Verifiable E-Voting with a Trustless Bulletin Board

Daniel Rausch
Institute of Information Security
University of Stuttgart
Stuttgart, Germany
daniel.rausch@sec.uni-stuttgart.de

Nicolas Huber
Institute of Information Security
University of Stuttgart
Stuttgart, Germany
nicolas.huber@sec.uni-stuttgart.de

Ralf Küsters
Institute of Information Security
University of Stuttgart
Stuttgart, Germany
ralf.kuesters@sec.uni-stuttgart.de

***Abstract*—Voter privacy and end-to-end (E2E) verifiability are critical features of electronic voting (e-voting) systems to safeguard elections. To achieve these properties commonly a perfect bulletin board (BB) is assumed that provides consistent, reliable, and tamper-proof storage and transmission of voting data. However, in practice, BBs operate in asynchronous and unreliable networks, and hence, are susceptible to vulnerabilities such as equivocation attacks and dropped votes, which can compromise both verifiability and privacy. Although prior research has weakened the perfect BB assumption, it still depends on trusting certain BB components.**

In this work, we present and initiate a formal exploration of designing e-voting systems based on fully untrusted BBs. For this purpose, we leverage the notion of accountability and in particular use accountable BBs. Accountability ensures that if a security breach occurs, then cryptographic evidence can identify malicious parties. Fully untrusted BBs running in asynchronous networks bring new challenges. Among others, we identify several types of attacks that a malicious but accountable BB might be able to perform and propose a new E2E verifiability notion for this setting. Based on this notion and as a proof of concept, we construct the first e-voting system that is provably E2E verifiable and provides vote privacy even when the underlying BB is fully malicious. This establishes an alternative to traditional e-voting architectures that rely on (threshold) trusted BB servers.

***Index Terms*—E-Voting, Accountability, Bulletin Board**

I. Introduction

As democratic societies increasingly adopt electronic voting (e-voting) systems to enhance the accessibility, convenience, and efficiency of elections, the challenge of ensuring vote privacy and end-to-end (E2E) verifiability has gained significant research attention. Vote privacy is essential for maintaining the confidentiality of individual votes, while E2E verifiability allows everyone, including voters, candidates, and even external observers, to confirm that the election result reflects the intended votes of all voters. Both properties are central to protecting the integrity of elections, against accidental programming errors as well as malicious actors.

A key goal of e-voting research is minimizing the trust assumptions needed for security [1]–[7]. This is especially crucial for verifiability which should, ideally, allow a voter to check whether the result is correct and takes their own vote into account even in the worst case where all parties running the election act maliciously and collude with each other. One common trust assumption that has garnered interest concerns the bulletin board (BB) component.

E-Voting with BBs. A BB is supposed to act as a shared broadcast channel with memory [8]. It consists of one or more servers that maintain the internal memory/state, i.e., a list of client inputs that have been added successfully to the BB. A secure implementation of a BB is typically expected to provide at least *consistency (of outputs)* [9], [10]. That is, the internal memory list is append-only and clients reading from the BB always obtain prefixes of that list.

Most e-voting systems rely on a BB to share and publish data among participants and external observers, including election parameters, encrypted votes, the election result, and verification data such as zero-knowledge proofs (ZKPs). Security analyses of e-voting systems, such as [11]–[18], commonly assume that the underlying BB is not just trusted in the sense that it always provides consistent outputs from an append-only immutable state. They rather use the much stronger assumption of the BB being *perfect* [5], [10], at least in some parts/aspects, such that additionally *(i)* all outputs read by clients are not just consistent but even identical, *(ii)* all inputs will always end up in the internal memory, i.e., inputs cannot be dropped, *(iii)* new inputs are immediately added to the internal memory and then readable by everyone without delay, and/or *(iv)* items in the BB are ordered chronologically by when clients submitted them. Such BBs that are not just trusted but in some parts or even fully perfect cannot exist in reality. In an actual implementation of a BB running in an asynchronous unreliable network, called a *real BB* in what follows, inputs might get lost during transmission and never end up in the internal memory, inputs take time to become available as outputs and might change their order in the process, and outputs received at the same time by clients might not be identical but only consistent as they might have been generated and sent by servers at different points in time. Malicious servers in a real BB can even *equivocate outputs* [5], that is, provide different clients with inconsistent outputs that are not prefixes of the same internal memory list. Such inconsistent outputs can contradict each other, e.g., by showing different sets of votes that were tallied in an election.

This gap is not just a theoretical concern. The assumption of a perfect BB rather precludes major attack classes available to real BBs [5], [7], [19], [20]. For example, Hirschi et al. [5] observe that a malicious real BB might be able to use equivocation to break verifiability, say, by dropping some votes while presenting each affected voter with a modified

view of the election that still includes their own vote. Voters performing verification might then mistakenly be convinced that the result took their vote into account when it was actually dropped. Hirschi et al. also show that several prominent e-voting systems, namely Helios [21], Belenios [22], and Civitas [23], are susceptible to such equivocation attacks on verifiability. In [20], Cortier and Lallemand observe that privacy can be compromised as well when real BBs are used. For example, the BB might selectively drop all but one vote of interest, which is then tallied and revealed as the result of the election. Notably, this issue can occur even when all BB servers are trusted as a network adversary can drop inputs sent by clients before even reaching servers.

Current State. Several works on provably secure e-voting address the trust assumption of perfect BBs. They can roughly be classified into two major approaches.

One approach aims to perform security analyses of e-voting systems directly based on real BBs rather than using (partly) perfect BBs as an abstraction, e.g., [4], [5], [9], [24]. This approach assumes a trusted threshold of servers – without a single trusted server, equivocation is trivial – which is much more realistic than a perfect BB. On a conceptual level, the work by Hirschi et al. [5] is particularly noteworthy. Among others, they identify fundamental classes of attacks on verifiability and propose new general verifiability notions for analyzing voting systems following this approach.

An alternative approach used by works such as [6], [7], [19] is to retain a (partly) perfect BB but to give the adversary additional capabilities to more closely resemble some of the capabilities of real BBs in asynchronous networks, e.g., by allowing an adversary to selectively drop or reorder inputs to the perfect BB or by giving the adversary full control over the BB but only during some phases of the election. Conceptually, the work by Cortier et al. [7] is one of the most important ones as they propose several new privacy notions for this approach.

Both approaches have in common that they cannot entirely eliminate the trust assumption on the BB, which is particularly troublesome for verifiability. In this work, we, therefore, *propose and explore a third approach* based on the notion of accountability, which, as we will formally prove for the first time, can be used to construct *secure e-voting systems based on entirely untrusted, hence possibly fully malicious, real BBs*.

Accountability. The works constructing or using real BBs mentioned above aim for so-called *preventive security* [25]. That is, it should be impossible to break a security property/goal of a BB, such as consistency, even if some servers running the BB are actively misbehaving. While very desirable, such preventive security guarantees generally necessitate trusting at least a threshold of servers.

In contrast to preventive security, the concept of *(individual) accountability* [25]–[28] intuitively ensures the following: if some intended security property/goal of a protocol, e.g., consistency in a BB, is violated, then one can obtain undeniable cryptographic evidence that identifies at least one misbehaving protocol participant that has deviated from the protocol. This not only ensures that a security breach will be noticed - thus offering a type of verifiability [26] - but further enables holding parties accountable for misbehavior, e.g., via financial or contractual penalties, which strongly incentivizes malicious parties to follow the protocol honestly and not break security properties in the first place. As accountability does not preclude a breach of a security property – it only ensures that this will be observable and a culprit can be identified after the fact – it is possible to achieve accountability-based security even in a fully malicious setting without trusted parties (typically still requiring some cryptographic assumptions, e.g, a PKI). We note that accountability-based and preventive security are orthogonal concepts with different (dis-)advantages that can be used independently to protect a protocol. However, they can also be combined using accountability as a second layer of defense in case trust assumptions underlying preventive security are not met (see [27] for more details).

Accountability is widely used in many security fields, including auctions [29]–[31], secure multi-party computation (MPC) [32]–[37], public key infrastructures (PKIs) [38]–[40], and distributed ledgers [41]–[49]. Accountability has also been used in the area of e-voting [6], [21], [22], [26], [50], [51]. However, works that formally prove accountability of an e-voting system still assume an at least partly perfect BB. While accountability is often mentioned as desirable and sometimes also claimed for real BBs [9], [24], [52], only recently did Graf et al. establish the first provably secure BB called $\mathsf{Fabric}_{\mathsf{BB}}$ [10] which offers accountability w.r.t. consistency. That is, as long as there is no evidence of misbehavior, all clients will always receive consistent outputs. Notably, $\mathsf{Fabric}_{\mathsf{BB}}$ achieves this property without assuming any trusted servers, i.e., in a fully malicious setting. The security result is shown in a Universal Composability (UC) model. All guarantees thus hold even when $\mathsf{Fabric}_{\mathsf{BB}}$ is used as a subroutine within a higher-level protocol.

This work. As mentioned, our goal is to obtain provably secure E2E verifiable and private e-voting systems without trusted servers running the BB. The underlying idea is conceptually simple: Use a BB that provides accountability w.r.t. consistency even when all parties are malicious. The election verification procedure should then check whether any evidence of misbehavior in the BB was obtained and, if so, reject the election result. If no misbehavior is detected and, hence, consistency holds, verification proceeds to check the correctness of the election material stored on the BB.

While this approach has not been formally considered so far and no security notions or proofs for this setting exist, the idea itself has already been mentioned and was sometimes even used informally. For example, both Helios [21] and Belenios [22], which have been formally proven secure based on (partly) perfect BBs [6], [11], [12], observe that an actual BB implementation might misbehave and require that its outputs should thus be monitored/audited to detect such misbehavior. This is a type of accountability-based security. Considering such auditing only informally outside of a security analysis,

however, is not sufficient as illustrated by Hirschi et al. [5], who found several attacks on verifiability of Helios and Belenios that a malicious BB can perform without being detected by the auditing procedures suggested by those systems.

Investigating this approach more formally raises several questions that we address in this work such as: How do we define verifiability and privacy in a setting with a potentially fully malicious BB? Can we adapt or re-use previous notions for perfect and threshold-trusted BBs, at least to some extent? Is obtaining a provably secure e-voting system using accountable BBs even feasible? Are there inherent issues and limitations of this approach that protocol designers have to handle?

