mathsf{ibe}); b' = 1]| \end{aligned}$$

is a negligible function in ℓ.

Appendix C. General Computational Model

In this section, we explain the computational model for our security analysis (Section 7) in more detail.

Process. A *process* is a set of probabilistic polynomial-time interactive Turing machines (ITMs, also called *programs*) which are connected via named tapes (also called *channels*). Two programs with a channel of the same name but opposite directions (input/output) are connected by this channel. A process may have external input/output channels, those that are not connected internally. At any time of a process run, one program is active only. The active program may send a message to another program via a channel. This program then becomes active and after some computation can send a message to another program, and so on. Each process contains a *master program*, which is the first program to be activated and which is activated if the active program did not produce output (and hence, did not activate another program). If the master program is active but does not produce output, a run stops.

We write a process π as $\pi = p_1 \| \cdots \| p_l$, where $p_1, \ldots, p_l$ are programs. If π_1 and π_2 are processes, then $\pi_1 \| \pi_2$ is a process, provided that the processes are connectible: two processes are *connectible* if common external channels, i.e., channels with the same name, have opposite directions (input/output); internal channels are renamed, if necessary. A process π where all programs are given the security parameter 1^ℓ is denoted by $\pi^{(\ell)}$. In the processes we consider, the length of a run is always

polynomially bounded in ℓ. Clearly, a run is uniquely determined by the random coins used by the programs in π.

Protocol. A protocol P is modeled via a process, where different participants and components are represented via one ITM each. Typically, a protocol contains a *scheduler* S as one of its participants which acts as the master program of the protocol process (see below). The task of the scheduler is to trigger the protocol participants and the adversary in the appropriate order. For example, in the context of e-voting, the scheduler would trigger protocol participants according to the phases of an election.

The honest programs of the agents of P are typically specified in such a way that the adversary A can corrupt the programs by sending the special message $\mathsf{corrupt}$. Upon receiving such a message, the agent reveals all or some of its internal state to the adversary and from then on is controlled by the adversary. Some agents, such as the scheduler, will typically not be corruptible, i.e., they would ignore $\mathsf{corrupt}$ messages. Also, agents might only accept $\mathsf{corrupt}$ messages upon initialization, modeling static corruption. This is the case for our security analysis of Epoque.

We say that an agent a is *honest in a protocol run* r if the agent has not been corrupted in this run, i.e., has not accepted a $\mathsf{corrupt}$ message throughout the run. We say that an agent a is *honest* if for all adversarial programs π_{A} the agent is honest in all runs of $\hat{\pi}_{\mathsf{P}} \| \pi_{\mathsf{A}}$, i.e., a always ignores all $\mathsf{corrupt}$ messages.

Property. A *property* γ *of* P is a subset of the set of all runs of P.[22] By $\neg\gamma$ we denote the complement of γ.

Appendix D. Judging Procedure of Epoque

In this section, we precisely define the honest program $\hat{\pi}_{\mathsf{J}}$ of the judge J in Epoque. Recall that we assume that J is honest. We note that the honest program $\hat{\pi}_{\mathsf{J}}$ of J, as defined below, uses only publicly available information, and therefore every party, including the voters as well as external observers, can run the judging procedure.

The program $\hat{\pi}_{\mathsf{J}}$, whenever triggered by the scheduler S, reads data from the bulletin board and verifies its correctness, including correctness of posted complaints. The judge outputs verdicts (as described below) on a distinct tape. More precisely, the judge outputs verdict in the following situations:

(J1) If a party a deviates from the protocol specification in an obvious way, then J blames a individually by outputting the verdict $\mathsf{dis}(\mathsf{a})$. This is the case if the party a, for example, (i) does not publish data when expected, or (ii) publishes data which is not in the expected format, or (iii) publishes a NIZKP which is not correct, etc.

(J2) If, at the end of the ballot submission phase, the list of ballots $\vec{b}$ published by bulletin board B contains more than two ballots of the same voter, or a ballot with an invalid NIZKP, or two ballots that contain the same (partial) entries, then the judge outputs the verdict $\mathsf{dis}(\mathsf{B})$.

(J3) If, at the beginning of the tallying phase, trustee T_k publishes the correct individual secret key msk_k^i of voter V_i and e_k^i decrypts (using msk_k^i) to an *invalid* opening of c_k^i, then the judge outputs the verdict $\mathsf{dis}(\mathsf{VSD}_i)$.