Contributions. We start the first formal investigation of the above idea and establish fundamental insights for this third approach. Our work is thus best seen as a counterpart to the foundational works of Hirschi et al. [5] and Cortier et al. [7], who established security notions and general results for the other two existing approaches. More specifically, our contributions are as follows:

E2E Verifiability: We observe that existing verifiability notions that assume perfect BBs, such as the ones studied in [18], and the new verifiability notions for threshold-trusted real BBs by Hirschi et al. [5] cannot be applied to voting systems with untrusted but accountable BBs. We, therefore, propose a novel E2E verifiability notion that is highly general: it applies not just to e-voting systems with accountable BBs but also, e.g., to those with perfect or threshold trusted real BBs.

Insights Into Individual and Universal Verifiability: Instead of E2E verifiability ("Does the election result correspond to the intended votes of all honest voters and at most one vote per dishonest voter?"), the literature often considers the combination of *individual verifiability* ("Can a voter check that their encrypted ballot is included in the set that was tallied?") and *universal verifiability* ("Was the election result derived correctly from the set of encrypted ballots?"). We describe two novel attack classes on E2E verifiability that a malicious BB, depending on the voting system, might be able to perform without breaking individual or universal verifiability. Together with two related attack classes due to malicious BBs previously observed in [5], this shows that an E2E verifiability notion is necessary for analyzing the security of e-voting systems using real BBs; checking individual and universal verifiability is insufficient. This result supplements [6], [18] who observed that even for systems assuming a perfect BB splitting E2E verifiability into subproperties can be problematic.

Privacy: Existing privacy notions often hard-code a perfect BB into their definition, e.g., [11], [14]–[16], [53]. This includes the definitions proposed by Cortier et al. [7], which still assume a partly perfect BB that cannot equivocate. Some privacy notions are defined more generally, e.g., the one by Küsters et al. [54] is defined abstractly for arbitrary voting systems but so far has only been applied to systems where the BB is assumed to be at least partly perfect [50], [54]–[58]. We show that and how the definition by Küsters et al. can also be applied to analyze the privacy of e-voting systems using a potentially malicious real BB. A major challenge is that untrusted BBs can always drop all but one vote, which will then be tallied and published, thus trivially breaking privacy of that vote [20], [55]. We address this impossibility by considering a class of risk-avoiding adversaries.

A Simple Proof-Of-Concept Voting System: As a case study, we construct a concrete e-voting system based on an arbitrary trustless BB that is accountable w.r.t. consistency. This system, called ABBOVE (Accountable Bulletin Board-based Voting) follows the line of homomorphic aggregation-based systems such as Helios and Belenios [21], [22] and is purposefully kept simple to focus on identifying potential issues, limitations, and requirements inherent to this new approach. It should be seen as a proof of concept that includes only a minimal set of features and does not offer advanced mechanisms such as receipt-freeness [59], [60] or coercion-resistance [54], [60].

One of the main challenges in designing a secure system in this approach turns out to be asynchronous networks, including delays and message drops, that do not exist in perfect BBs. For example, even when all BB servers behave honestly, consistent outputs might still differ arbitrarily in length. A voting system, therefore, needs to provide stability under message extension to prevent attacks where, say, the election result could be changed by returning a longer or shorter sequence of messages. This property requires care due to new aspects not present or left implicit in settings with perfect BBs. Another example is the lack of reliability, i.e., BB inputs are not guaranteed to become part of the output. Among others, this rules out some standard constructions used in e-voting systems, such as the distributed threshold ElGamal key generation protocol proposed in [16], which requires a reliable broadcast channel to ensure secrecy of the key.

Feasibility Result: We show that ABBOVE achieves both privacy and our new E2E verifiability notion, thus formally confirming for the first time that the long-standing idea of using accountability to protect against malicious untrusted BBs is indeed sound and applicable to the class of homomorphic aggregation-based e-voting systems. We show this result not for a specific BB but more generally for any BB accountable w.r.t. consistency in a UC model. As an immediate corollary, we obtain that this holds for $\mathsf{Fabric}_{\mathsf{BB}}$ in particular. By this, we also establish the first concrete provably E2E verifiable and private e-voting system without assuming any trusted BB servers running in an asynchronous network.

In our verifiability proof, we bridge the gap between a game-based verifiability notion (defined as a trace property over protocol runs) and the UC security notion of the BB, which establishes computational indistinguishability of two systems but, in general, does not imply that arbitrary trace properties carry over from one system to the other (for interested readers we provide simple counterexamples in the full version [61]). This uncommon combination of different styles of security definitions might be useful also for other works on e-voting.

Structure. We recall the definition of UC-secure accountable bulletin boards in Section II. Since the paper is easier to

follow after seeing a concrete example voting system, we start by describing and discussing ABBOVE in Section III. In Sections IV and V, we define a new notion for and prove E2E verifiability. In Section VI, we study and prove privacy. Additional details are available in the full version [61].

II. Recap: UC-Secure Accountable BBs

Universal composability (UC) is a simulation-based paradigm for modeling and proving the security of protocols (e.g., [62]–[64]). In a UC security analysis, one first defines an *ideal protocol/ideal functionality* $\mathcal{F}$ that specifies a protocol's intended security and functional properties, i.e., $\mathcal{F}$ is secure by definition but typically cannot be run in reality. To then analyze the security of an actual protocol $\mathcal{P}$, the *real protocol*, one shows that there exists a probabilistic polynomial-time (ppt) ideal adversary/simulator $\mathcal{S}$ controlling the network of $\mathcal{F}$ such that no ppt distinguisher $\mathcal{E}$, called *environment*, can tell whether it is running in the *real world* with $\mathcal{P}$ or in the *ideal world* with $\mathcal{F}$ and $\mathcal{S}$ (written $\mathcal{F} \mid \mathcal{S}$). Since $\mathcal{F}$ is secure by definition, the real protocol $\mathcal{P}$ must, therefore, be at least as good. One also says that $\mathcal{P}$ is a secure implementation of or (UC-)realizes $\mathcal{F}$, written $\mathcal{P} \leq \mathcal{F}$.

UC models provide a so-called composition theorem which states that, if $\mathcal{P} \leq \mathcal{F}$, then a ppt higher-level protocol $\mathcal{Q}$ using $\mathcal{P}$ as a subroutine (written $\mathcal{Q} \mid \mathcal{P}$) realizes $\mathcal{Q}$ using $\mathcal{F}$ (i.e., $\mathcal{Q} \mid \mathcal{P} \leq \mathcal{Q} \mid \mathcal{F}$). That is, one can first analyze $\mathcal{Q}$ w.r.t. the simpler $\mathcal{F}$, typically showing that $\mathcal{Q} \mid \mathcal{F}$ realizes some other ideal functionality $\mathcal{F}'$, and then replace $\mathcal{F}$ by $\mathcal{P}$ without further proof.

An Ideal Functionality for Accountable BBs. To securely access data stored in a real BB run by servers, higher-level protocols generally have to run a specific BB client software as a subroutine. For example, clients might have to use server public keys to verify signatures on outputs before returning them to higher-level protocols, might have to query multiple servers and combine a threshold of responses to obtain an overall output, or might have to sign inputs from higher-level protocols due to an access control mechanism. In [10], Graf et al. therefore defined an ideal bulletin board functionality $\mathcal{F}_{\mathsf{BB}}$ that abstractly captures not just BB servers but also clients while formalizing accountability w.r.t. consistency for outputs returned by clients to higher-level protocols.

Intuitively, $\mathcal{F}_{\mathsf{BB}}$ contains several BB clients that offer an input/output interface for the environment/higher-level protocols such as e-voting systems. The environment can send an input $(\texttt{write}, m)$ to a client to request that it tries to add the message m to the BB. The environment can also send an input `read` to a client to request that it tries to read the current state of the BB and return the result as an output. Whenever a client in $\mathcal{F}_{\mathsf{BB}}$ receives such input, $\mathcal{F}_{\mathsf{BB}}$ forwards it to the network adversary, who can then decide whether and how to proceed. Notably, an adversary can decide to drop or delay the result of an input, and they can also choose the output for successful read requests subject to some restrictions explained below. Internally, $\mathcal{F}_{\mathsf{BB}}$ keeps an ordered append-only sequence of messages $\mathsf{msglist}$. The adversary can extend $\mathsf{msglist}$ in arbitrary ways at any time, e.g., by adding a message m that was part of an input $(\texttt{write}, m)$ received by a client or by adding new messages of their choosing. Finally, the adversary can corrupt clients and servers at any time to gain full control over them.

To define accountability of security properties, $\mathcal{F}_{\mathsf{BB}}$ includes a so-called judge $\mathsf{J}_{\mathsf{BB}}^{\mathsf{acc}}$. Unlike clients, $\mathsf{J}_{\mathsf{BB}}^{\mathsf{acc}}$ does not model a dedicated party but rather abstractly captures an auditing algorithm; this is a common modeling technique for formalizing accountability and verifiability [18], [26], [27]. In a realization $\mathcal{P}_{\mathsf{BB}}$ of $\mathcal{F}_{\mathsf{BB}}$, the implementation of $\mathsf{J}_{\mathsf{BB}}^{\mathsf{acc}}$ specifies the exact data and algorithm used to detect misbehavior, possibly resulting in a *verdict* that identifies a malicious party that can be held accountable. Here, we consider only realizations of $\mathsf{J}_{\mathsf{BB}}^{\mathsf{acc}}$ that run on publicly available data such that any party can run the judging algorithm in practice. Hence, such a judge represents arbitrary parties that try to check whether consistency holds; such parties can be BB clients or servers, parties from a higher-level protocol, or outside observers.

Accountability w.r.t. consistency is then formalized by $\mathcal{F}_{\mathsf{BB}}$ as follows. As long as the judge $\mathsf{J}_{\mathsf{BB}}^{\mathsf{acc}}$ in $\mathcal{F}_{\mathsf{BB}}$ has not yet computed a verdict, then whenever the adversary instructs an honest client to return an output r to a read request, r is a prefix of $\mathsf{msglist}$. Conversely, once $\mathsf{J}_{\mathsf{BB}}^{\mathsf{acc}}$ has identified at least one malicious party in a verdict, the adversary can return arbitrary, possibly contradictory, outputs r. The environment/higher-level protocols can query $\mathsf{J}_{\mathsf{BB}}^{\mathsf{acc}}$ to obtain the current verdict, if any. This captures that a party (possibly from the higher-level protocol) can, at any point in time, decide to run the judging algorithm of a realization of $\mathcal{P}_{\mathsf{BB}}$ on currently available public data to detect misbehavior. If there is no verdict yet, then consistency holds.

Observe that from this definition of $\mathcal{F}_{\mathsf{BB}}$, it follows that any realization $\mathcal{P}_{\mathsf{BB}}$ provides accountability w.r.t. consistency: Assume that the implementation of the judging algorithm $\mathsf{J}_{\mathsf{BB}}^{\mathsf{acc}}$ in $\mathcal{P}_{\mathsf{BB}}$ does not detect misbehavior and hence does not return a verdict. Then, by indistinguishability, the ideal judge in $\mathcal{F}_{\mathsf{BB}}$ also does not return a verdict. In this case $\mathcal{F}_{\mathsf{BB}}$ guarantees that the outputs of honest clients are consistent. Again, by indistinguishability, all outputs of honest clients in the realization $\mathcal{P}_{\mathsf{BB}}$ are thus also consistent.

UC-Secure Accountable BBs Exist. This work is built on, and all our results are shown for an arbitrary real BB $\mathcal{P}_{\mathsf{BB}}$ that realizes $\mathcal{F}_{\mathsf{BB}}$. Recently, Graf et al. [10] constructed the first such BB called $\mathsf{Fabric}_{\mathsf{BB}}$. This BB is a slight extension of the prominent Hyperledger Fabric protocol [65] with an efficient implementation. Graf et al. showed:

Theorem 1 (Security of $\mathsf{Fabric}_{\mathsf{BB}}$, informal)**.** *It holds true that* $\mathsf{Fabric}_{\mathsf{BB}} \leq \mathcal{F}_{\mathsf{BB}}$. *This holds in an asynchronous network, even if all BB servers and arbitrary clients are malicious.*

We note that the technical details of $\mathsf{Fabric}_{\mathsf{BB}}$ are irrelevant to this work since all results are shown using only that $\mathsf{Fabric}_{\mathsf{BB}} \leq \mathcal{F}_{\mathsf{BB}}$. For the details of $\mathsf{Fabric}_{\mathsf{BB}}$, refer to [10].

III. A Simple E-Voting System based on a Trustless BB

In Section III-A, we describe our proof of concept system ABBOVE, which follows a standard homomorphic aggregation-based design in the style of Helios, Belenios, and similar systems [21], [22], [50] but uses an accountable trustless BB. In Section III-B, we highlight and discuss important properties necessary for building securely on such a BB.

A. System Description

Building blocks. ABBOVE uses the following standard cryptographic primitives: Full threshold exponential ElGamal encryption $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ where n_{talliers} parties each hold a share of the private key. An EUF-CMA-secure *signature scheme* $\mathcal{S}$ for authentication. A *non-interactive zero-knowledge proof* (NIZKP) $\pi^{\mathsf{KeyShareGen}}$ for proving knowledge and correctness of a private decryption key share; a NIZKP π^{Enc} for proving knowledge and correctness of a plaintext vector contained in a ciphertext vector; a NIZKP π^{DecShare} for proving that a private decryption key share was correctly applied to a given ciphertext (see, e.g., [11] for possible NIZKPs. We recall formal security definitions of the required cryptographic primitives in the full version [61]). We also assume a secure PKI.

We use an arbitrary (possibly trustless) real BB protocol $\mathcal{P}_{\mathsf{BB}}$ that realizes $\mathcal{F}_{\mathsf{BB}}$. Note that while $\mathcal{P}_{\mathsf{BB}} \leq \mathcal{F}_{\mathsf{BB}}$ implies that as long as no misbehavior is detected all outputs remain consistent, such a $\mathcal{P}_{\mathsf{BB}}$ does not necessarily prevent an adversary from, e.g., dropping, reordering, or delaying inputs arbitrarily. It might also happen that outputs returned to different parties, while consistent, differ in length. All of this can simply be due to an asynchronous, unreliable real-world network. Dealing with these potential issues is the main challenge in designing a voting system on top of such a BB (see also the discussion in Section III-B).

Protocol participants. ABBOVE is run among an election authority Auth, voters $\mathsf{V}_1, \ldots, \mathsf{V}_{n_{\mathsf{voters}}}$,[1] and talliers $\mathsf{T}_1, \ldots, \mathsf{T}_{n_{\mathsf{talliers}}}$. All parties also act as clients in $\mathcal{P}_{\mathsf{BB}}$ by running the BB client code as a subroutine. We say "publish a message m on $\mathcal{P}_{\mathsf{BB}}$" to mean that a party runs the client code to submit m as input, which then may or may not end up in the BB state; analogously for "read/obtain some data from $\mathcal{P}_{\mathsf{BB}}$". While we treat $\mathcal{P}_{\mathsf{BB}}$ as a black box, further internal parties/servers are typically running the BB.

The election authority is responsible for determining trusted setup parameters but is not involved in evaluating the election; it is thus, besides the PKI, the only trusted entity. The talliers will tally the voters' ballots. In order to avoid that a small set of malicious talliers learns how each voter voted, we distribute the secret decryption key among them so that all n_{talliers} talliers must collaborate to decrypt ballots.

The election authority, all voters, and all talliers own a signing key pair for $\mathcal{S}$ to authenticate their messages. To publish complaints when voters detect that their vote was dropped, we require that voters can use a separate reliable channel, i.e., a public channel without message loss; all other communication is via unreliable networks/the unreliable $\mathcal{P}_{\mathsf{BB}}$ (see the discussion in Section III-B).

Overview. There are four protocol phases: *Setup*, *voting*, *tallying*, and *public result verification*. During tallying, votes are aggregated homomorphically before decryption to protect the privacy of individual votes. Voters can run a *vote verification* algorithm to check whether their vote was counted or dropped.

Setup phase. The election authority Auth determines the parameters of the current election, including i) the ElGamal group $\mathbb{G}$ of size p with generator g, ii) the list of eligible voters $\mathbf{id}_{\mathsf{voters}}$ where each voter V_i is identified by a unique ID idv_i and their public signing key $\mathsf{pk}^{\mathcal{S}}_{\mathsf{V}_i}$, iii) the list of talliers identified via an ID idt_i and their public signing key $\mathsf{pk}^{\mathcal{S}}_{\mathsf{T}_i}$, and iv) the set $\mathsf{C} \subseteq \{0, \ldots, n_{\mathsf{vpc}}\}^{n_{\mathsf{cand}}}$ of valid non-abstention votes,[2] called *choice space*, where n_{cand} denotes the number of candidates and n_{vpc} is an upper bound for the number of votes that a voter may assign to each candidate. For example, in an election with three candidates where voters can assign a single vote to one candidate, we set $\mathsf{C} = \left\{(c_1, c_2, c_3) \mid c_i \in \{0,1\}, \sum_{i=1}^{3} c_i = 1\right\}$. Auth signs and then publishes the parameters on $\mathcal{P}_{\mathsf{BB}}$.

All talliers T_i locally choose a private ElGamal encryption key share $\mathsf{sk}^{\mathcal{E}}_i \leftarrow^{\$} \mathbb{Z}_p$ uniformly at random, compute a corresponding public key share $\mathsf{pk}^{\mathcal{E}}_i = g^{\mathsf{sk}^{\mathcal{E}}_i}$, and create a NIZKP $\pi^{\mathsf{KeyShareGen}}_i$ to prove knowledge of $\mathsf{sk}^{\mathcal{E}}_i$ corresponding to $\mathsf{pk}^{\mathcal{E}}_i$. Each tallier T_i then signs and publishes the message $(\mathsf{idt}_i, \mathsf{pk}^{\mathcal{E}}_i, \pi^{\mathsf{KeyShareGen}}_i)$ on $\mathcal{P}_{\mathsf{BB}}$.

A protocol participant considers the setup finished if, after reading from $\mathcal{P}_{\mathsf{BB}}$, they see the election parameters and one key share with valid signature per tallier (if there are multiple messages by the same signer, then all but the first are ignored). Note that this view can differ for each reader since $\mathcal{P}_{\mathsf{BB}}$ might not (yet) have delivered some of the setup messages. After the setup is finished, the protocol participant can check the ZKPs and, if they are valid, compute the overall public encryption key $\mathsf{pk}^{\mathcal{E}} = \prod_{i \in \{1,\ldots,n_{\mathsf{talliers}}\}} \mathsf{pk}^{\mathcal{E}}_i$ of the election. In that case this protocol participant considers the system ready for voting. Otherwise, $\mathsf{pk}^{\mathcal{E}}$ is undefined, and voting is not possible.

Voting phase. A voter V_i can abstain or cast a well-formed vote $\mathbf{v} \in \mathsf{C}$. In the latter case, they wait until they can obtain $\mathsf{pk}^{\mathcal{E}}$ from their output of $\mathcal{P}_{\mathsf{BB}}$ and then encrypt each entry $\mathbf{v}[j]$ by computing an ElGamal ciphertext $\mathbf{ct}_i[j] := (g^r, g^{\mathbf{v}[j]} \cdot (\mathsf{pk}^{\mathcal{E}})^r)$ with $r \leftarrow^{\$} \mathbb{Z}_p$. This results in a ciphertext vector $\mathbf{ct}_i$, where each entry encrypts the number of votes assigned by V_i to candidate j. Voter V_i also computes a NIZKP π^{Enc}_i to prove that they know which vote $\mathbf{v}$ the vector $\mathbf{ct}_i$ encrypts and that $\mathbf{v} \in \mathsf{C}$. Finally, V_i signs their ballot $\mathsf{b}_i = (\mathsf{idv}_i, \mathbf{ct}_i, \pi^{\mathsf{Enc}}_i)$ and publishes the signed ballot on $\mathcal{P}_{\mathsf{BB}}$. They also store the signed ballot locally for verification.