(J4) If, at the beginning of the tallying phase, trustee T_k publishes either (i) an incorrect individual secret key, or (ii) a correct individual secret key msk_k^i of voter V_i and e_k^i decrypts (using msk_k^i) to a *valid* opening of c_k^i, then the judge outputs the verdict $\mathsf{dis}(\mathsf{T}_k)$.

(J5) If, during the tallying phase, trustee T_k publishes v_k^j and r_k^j such that $0 \leftarrow \mathsf{Open}(\mathsf{prm}_{\mathsf{com}}, v_k^j, c_k^j, r_k^j)$, where $c_k^j \leftarrow \sum_{i=1}^{n_{\mathsf{V}}} c_k^{i,j}$, then the judge outputs the verdict $\mathsf{dis}(\mathsf{T}_k)$.

If the judge J outputs $\mathsf{dis}(\mathsf{T}_k)$ for some T_k, then the judge outputs reject on a distinct tape, and the protocol aborts immediately. Otherwise, the judge outputs accept.

Appendix E. Accountability

In this section, we first recall the accountability framework and definition that has been introduced in [51]. Afterwards, we apply this definition to analyze accountability of Epoque.

E.1. Accountability Framework

To specify accountability in a fine-grained way, the notions of verdicts, constraints and accountability properties are used.

E.1.1. Verdicts. A verdict can be output by the judge (on a dedicated output channel) and states which parties are to be blamed (that is, which ones, according to the judge, have misbehaved). In the simplest case, a verdict can state that a specific party misbehaved (behaved dishonestly). Such an *atomic verdict* is denoted by $\mathsf{dis}(\mathsf{a})$ (or $\neg\mathsf{hon}(\mathsf{a})$). It is also useful to state more fine grained or weaker verdicts, such as "a or b misbehaved". Therefore, in the general case, we will consider verdicts which are boolean combinations of atomic verdicts.

More formally, given a run r of a protocol P (i.e., a run of some instance $\hat{\pi}_{\mathsf{P}} \| \pi_{\mathsf{A}}$ of P), we say that a verdict ψ *is true in* r, if and only if the formula ψ evaluates to true with the proposition $\mathsf{dis}(\mathsf{a})$ set to false if party a is honest in r, i.e., party a runs $\hat{\pi}_{\mathsf{a}}$ in r and has not been (statically) corrupted in r. For the following, recall that the instance $\hat{\pi}_{\mathsf{P}} \| \pi_{\mathsf{A}}$ is part of the description of r. By this, we can talk about sets of runs of different instances.

E.1.2. Accountability constraints. Who should be blamed in which situation is expressed by a set of *accountability constraints*. Intuitively, for each undesired situation, e.g., when the goal $\gamma(\varphi)$ is not met in a run of $\mathsf{P}_{\mathsf{Epoque}}$, we would like to describe who to blame.

22. Recall that the description of a run r of P contains the description of the process $\hat{\pi}_{\mathsf{P}} \| \pi_{\mathsf{A}}$ (and hence, in particular the adversary) from which r originates. Therefore, γ can be formulated independently of a specific adversary.

More formally, an accountability constraint is a tuple $(\alpha, \psi_1, \ldots, \psi_k)$, written $(\alpha \Rightarrow \psi_1 \mid \cdots \mid \psi_k)$, where α is a property of P (recall that, formally, this is a set of runs of P) and $\psi_1, \ldots, \psi_k$ are verdicts. Such a constraint *covers* a run r if $r \in \alpha$. Intuitively, in a constraint $\Gamma = (\alpha \Rightarrow \psi_1 \mid \cdots \mid \psi_k)$ the set α contains runs in which some desired goal of the protocol is *not* met (due to the misbehavior of some protocol participant). The formulas $\psi_1, \ldots, \psi_k$ are the possible (minimal) verdicts that are supposed to be stated by J in such a case; J is free to state stronger verdicts. Formally, for a run r, J *ensures* Γ *in* r, if either $r \notin \alpha$ or J states a verdict ψ in r that implies one of the verdicts $\psi_1, \ldots, \psi_k$ (in the sense of propositional logic).