[1] Since ABBOVE is a proof-of-concept system to establish viability, we simplify by not distinguishing between voters and their voting devices.

[2] We use a special vote $\texttt{abstain} \notin C$ to describe abstention. $\widetilde{\mathsf{C}} := \mathsf{C} \cup \{\texttt{abstain}\}$ denotes the set of all valid votes, including abstention.

If a voter never sees the setup phase finish, they sign a complaint $(\mathsf{idv}_i, \texttt{UnableToVote})$, indicating that their vote was dropped, and publish their complaint via their reliable channel for complaints.

The election authority Auth determines when voting closes and then publishes a signed message `VotingClosed` on $\mathcal{P}_{\mathsf{BB}}$. Just as for the setup phase, the view on whether the voting phase has finished might differ for each reader of $\mathcal{P}_{\mathsf{BB}}$.

Vote verification. A voter V_i who submitted a signed ballot b_i can verify that their vote $\mathbf{v}$ will be counted by reading the current list of ballots from $\mathcal{P}_{\mathsf{BB}}$ and checking that (i) b_i (including the signature) appears in the output and (ii) there is no valid signed `VotingClosed` message before b_i. If this fails (and the voter no longer wants to wait for any possibly delayed messages from $\mathcal{P}_{\mathsf{BB}}$), they sign a complaint $(\mathsf{idv}_i, \texttt{VoteVerifFailed})$ and publish it via their reliable channel for filing complaints.

Vote verification can be performed at any point in time, also immediately after submitting a vote, which conforms to the idea of "Vote&Go" [5], [66]. This improves the probability p_{verif} that a voter performs vote verification which is needed to detect dropped votes (cf. Section III-B).

Tallying phase. Once a tallier T_i reading from $\mathcal{P}_{\mathsf{BB}}$ sees that the voting phase has finished, T_i starts tallying as follows.

1. Homomorphic aggregation. Let $\mathbf{b}$ be the ordered list of all ballots for the current election contained in T_i's output of $\mathcal{P}_{\mathsf{BB}}$ after the election setup has finished with a well-defined $\mathsf{pk}^{\mathcal{E}}$ and before the first signed `VotingClosed` message. T_i then removes ballots (i) that do not contain the ID idv_j of an eligible voter, (ii) where the signature is invalid under the public key $\mathsf{pk}_j^{\mathcal{S}}$ corresponding to the ID idv_j in the ballot, or (iii) where the NIZKP is not valid. Then, T_i removes all but the first ballot for any given idv_j such that, for each voter, at most one vote is counted. Finally, T_i performs ballot weeding, i.e., removes all ballots from $\mathbf{b}$ that are duplicates w.r.t. the ciphertext vector $\mathbf{ct}$ only keeping the first one. Ballot weeding protects against *replay attacks* which jeopardize vote privacy [67], [68]. From the resulting set of valid ballots $\mathbf{b}_{\mathsf{valid}}$, if it is non-empty, T_i then homomorphically aggregates all vectors $\mathbf{ct}_j$ entrywise to obtain a vector $\mathbf{ct}_{\mathsf{aggr}}$ where each entry is of the form $\mathbf{ct}_{\mathsf{aggr}}[j] = (g^{\sum_i r_i}, g^{\sum_i \mathbf{v}_i[j]} \cdot (\mathsf{pk}^{\mathcal{E}})^{\sum_i r_i})$.

2. Partial decryption. T_i locally computes a vector of partial decryptions $\mathbf{ds}_i$ by applying their secret key share $\mathsf{sk}_i^{\mathcal{E}}$ to each entry of $\mathbf{ct}_{\mathsf{aggr}}$. That is, for $\mathbf{ct}_{\mathsf{aggr}}[j] = (h_{1,j}, h_{2,j})$, the tallier computes $\mathbf{ds}_i[j] = (h_{1,j})^{\mathsf{sk}_i^{\mathcal{E}}}$. They further generate proofs $\pi_{i,j}^{\mathsf{DecShare}}$ for each partial decryption $\mathbf{ds}_i[j]$, showing that it was computed using secret witness $\mathsf{sk}_i^{\mathcal{E}}$. T_i combines $\mathbf{ds}_i$ and all proofs into a message, signs it, and publishes the signed message on $\mathcal{P}_{\mathsf{BB}}$.

Computing the result. Everyone can compute the election result by reading from $\mathcal{P}_{\mathsf{BB}}$: Start by recomputing $\mathbf{ct}_{\mathsf{aggr}}$ in the same way as talliers do. If the output returned by $\mathcal{P}_{\mathsf{BB}}$ contains, after the (first) valid signed `VotingClosed` message, at least one valid signed partial decryption message with valid proofs $\pi_{i,j}^{\mathsf{DecShare}}$ for each of the n_{talliers} many talliers, then take the first such message for each tallier to obtain partial decryption vectors $\mathbf{ds}_i$. For each entry/candidate j and $\mathbf{ct}_{\mathsf{aggr}}[j] = (h_{1,j}, h_{2,j})$, compute $h_{2,j} \cdot (\prod_{i \in \{1,\ldots,n_{\mathsf{talliers}}\}} \mathbf{ds}_i[j])^{-1} = g^{\sum_i \mathbf{v}_i[j]}$. The votes $\sum_i \mathbf{v}_i[j]$ for candidate j can then be obtained by computing the discrete logarithm; this is feasible since the set of possible exponents is small in this case.

If the above fails, e.g., because in the output read from $\mathcal{P}_{\mathsf{BB}}$ the setup or voting phases have not yet concluded, the setup failed and $\mathsf{pk}^{\mathcal{E}}$ is undefined, or there are not (yet) valid partial decryption messages for all talliers, then there is no election result (an error election result). This, again, depends on the party reading from $\mathcal{P}_{\mathsf{BB}}$ and might be a temporary error.

Election verification / E2E verifiability. By reading from $\mathcal{P}_{\mathsf{BB}}$ and given the published voter complaints, everyone can verify that the election result indeed corresponds to the will of the voters. We define the exact procedure as part of our verifiability proof in Section V.

B. Discussion

Reliable channels for complaints. All e-voting systems, irrespective of the type of BB they use, have to deal with the following general issue: if the tally is missing a ballot from some voter V_i, then only V_i themself can determine whether this is correct, i.e., whether they intended to abstain or their ballot was dropped maliciously. A standard approach to solve this issue is to require that voters verify whether their ballot was dropped and, if so, notify election authorities. Suppose an adversary can drop or hide these notifications/complaints. In that case the adversary can break E2E verifiability because they can drop the corresponding votes without the public noticing that the election result is incorrect.

Reliable channels for complaints are thus inherently required for verifiable elections, yet this requirement often has remained implicit or is covered up by much stronger assumptions. In works that consider individual and universal verifiability (e.g., [2], [12]), handling failures of vote verification is typically considered out of scope of the cryptographic protocol. While analyzing E2E verifiability requires treating complaints explicitly as part of the protocol – at least for systems where submitted ballots might get dropped – works such as [50] post complaints on a perfect BB that already offers much stronger properties. This, however, is insecure when a BB might misbehave or networks are asynchronous. E-Voting systems based on real BBs thus have to explicitly require a separate reliable channel for E2E verifiability.

This is still a drastic improvement over assuming a perfect or a real but reliable BB. Not only does this assumption cover far fewer messages. It is also not necessary that complaints are consistent, ordered in a specific manner such as chronologically, or immediately visible to everyone. In practice, a voter could, e.g., send their complaint to many different servers that provide it to anyone verifying the election but that, unlike real BBs, do not have to interact and synchronize with each other; the same server does not even have to

provide consistent sets of complaints to different users. As long as the complaint is not dropped on at least one route, this is sufficient for security. We also note that our E2E verifiability analysis (Section V) can easily be adapted to a weaker assumption where channels are somewhat unreliable such that a certain number of complaints might get dropped, weakening verifiability accordingly, cf. Footnote 6.

Stability under message extension. Since even consistent outputs of $\mathcal{P}_{\mathsf{BB}}$ can still differ in length, it is crucial that once a property of the election system - such as the public election key, a change of two phases, the election result, or the ballot that is counted towards the result for a specific user - can be determined from an output, then any longer output will lead to the same property. Otherwise, verifiability can break down in various ways. For example, if a tallier can later change the public encryption key or the point in time when setup was concluded, say, by submitting a second setup message with a different public key share, then this would retroactively invalidate already submitted and verified votes. This is a security critical point that has to be considered in specifications, unlike for systems using perfect BBs.

Further subtle differences compared to using a perfect BB. Voting systems building on top of perfect BBs might optionally use but do not need explicit messages to indicate the start of a new protocol phase. This is because readers of such BBs are synchronized by always seeing the full and current state of the BB. However, for a real BB where voters might not see all stored messages, an explicit `VotingClosed` message or an equivalent mechanism becomes mandatory. Without this, it could happen that vote verification succeeds because the voter finds their ballot at, say, position 100 of an output of $\mathcal{P}_{\mathsf{BB}}$. However, it is not counted because $\mathcal{P}_{\mathsf{BB}}$ might show only the first 99 votes to talliers when they start tallying.

Also, if election parameters are distributed via a real BB, then the BB or even just a network adversary can prevent voters from running the voting algorithm, effectively dropping their vote, by never delivering necessary data such as the public encryption key. By the same reasoning as for submitted but dropped ballots, handling this case in verification requires a complaint `UnableToVote`. Note that this type of complaint is not needed in systems using reliable perfect BBs.

IV. E2E Verifiability For Malicious BBs

In this section, we observe that we cannot apply existing verifiability notions for analyzing e-voting systems where the BB can misbehave arbitrarily – including accountable BBs as a special case. We hence propose a new, more general E2E verifiability notion by instantiating the *KTV verifiability framework* [26], a general framework for designing new verifiability notions for arbitrary types of protocols. KTV appears to be the best choice for this task since it covers standard e-voting verifiability notions as instances [18] and forms the basis of the new notions for threshold trusted BBs proposed in [5].

Our insights and results complement and extend those by Hirschi et al. [5], who previously studied verifiability notions in the presence of misbehaving BBs, observed that existing notions could not be applied, and proposed a different solution that, however, is not applicable to accountable BBs. We compare both works in Section IV-D.

We start by briefly recalling the KTV framework including existing verifiability notions for e-voting systems.

A. Recap: The KTV Verifiability Framework

Judges. Just as for accountability in Section II, the KTV verifiability framework uses the concept of a *judge* J to formalize the *(judging) algorithm/procedure* used to verify the correctness of a protocol run. Intuitively, its task is to detect whether a desired *security goal*, such as the correctness of an election result, has been violated and output `reject` in this case; otherwise, it may output `accept`. To make this decision, the judge might *collect or receive some data as evidence* and *performs certain checks*, such as the verification of zero-knowledge proofs, that depend on the protocol specification. For e-voting protocols, the input to the judge typically consists solely of public information, i.e., in our case, the information posted on the bulletin board and complaints issued by voters via their public reliable channels. As in Section II, one can thus think of J as a "virtual" entity that only exists for modeling purposes. In reality, the judging procedure can be carried out by any party, including external observers and also voters themselves, who have access to the same evidence.

Goals. The KTV verifiability framework is centered around the notion of a security goal of a protocol $\mathcal{P}$. Formally, a goal γ is simply a set of protocol runs that contains exactly those runs that are "correct" in some protocol-specific sense. Specifying a new verifiability definition via the KTV framework mainly entails defining a suitable goal.

Cortier et al. [18] expressed and compared several established verifiability notions for e-voting systems as KTV goals. They conclude by recommending the following goal $\gamma_k(\phi)$ for defining and analyzing the E2E verifiability of e-voting systems. Here, ϕ is a boolean formula describing which combinations of protocol participants are assumed to be honest for verifiability.[3] Intuitively, $\gamma_k(\phi)$ is met if the election result can be obtained with at most $k \in \mathbb{N}$ modifications (e.g., dropping votes) to the intended votes of honest voters. The parameter k allows for computing and comparing how likely different numbers of vote manipulations might go undetected by verification. For example, some voters might not check that their ballot actually ended up in the BB, giving an adversary a non-zero chance δ to drop those ballots without this being detected. However, in a good voting system, δ should decrease the more votes are dropped, i.e., the higher k. This goal and close variants have been used to analyze several e-voting systems (see, e.g., [6], [26], [50], [54], [56]).

Formally, $\gamma_k(\phi)$ is defined using a distance function $dist$. This in turn is defined based on a counting function $f_{\text{count}} \colon \widetilde{\mathsf{C}}^* \to \mathbb{N}^{\widetilde{\mathsf{C}}}$ which, for a vector $\vec{\mathbf{v}} = (\mathbf{v}_1, \ldots, \mathbf{v}_l) \in \widetilde{\mathsf{C}}^*$

[3] A boolean formula captures static corruption. As described in [18], this can easily be generalized for arbitrary conditions on dynamic corruptions.

(representing a multiset of valid votes including abstention; cf. Footnote 2), counts how many times each vote $\mathbf{v} \in \widetilde{\mathsf{C}}$ occurs in $\vec{\mathbf{v}}$. For example, for $\widetilde{\mathsf{C}}$ consisting of three options A, B, and C, we get $f_{\text{count}}(B, B, C) = (0, 2, 1)$, i.e., it assigns 0 to A, 2 to B, and 1 to C. By $f_{\text{count}}(\vec{\mathbf{v}})[\mathbf{v}]$ we denote the $\mathbf{v}$-th entry of $f_{\text{count}}(\vec{\mathbf{v}})$, i.e., the number of times vote $\mathbf{v}$ was counted by $f_{\text{count}}(\vec{\mathbf{v}})$ - e.g., $f_{\text{count}}(B, B, C)[B] = 2$. For two vectors of votes $\vec{\mathbf{v}}_0, \vec{\mathbf{v}}_1$, the distance function $dist$ is then defined by

$$dist(\vec{\mathbf{v}}_0, \vec{\mathbf{v}}_1) = \sum_{\mathbf{v} \in \widetilde{\mathsf{C}}} \left| f_{\text{count}}(\vec{\mathbf{v}}_0)[\mathbf{v}] - f_{\text{count}}(\vec{\mathbf{v}}_1)[\mathbf{v}] \right| .$$

For example, $dist((B, C, C), (A, C, C, C)) = 3$.

Now, given a run r of a voting protocol, let $(\mathbf{v}_i \in \widetilde{\mathsf{C}})_{i \in \mathbf{id}^{\mathsf{hon}}_{\mathsf{voters}}}$ be the vector of intended votes of the honest voters $\mathbf{id}^{\mathsf{hon}}_{\mathsf{voters}}$ in r. Then, the goal $\gamma_k(\phi)$ is satisfied in r (i.e., r belongs to $\gamma_k(\phi)$) if one of the following conditions holds:

(a) the honesty assumptions are not met (ϕ is false) since then the judge cannot be required to detect errors, or

(b) ϕ holds true in r and there exist votes $(\mathbf{v}'_i \in \widetilde{\mathsf{C}})_{i \in \mathbf{id}^{\mathsf{dishon}}_{\mathsf{voters}}}$ (representing possible votes of the dishonest voters $\mathbf{id}^{\mathsf{dishon}}_{\mathsf{voters}}$ in r) and votes $\vec{\mathbf{v}}_{\mathsf{real}} = (\mathbf{v}^{\mathsf{real}}_i \in \widetilde{\mathsf{C}})_{1 \leq i \leq n_{\mathsf{voters}}}$(representing votes of honest and dishonest voters that led to the published election result) such that:

  (i) a (non-error) election result is published in r and this result is equal to the result that is computed from $\vec{\mathbf{v}}_{\mathsf{real}}$,

  (ii) $dist(\vec{\mathbf{v}}_{\mathsf{ideal}}, \vec{\mathbf{v}}_{\mathsf{real}}) \leq k$,

  where $\vec{\mathbf{v}}_{\mathsf{ideal}}$ is the concatenation of the vectors $(\mathbf{v}_i)_{i \in \mathbf{id}^{\mathsf{hon}}_{\mathsf{voters}}}$ and $(\mathbf{v}'_i)_{i \in \mathbf{id}^{\mathsf{dishon}}_{\mathsf{voters}}}$.

Verifiability. Once a goal γ is defined, this yields a corresponding verifiability definition following a simple idea: The judge J should accept a run only if the goal γ is met, except for an error probability of δ called *verifiability tolerance*. While $\delta = 0$ is desirable, as explained above there is usually a chance $\delta > 0$ for an attack to go unnoticed. The value of δ typically depends on but should decrease exponentially in the number of vote modifications k for good voting systems.

Formally, we say that a function $f : \mathbb{N} \to [0, 1]$ is *δ-bounded* if, for every $c > 0$, there exists η_0 such that $f(\eta) \leq \delta + \eta^{-c}$ for all $\eta > \eta_0$. Then:

Definition 1 (Verifiability [26])**.** *Let $\mathcal{P}$ be a protocol that includes a judge J, i.e., specifies a verification procedure. Let $\delta \in [0, 1]$ be a tolerance and γ be a goal. Then we say that* the protocol $\mathcal{P}$ is (γ, δ)-verifiable by the judge/verification procedure J *if for all ppt adversaries $\mathcal{A}$ running with the protocol $\mathcal{P}$ (written $(\mathcal{A} \mid \mathcal{P}) =: \pi$) the probability $\mathsf{Pr}[\pi(1^\eta) \mapsto \neg\gamma, (\mathsf{J} : \texttt{accept})]$ that π running with security parameter η produces a run which is not in γ but mistakenly accepted by J is δ-bounded as a function of η.*

B. Verifiability in the Context of Malicious BBs

The goal $\gamma_k(\phi)$ as well as all other goals/corresponding definitions studied in [18] are well-defined only if, in runs where ϕ is true, (i) there exists a uniquely defined election result and (ii) there exists a uniquely defined set of eligible voters (which defines the sets $\mathbf{id}^{\mathsf{hon}}_{\mathsf{voters}}$ and $\mathbf{id}^{\mathsf{dishon}}_{\mathsf{voters}}$). Both requirements are easy to achieve by assuming a trusted BB that guarantees to distribute the same view on this information to everyone. However, a real BB, including accountable ones, might misbehave and provide different contradicting views to different parties such that these goals cannot be applied. Notably, these requirements are not just technicalities of the definition of $\gamma_k(\phi)$ but preclude general classes of attacks on verifiability that a real BB might perform and which should be detected by a security notion:

As for (i), a malicious BB might collude with malicious talliers to show each voter different sets of ballots with correct decryption proofs for the corresponding aggregated ciphertexts such that each set yields a different election result. The election then fails at a fundamental level since it does not establish agreement on a result and should, therefore, also not succeed at verification. However, each voter would still be able to verify and hence be convinced that their own local election result is the correct one in the sense that their encrypted ballot was included in the set of ballots that they see and their result was computed correctly from this set of ballots. In practice, one can easily avoid this issue by broadcasting an official result outside of the BB, e.g., via television. However, if such a separate secure broadcast channel is necessary for security, then this should be detected by a good security notion.

As for (ii), if the BB distributes the set of eligible voters in a way that a malicious BB can send different sets to different parties, then it can again collude with malicious talliers to run the following attack: Let A, B, C be three voters who want to vote for X and let D, E be two voters who want to vote for Y. The BB shows A the set of eligible voters $\{A, D, E\}$, shows $\{B, D, E\}$ to B, and shows $\{C, D, E\}$ to everyone else including C, D, E. Now, even if the BB shows all submitted ballots to everyone, each voter would consider a different set of ballots to be eligible and, thus, to be tallied. If the BB provides different proofs of decryption depending on which ballots are expected to be tallied, it can convince all voters that their vote is tallied and that tallying was performed correctly. Yet, all parties agree on the incorrect result Y rather than X.

The above also establishes two novel gaps between the commonly used combination of individual+universal verifiability and full E2E verifiability: A malicious BB might convince voters that their vote is included in the tally (individual) and tallying was performed correctly (universal), yet this does not imply that there is a unique result that takes the honest votes into account (E2E). Together with two related but different gaps due to malicious BBs observed in [5] (see Section IV-D), this shows that an E2E verifiability notion is necessary to obtain meaningful security results when BBs might misbehave. Since this is still missing, we next propose the first such notion.