E.1.3. Accountability property. A set Φ of accountability constraints for a protocol P is called an *accountability property* of P. An accountability property Φ should be defined in such a way that it covers all relevant cases in which a desired goal is not met, i.e., whenever some desired goal of P is not satisfied in a given run r due to some misbehavior of some protocol participant, then there exists a constraint in Φ which covers r. In particular, in this case the judge is required to state a verdict.

E.1.4. Notation. Let P be a protocol with the set of agents Σ and an accountability property Φ of P. Let π be an instance of P and $\mathsf{J} \in \Sigma$ be an agent of P. We write $\Pr[\pi^{(\ell)} \mapsto \neg(\mathsf{J}\colon \Phi)]$ to denote the probability that π, with security parameter 1^ℓ, produces a run such that J does not ensure Γ in this run for some $\Gamma \in \Phi$.

Definition 4 (Accountability[23]). Let P be a protocol with the set of agents Σ. Let $\mathsf{J} \in \Sigma$ be the judge, and Φ be an accountability property of P. Then, *the protocol* P *is* Φ*-accountable w.r.t. the judge* J if for all adversaries π_A and $\pi = (\hat{\pi}_\mathsf{P} \| \pi_\mathsf{A})$, the probability $\Pr[\pi^{(\ell)} \mapsto \neg(\mathsf{J}\colon \Phi)]$ is negligible as a function of ℓ.

E.1.5. Individual accountability. In practice, so-called *individual accountability* is highly desirable in order to deter parties from misbehaving. Formally, $(\alpha \Rightarrow \psi_1 \mid \cdots \mid \psi_k)$ provides *individual accountability* if for every $i \in \{1, \ldots, k\}$ there exists a party a such that ψ_i implies $\mathsf{dis}(\mathsf{a})$. In other words, each $\psi_1, \ldots, \psi_k$ determines at least one misbehaving party.

E.2. Accountability of Epoque

We are now able to precisely analyze the accountability level provided by Epoque. For this, we first define the accountability constraints and property of Epoque. Then, we state and prove the accountability theorem.

E.2.1. Accountability constraints. The accountability theorem for Epoque (see below) states that if the adversary breaks the goal $\gamma(\varphi)$ in a run of $\mathsf{P}_{\mathsf{Epoque}}$, then (at least) one misbehaving trustee can be blamed individually (with a certain probability). The accountability constraint for this situation is $\neg\gamma(\varphi) \Rightarrow \mathsf{dis}(\mathsf{T}_1) | \ldots | \mathsf{dis}(\mathsf{T}_{n_\mathsf{T}})$. Now, the judge J ensures this constraint in a run r if $r \notin \neg\gamma(\varphi)$ or the verdict output by J in r implies $\mathsf{dis}(\mathsf{T}_k)$ for some T_k.

E.2.2. Accountability property. For $\mathsf{P}_{\mathsf{Epoque}}$ and the goal $\gamma(\varphi)$, we define the accountability property Φ to consist of the constraint mentioned above. Clearly, this accountability property covers $\neg\gamma(\varphi)$ by construction, i.e., if $\gamma(\varphi)$ is not satisfied, these constraints require the judge J to blame some trustee. Note that in the runs covered by this constraint all verdicts are atomic. This means that Φ requires that, whenever the goal $\gamma(\varphi)$ is violated, an individual trustee T_k is blamed (*individual accountability*).

For the accountability theorem, we make the same assumptions (V1) to (V2) as for the verifiability theorem (see Section 7.2). Now, the following theorem states the accountability result of Epoque.

Theorem 3 (Accountability). Under the assumptions (V1) to (V2) (Section 7.2) and the mentioned judging procedure run by the judge J, $\mathsf{P}_{\mathsf{Epoque}}(n_\mathsf{V}, n_\mathsf{T}, n_{\mathsf{cand}}, \mu)$ is Φ-accountable w.r.t. the judge J.

Recall that, following [51], this accountability theorem implies the verifiability theorem (Theorem 1). The proof is provided in [14].

23. Similarly to the verifiability definition, we also require that the judge J is *computationally fair* in P, i.e., for all instances π of P, the judge J states false verdicts only with negligible probability. For brevity of presentation, this is omitted here (see [51] for details). This condition is typically easy to check. In particular, it is easy to check that the judging procedure for Epoque does not blame honest parties.