C. Generalizing E2E Verifiability

We generalize the goal $\gamma_k(\phi)$, thus obtaining a new E2E verifiability notion via Definition 1 that is applicable in a broader range of settings following a natural idea: We remove any runs from the goal where the election result or the set of

eligible participants is not uniquely determined and identical for all honest parties that compute a result resp. read a set. This adaptation makes the new generalized goal $\gamma_k^{\text{gen}}(\phi)$ well-defined even if the BB misbehaves. It also strengthens the corresponding verifiability notion as now the verification procedure/judge must additionally reject the election as invalid whenever the cases (i) and (ii) from Section IV-B occur, which captures that these are possible attacks that should be detected. We note that $\gamma_k^{\text{gen}}(\phi)$ does not require *all* parties to be able to compute/see both the set of eligible voters and the election result. Indeed, in a real-world unreliable network, it can always happen that, e.g., some parties have not yet received all outputs from the BB and hence cannot yet compute the verified result. Formally:

Definition 2 (General E2E Verifiability)**.** *Let ϕ be a boolean formula describing honesty assumptions and let $k \in \mathbb{N}$. Let r be a run of an arbitrary voting system (as in Definition 1), which might include a malicious BB. Then r is in $\gamma_k^{gen}(\phi)$ if the honesty assumptions do not hold, i.e., ϕ is false, or the honesty assumptions are met and all of the following is true:*

- *a) The set containing all non-error election results computed by honest parties in r has size 1. That is, the election has produced a result, say result_r, it is unique, and all honest parties that computed a non-error result in r agree on it.*
- *b) The set containing all sets of eligible voters obtained by honest parties in r has size 1. That is, there is exactly one set of eligible voters in r, and all honest parties that see such a set also agree on it.*
- *c) Let $(\mathbf{v}_i \in \widetilde{\mathsf{C}})_{i \in \mathbf{id}_{\text{voters}}^{hon}}$ be the vector of intended votes of the honest eligible voters $\mathbf{id}_{\text{voters}}^{hon}$ in r. Then there exist votes $(\mathbf{v}'_i \in \widetilde{\mathsf{C}})_{i \in \mathbf{id}_{\text{voters}}^{dishon}}$ for the dishonest eligible voters $\mathbf{id}_{\text{voters}}^{dishon}$ and votes $\vec{\mathbf{v}}_{\mathsf{real}} = (\mathbf{v}_i^{\mathsf{real}} \in \widetilde{\mathsf{C}})_{1 \leq i \leq n_{\text{voters}}}$ such that result_r equals the result computed from all votes in $\vec{\mathbf{v}}_{\mathsf{real}}$, and $dist(\vec{\mathbf{v}}_{\mathsf{ideal}}, \vec{\mathbf{v}}_{\mathsf{real}}) \leq k$ where $\vec{\mathbf{v}}_{\mathsf{ideal}}$ is the concatenation of $(\mathbf{v}_i)_{i \in \mathbf{id}_{\text{voters}}^{hon}}$ and $(\mathbf{v}'_i)_{i \in \mathbf{id}_{\text{voters}}^{dishon}}$.*

Discussion. While the main motivation for our new E2E verifiability definition based on the goal $\gamma_k^{\text{gen}}(\phi)$ are e-voting systems using fully untrusted but accountable BBs, Definition 2 is actually agnostic to the type of BB or whether a BB is used at all. It thus applies more generally to several other e-voting systems, including: *(i)* Systems based on perfect BBs where Conditions a) and b) in Definition 2 are trivially met in all runs and do not impose additional requirements on the judge. The goal $\gamma_k(\phi)$ is thus captured as a special case of $\gamma_k^{\text{gen}}(\phi)$. *(ii)* Systems based on real BBs that ensure (preventive security of) consistency assuming a trusted threshold of servers. *(iii)* Systems that establish agreement on eligible voters and/or the election result via a separate channel outside of a BB – say, by broadcasting the result on television – such that even a fully untrusted BB without accountability cannot present different results to different parties. *(iv)* Systems that do not use any BBs, e.g., because they are distributed protocols run among the voters themselves.

Conditions a) and b) mention honest parties that may or may not have obtained/computed the set of eligible voters/the election result. Typically, as for instance in ABBOVE in Section V-C, such parties include all voters, the (potentially external) parties running the verification procedure modeled by the judge, and any other parties with access to the BB.

Condition a) gives guarantees only for parties that successfully compute a (non-error) election result. It does not, however, require that *all* honest parties who run the election result computation are successful. This is because in asynchronous real world networks it can always happen that, say, Alice can compute a non-error election result while Bob at the same time might not (yet) see, e.g., all messages from the tallying phase such that he is unable to compute the result. For the same reason, Condition b) also gives guarantees only for those parties who succeed at reading a set of eligible voters.

Security analyses using previous verifiability notions based on perfect BBs only have to model runs of an e-voting system up to the point where talliers publish their final messages. Since Condition a) adds a statement about different parties computing a result from those messages – possibly at different points in time – a security analysis using Definition 2 should in addition also model arbitrary parties running the election result computation. We recommend letting the adversary decide who runs this computation and when. Since verifiability is shown for all adversaries, this then covers arbitrary combinations and scheduling of parties computing the election result.

Our goal $\gamma_k^{\text{gen}}(\phi)$ and the corresponding verifiability notion consider only a single judge. While one might expect that one has to consider multiple concurrent judges to define (E2E) verifiability in a setting where a malicious BB can provide different views to different parties running the same verification procedure, this is not necessary: If a single judge – representing an arbitrary but fixed party running the judging procedure on its local view of BB – can verify the election result, then the goal $\gamma_k^{\text{gen}}(\phi)$ implies that this result is correct, i.e., was obtained with less than k modifications to honest votes, and that all other honest parties who obtain an election result agree on this correct one. Conversely, if parties do not agree on a unique election result or the result is unique but incorrect, then no matter who runs the judge/which BB view is used, the judge will reject.

D. Comparison to Hirschi et al. [5]

Hirschi et al. also start from the verifiability definitions (expressed as KTV goals) in [18]. They observe that all of these definitions use perfect BBs providing identical outputs to all parties after the end of the election and are thus no longer well-defined for real BBs run by potentially malicious servers. Hirschi et al. then identify two general attack classes not covered by these definitions where a malicious BB shows different sets of ballots to different voters such that individual and universal verifiability checks succeed, yet all voters agree on the same but incorrect result, i.e., E2E verifiability is broken.

To solve well-definedness of previous verifiability notions in the presence of real BBs, Hirschi et al. propose a novel security notion for real BBs called *final consistency*. Intuitively, a BB achieves final consistency if it reaches a "final" state where it stops accepting inputs after an election ends and then guarantees that, whenever a party receives a final state output from the BB, then this output will be identical to final state outputs received by all other parties. Identical final outputs are sufficient such that verifiability definitions based on perfect BBs, such as the ones studied in [18], remain well-defined when the perfect BB is replaced by a final consistent one. They therefore add the requirement to verifiability notions that the BB used in an e-voting system has the final consistency property. Hirschi et al. call these new notions *verifiability+* notions. They also show that real BBs with final consistency can be constructed from digital signatures and assuming a trusted threshold of servers running the BB. They further prove that verifiability results based on perfect BBs imply the corresponding verifiability+ version using a final consistent BB under some mild additional requirements.

The requirements and general attack classes (i) and (ii) that we identify in Section IV-B are closely related to the work of Hirschi et al. but new and distinct. Crucially, while finally consistent BBs are sufficient for previous verifiability definitions to remain well-defined, this requirement imposed by verifiability+ is stronger than requirements (i) and (ii) we identify. Verifiability+ cannot be applied to a wide range of voting systems such as those using fully untrusted BBs, systems using accountable BBs (as accountability does not preclude misbehavior such as different final outputs), systems using BBs that run continuously and never reach a final state, say, because they are shared with other elections, and systems without BBs such as distributed protocols run among voters.

In contrast, our general verifiability notion given in Definition 2, which is based on the less demanding and more abstract requirements (i) and (ii), does not mandate a specific BB – or even any BB at all. It is in fact applicable to all of the above cases. It further captures the verifiability+ variant of $\gamma_k(\phi)$ as a special case: Consider a system with a finally consistent BB. As long as the trust assumption for that BB holds, all readers obtain identical final outputs and hence will always compute the same set of eligible voters and election result from those outputs. The first two conditions of Definition 2 are thus always trivially met such that, just as in verifiability+, the judging procedure only has to verify the original $\gamma_k(\phi)$.

V. E2E Verifiability of ABBOVE

In this section, we use our new E2E verifiability definition from Section IV-C to analyze the verifiability of ABBOVE.

A. Computational Model

We model ABBOVE using a general and established computational framework (see, e.g., [18], [26], [50], [54], [56]) which is based on the notion of a process. A process $\pi_{\mathcal{P}}$ modeling some protocol $\mathcal{P}$ is a set of interacting ppt Turing machines (ITMs, sometimes also called *programs*) that can send messages to each other via connected tapes. At any point in time, only one machine is actively running, namely, the most recent one to receive a message. The protocol $\mathcal{P}$ runs alongside an adversary $\mathcal{A}$, modeled via another process $\pi_{\mathcal{A}}$, which controls the network and may corrupt protocol participants by sending a special `corrupt` signal via a connected tape to their ITMs. A corrupted ITM then acts as a pure message forwarder and thus gives full control to $\mathcal{A}$. Here, we only consider static corruption of parties. We write $\pi = (\pi_{\mathcal{P}} \mid \pi_{\mathcal{A}})$ for the combined process. We often use $\mathcal{P}$ and $\pi_{\mathcal{P}}$ interchangeably; the same for $\mathcal{A}$ and $\pi_{\mathcal{A}}$.

ABBOVE can be modeled straightforwardly as a protocol $\mathcal{P}_{\mathsf{ABBOVE}}(n_{\mathsf{voters}}, n_{\mathsf{talliers}}, \mathsf{C}, \mu_{\mathrm{vote}}, p_{\mathrm{verif}})$ in the above sense. Recall from Section III that we denote the number of voters by n_{voters}, the number of talliers by n_{talliers}, and the set of valid votes by $\widetilde{\mathsf{C}} := \mathsf{C} \cup \{\mathtt{abstain}\}$. By μ_{vote}, we denote a distribution over $\widetilde{\mathsf{C}}$ according to which each honest voter chooses their *intended vote*, and by $p_{\mathrm{verif}} \in [0,1]$ the probability that an honest voter chooses to verify their vote. (By this, we model that the adversary knows both distributions.)

In our model of ABBOVE, the trusted voting authority Auth is part of an additional incorruptible agent, the scheduler Sch. Besides playing the role of the authority, Sch schedules all other agents in a run according to the protocol. In particular, while the adversary can determine the order of activations of parties, Sch ensures that all honest voters have been activated at least once to (try to) vote before Auth closes the voting phase. Sch also ensures that all honest voters get activated at least once after their vote submission to decide whether they want to run vote verification. Sch further collects any complaints from voters and forwards them directly to the judge while also making them available to $\mathcal{A}$, which captures reliable public complaint channels. Besides Sch, we also model J as incorruptible. Note that this is not a trust assumption. It rather captures that the judge represents an arbitrary party running the judging algorithm on public data (see Section IV-A). An honest party can thus always choose to run the judging algorithm themselves to guarantee a correct execution without needing to trust a third party. We let the adversary decide when J runs election verification, which then produces an overall output `reject` or `accept` in that run.

Our model of ABBOVE includes an arbitrary accountable real BB $\mathcal{P}_{\mathsf{BB}}$ defined as a UC protocol. This is possible because UC protocols are defined also using interacting ppt Turing machines. Since each party P of the voting protocol runs the client code of $\mathcal{P}_{\mathsf{BB}}$, we formally represent a party P by two ITMs in our model: A higher-level program π_{hl} which runs the logic of the voting protocol and communicates via a directly connected tape with a subroutine $\pi_{\text{client-BB}}$ that runs the client logic of $\mathcal{P}_{\mathsf{BB}}$.[4] Since, in reality, both programs represent the same party and run on the same computer, we consider the corruption of one program to simultaneously corrupt the other.

[4]Interested readers might want to look at Figure 2 in Appendix A that illustrates the overall structure of our model.

B. Judge J

Next, we formally specify the judge J for ABBOVE. Since J defines the exact verification procedure used for verifying a run of the election, this is an integral part of the protocol specification of ABBOVE that we now complete.

The judge J has access to all published complaints of voters, which captures reliable complaint channels. It runs the accountability judging procedure $\mathsf{J}_{\mathsf{BB}}^{\mathsf{acc}}$ of $\mathcal{P}_{\mathsf{BB}}$ as a subroutine. Verification then proceeds as follows:

1) Read from $\mathcal{P}_{\mathsf{BB}}$ to obtain a sequence of messages msgseq and run the judging procedure $\mathsf{J}_{\mathsf{BB}}^{\mathsf{acc}}$ as specified by $\mathcal{P}_{\mathsf{BB}}$. If this detects a misbehaving party (blamed in a verdict), then J outputs `reject`. Otherwise, msgseq is consistent with all outputs read by honest parties; the following computations are thus performed based on msgseq.
2) Try to compute the election result as per Section III. Recall that this involves, among others, computing the public encryption key from setup messages, recomputing the aggregated ciphertexts, and verifying signatures and NIZKPs. If this fails, i.e., the result is $\perp$, output `reject`.
3) Let $\mathbf{id}_{\mathsf{voters}}$ be the set of eligible voters from the election setup message contained in msgseq (if msgseq does not contain this message or it is invalid, then the previous step already rejects since no election result can be computed). Check if there is a complaint with a valid signature signed by one of the voters in $\mathbf{id}_{\mathsf{voters}}$. If so, output `reject`.[5]
4) Otherwise, `accept` the election and its result.

C. E2E Verifiability Analysis of ABBOVE

Our security result uses the following assumptions:

(V1) π^{Enc} and π^{DecShare} are NIZKPs and the signature scheme $\mathcal{S}$ is EUF-CMA-secure (we recall formal security definitions of cryptographic primitives we use in the full version [61]).

(V2) There is at most a negligible chance for two honest voters to generate the same ciphertext vector $\mathbf{ct}$. E.g., for non-trivial ElGamal groups this is implied by the DDH assumption.

(V3) $\phi := \mathsf{hon}(\mathsf{Sch}) \wedge \mathsf{hon}(\mathsf{J})$.

(V4) Voters have reliable channels to publish complaints outside the BB.

Theorem 2. *Let $\mathcal{F}_{\mathsf{BB}}$ be an ideal BB functionality as described in Section II that provides at least accountability w.r.t. consistency, possibly in addition to further security or functional properties. Let $\mathcal{P}_{\mathsf{BB}}$ be an arbitrary real BB protocol such that $\mathcal{P}_{\mathsf{BB}} \leq \mathcal{F}_{\mathsf{BB}}$. Let $\mathcal{P}_{\mathit{ABBOVE}}(n_{\mathsf{voters}}, n_{\mathsf{talliers}}, \mathsf{C}, \mu_{\mathit{vote}}, p_{\mathit{verif}})$ be the voting protocol based on $\mathcal{P}_{\mathsf{BB}}$ as above and including the judge J from Section V-B. Let $k \in \mathbb{N}$. Under the assumptions (V1) to (V4), we have that $\mathcal{P}_{\mathit{ABBOVE}}$ is $(\gamma_k^{\mathit{gen}}(\phi), \delta_k(p_{\mathit{verif}}))$-verifiable where*

$$\delta_k(p_{\mathit{verif}}) = (1 - p_{\mathit{verif}})^{k+1}.$$

Proof sketch (see the full version [61] for the complete proof). The proof consists of two parts. In the first part, we prove the result for a hybrid system $\pi_{\mathsf{ideal\text{-}BB}}$ that works as $\mathcal{P}_{\mathsf{ABBOVE}}$ but replaces $\mathcal{P}_{\mathsf{BB}}$ by $\mathcal{F}_{\mathsf{BB}}$ and its simulator (cf. Figure 2). We perform several game hops using accountability w.r.t. consistency provided by $\mathcal{F}_{\mathsf{BB}}$, standard arguments from verifiability proofs of homomorphic aggregation-based systems using perfect BBs, and additional arguments to handle attacks based on unreliability and BB outputs differing in length (see also Section III-B). Intuitively, accountability w.r.t. consistency and stability under message extension are the main tools to show that if Conditions a) and b) of $\gamma_k^{\mathsf{gen}}(\phi)$ are not met, then this is detected by the judge and the result is rejected. We are further able to show that the only way to change the election result – thus violating Condition c) – while remaining undetected with a non-negligible chance is by not including submitted ballots in the BB state before the `VotingClosed` message. As each dropped ballot changes the result by 1 and the affected voter will complain if they choose to verify their vote, we get an upper bound $(1 - p_{\mathsf{verif}})^{k+1}$ of more than k modifications remaining undetected.[6]

In the second part, we show that the verifiability error for $\mathcal{P}_{\mathsf{ABBOVE}}$ (where $\mathcal{P}_{\mathsf{BB}}$ is used) is negligibly close to the one for $\pi_{\mathsf{ideal\text{-}BB}}$ (where $\mathcal{F}_{\mathsf{BB}}$ is used). While one might expect this to trivially and always follow from the composability of UC security of $\mathcal{P}_{\mathsf{BB}}$, this is, in general, not implied because verifiability is a trace property defined over runs of systems. Intuitively, this is because trace properties might not be computable for UC environments. E.g., a trace property can be defined over and depend on some internal behavior such as variables and messages that the UC environment cannot observe or a trace property might not be decidable in polynomial runtime. The probability for such trace properties can be vastly different between real and ideal world without contradicting UC security. For interested readers, we discuss this gap in more detail and provide two simple counter examples in the full version [61]. Fortunately, we can show via a reduction that in our case the statement indeed follows. □

Corollary 1. *Under the above assumptions and for the same goal and δ_k as in Theorem 2, $\mathcal{P}_{\mathit{ABBOVE}}$ using $\mathsf{Fabric}_{\mathsf{BB}}$ is $(\gamma_k^{\mathit{gen}}(\phi), \delta_k(p_{\mathit{verif}}))$-verifiable.*

Proof. Follows by Theorems 1 and 2. □

Since Theorem 2 only requires $\mathsf{Fabric}_{\mathsf{BB}} \leq \mathcal{F}_{\mathsf{BB}}$, which holds true even when *all* parties running the BB are malicious by Theorem 1, this establishes, for the first time, *E2E verifiability of an e-voting system using a fully untrusted BB*.

VI. Privacy

We analyze the privacy of ABBOVE using the established privacy definition by Küsters et al. in [54]. This definition allows for computing the exact privacy level that an e-voting

[5] To improve robustness against malicious unjustified complaints, the judge can alternatively reject only if there are at least $d+1$ complaints. This leads to a slightly worse verifiability level similar to what we explain in Footnote 6.

[6] If complaint channels are assumed to only be somewhat reliable such that up to d complaints might get dropped, then the verifiability error δ degrades to $\sum_{i=0}^{d} p_{\mathsf{verif}}^{i} (1 - p_{\mathsf{verif}})^{k+1-i}$.

protocol achieves based on factors such as the choice space C, the number $n^{\text{hon}}_{\text{voters}}$ of (non-abstaining) honest voters, and the honest vote distribution μ_{vote}. Alternatively, the definition can also be used in the form of a more intuitive indistinguishability game. This game-based version has already been applied in, e.g., [50], [56] and is what we present and use here as well.

A. Privacy Notion.

In a nutshell, the definition from [54] defines the privacy of an e-voting protocol as the inability of an adversary to distinguish whether an arbitrary but fixed honest voter $\mathsf{v}_{\mathsf{obs}}$ (the *voter under observation*) voted for $\mathbf{v}_0$ or $\mathbf{v}_1$. Here, we consider $\mathbf{v}_0, \mathbf{v}_1 \in \mathsf{C}$, i.e., we do not require that a voting system hides whether $\mathsf{v}_{\mathsf{obs}}$ abstained.

More formally, the privacy notion considers an e-voting protocol $\mathcal{P}$ with choice space C, a security parameter η, a ppt adversary $\mathcal{A}$, and a voter under observation $\mathsf{v}_{\mathsf{obs}}$. Given some vote $\mathbf{v} \in \mathsf{C}$, we denote by $(\pi_{\mathsf{v}_{\mathsf{obs}}}(\mathbf{v}) \mid \pi^*_{\mathcal{P}} \mid \pi_{\mathcal{A}})$ the composition of $\mathsf{v}_{\mathsf{obs}}$'s program $\pi_{\mathsf{v}_{\mathsf{obs}}}(\mathbf{v})$ when taking $\mathbf{v}$ as their intended vote, all programs $\pi^*_{\mathcal{P}}$ of the remaining parties in $\mathcal{P}$, and the adversary's program $\pi_{\mathcal{A}}$. Note that the degree of how well the published aggregated election result hides $\mathsf{v}_{\mathsf{obs}}$'s vote, and hence the privacy of $\mathsf{v}_{\mathsf{obs}}$, depends on the number of honest voters that submit a vote. Conversely, abstaining honest voters do not improve privacy. In this section by $n^{\text{hon}}_{\text{voters}}$ we thus denote the number of non-abstaining honest voters, i.e., for those voters we consider an honest vote distribution μ_{vote} which does not return `abstain`. Honest voters who abstain are still captured but are subsumed in the remaining $n_{\text{voters}} - n^{\text{hon}}_{\text{voters}}$ voters, some of which might be malicious.

Now privacy is defined via the following *privacy game* $\mathbb{E}^{\mathsf{Priv}}(\mathcal{A}, \mathcal{P}, \eta, \mathsf{v}_{\mathsf{obs}})$:

$\mathbb{E}^{\mathsf{Priv}}(\mathcal{A}, \mathcal{P}, \eta, \mathsf{v}_{\mathsf{obs}})$:

1) $\mathcal{A}$ chooses two possible votes $\mathbf{v}_0, \mathbf{v}_1 \in \mathsf{C}$ and sends them to the challenger.
2) The challenger chooses a bit $b \leftarrow \{0,1\}$ uniformly at random, which determines $\mathsf{v}_{\mathsf{obs}}$'s vote $\mathbf{v}_b$.
3) Run the protocol $(\pi_{\mathsf{v}_{\mathsf{obs}}}(\mathbf{v}_b) \mid \pi^*_{\mathcal{P}} \mid \pi_{\mathcal{A}})$.
4) $\mathcal{A}$ guesses a bit $b' \in \{0,1\}$.
5) If $b' = b$, return 1, otherwise return 0.

We denote by $\Pr[\mathbb{E}^{\mathsf{Priv}}(\mathcal{A}, \mathcal{P}, \eta, \mathsf{v}_{\mathsf{obs}}) \mapsto 1]$ the probability that $\mathcal{A}$ wins $\mathbb{E}^{\mathsf{Priv}}(\mathcal{A}, \mathcal{P}, \eta, \mathsf{v}_{\mathsf{obs}})$, where the probability is taken over the random coins used by the parties in $(\pi_{\mathsf{v}_{\mathsf{obs}}} \mid \pi^* \mid \pi_{\mathcal{A}})$ as well as the random choice b by the challenger in step 2.

Privacy is then defined by comparing the winning probability of $\mathcal{A}$ attacking $\mathcal{P}$ with an ideal situation where an adversary/simulator $\mathcal{A}_{\mathsf{ideal}}$ attacks an ideal voting protocol/functionality $\mathcal{P}_{\mathsf{ideal}}$ that achieves the best privacy level by definition. More specifically, $\mathcal{P}_{\mathsf{ideal}}(\mathsf{C}, n_{\text{voters}}, n^{\text{hon}}_{\text{voters}}, \mu_{\text{vote}})$ is defined with the parameters as above. Intuitively, $\mathcal{P}_{\mathsf{ideal}}$ computes an election result as an aggregation of $\mathsf{v}_{\mathsf{obs}}$'s vote $\mathbf{v}_b$, $n^{\text{hon}}_{\text{voters}} - 1$ additional secret votes chosen independently according to μ_{vote}, and up to $n_{\text{voters}} - n^{\text{hon}}_{\text{voters}}$ votes from C but chosen by the adversary. $\mathcal{P}_{\mathsf{ideal}}$ reveals only the election result but nothing else to the adversary. It therefore captures the best case where an adversary does not learn anything about $\mathbf{v}_b$ besides what it can compute from an election result which includes at most one copy of $\mathbf{v}_b$ and (at least) $n^{\text{hon}}_{\text{voters}} - 1$ independent secret votes.[7] We recall the formal definition in Figure 3 in Appendix A.

Privacy is then defined and proven w.r.t. a specific set of adversaries, called "admissible adversaries" in what follows. We explain the purpose of this set and define it for ABBOVE later. Formally:

Definition 3 (Privacy)**.** *Let $\mathcal{P}$ be a voting protocol and let $\mathcal{P}_{\mathsf{ideal}}$ be an ideal voting protocol.*[8] *Then,* $\mathcal{P}$ achieves (ideal) privacy*, if for all admissible adversaries $\mathcal{A}$ on $\mathcal{P}$, there is an adversary $\mathcal{A}_{\mathsf{ideal}}$ on $\mathcal{P}_{\mathsf{ideal}}$ such that*

$$\left|\Pr[\mathbb{E}^{\mathsf{Priv}}(\mathcal{A}, \mathcal{P}, \eta, \mathsf{v}_{\mathsf{obs}}) \mapsto 1] - \Pr[\mathbb{E}^{\mathsf{Priv}}(\mathcal{A}_{\mathsf{ideal}}, \mathcal{P}_{\mathsf{ideal}}, \eta, \mathsf{v}_{\mathsf{obs}}) \mapsto 1]\right|$$

is negligible as a function in η.

The strength of such a privacy result depends on the parameters chosen for $\mathcal{P}_{\mathsf{ideal}}$. Importantly, a good privacy result should achieve a large value of $n^{\text{hon}}_{\text{voters}}$ as this determines how well the election result hides the vote of $\mathsf{v}_{\mathsf{obs}}$. Ideally, $n^{\text{hon}}_{\text{voters}}$ for $\mathcal{P}_{\mathsf{ideal}}$ should be close to or identical to the number of non-abstaining honest voters in the real voting protocol $\mathcal{P}$. To understand the actual level of privacy - i.e., the attacker's success probability - offered by different instances of the ideal protocol $\mathcal{P}_{\mathsf{ideal}}$, including how it varies with parameters like the choice space C, the number of honest voters $n^{\text{hon}}_{\text{voters}}$, and the vote distribution μ_{vote}, we refer interested readers to the analyses provided in [50], [54], [56].

B. Privacy in the Presence of Real BBs

As mentioned in Section I, an adversary $\mathcal{A}$ controlling a real BB or even just the network can trivially break privacy [6], [20], [50]: $\mathcal{A}$ might drop all but $\mathsf{v}_{\mathsf{obs}}$'s ballot which is then tallied and revealed as the result of the election. Formally, this means that in the presence of arbitrarily misbehaving adversaries, Definition 3 can only be shown for $\mathcal{P}_{\mathsf{ideal}}$ with $n^{\text{hon}}_{\text{voters}} = 1$, i.e., when there is no secret vote in the result that hides the one of $\mathsf{v}_{\mathsf{obs}}$ and, hence, $\mathsf{v}_{\mathsf{obs}}$'s vote is trivially revealed also in the ideal protocol.

In [7], Cortier et al. studied malicious BBs and found that, besides dropping inputs/votes, also re-ordering and modifying inputs can break privacy. In addition to these results, we note that equivocation enables further attacks on privacy. For example, suppose the BB distributes the public election key. In that case, the BB might provide different encryption keys to different voters. Among others, this can cause ballots to be successfully submitted yet not be counted towards the result: talliers seeing and using a different public key to verify the NIZKP π^{Enc} will consider π^{Enc} to be invalid w.r.t. their public key and hence drop the ballot. Crucially and unlike for ballots

[7] This also rules out ballot stuffing/replay attacks where, e.g., $\mathbf{v}_b$ is submitted multiple times to inflate its influence on the result [67], [68].

[8] Typically, $\mathcal{P}_{\mathsf{ideal}}$ uses the same parameters as $\mathcal{P}$ except for potentially $n^{\text{hon}}_{\text{voters}}$. We discuss this in what follows.

explicitly dropped by malicious parties, this way of removing a vote from the result would not be detected via vote verification by the affected voter who still finds their correct ballot in the output of the BB. In Appendix B, we describe another type of equivocation attack on privacy that affects systems using a decryption threshold $t < n_{\mathsf{talliers}}$.

By construction, for ABBOVE the above types of attacks come at a high risk that rational adversaries would typically avoid. Firstly, since we use an accountable BB, i.e., $\mathcal{P}_{\mathsf{BB}} \leq \mathcal{F}_{\mathsf{BB}}$, one can use contractual penalties for detected misbehavior to incentivize malicious BBs to remain consistent which prevents equivocation attacks (but attacks based on consistent outputs that differ in length might still be possible). Secondly, for vote dropping, we have shown as part of proving Theorem 2 that, as long as the BB provides consistent outputs, the only ways for an adversary to drop/modify honest votes will lead to one of two complaints being published – in case of `VoteVerifFailed` with chance p_{verif} of running vote verification. Often, it is reasonable to assume that rational adversaries do not want to risk invalidating the entire election due to complaints, which might cause an investigation, just to reveal a single or a few votes. This observation motivated the notion of risk-avoiding adversaries in [50], [55] for which meaningful privacy results can be shown. Also other works such as [7] follow a similar idea by defining privacy only for adversaries that are unwilling/unable to risk a failure in vote verification or who do not learn the election result if there is a failure. In fact, in [20] Cortier and Lallemand showed that successful vote verification is necessary for privacy.

We therefore show privacy of ABBOVE for the following class of rational adversaries. We call an adversary $\mathcal{A}$ *k-risk-avoiding* if the following two events (taken over all runs of $\mathcal{A}$ with ABBOVE) have negligible probability: *(i)* The BB provides inconsistent views to two (or more) honest parties. *(ii)* If an output returned by the BB to an honest party contains a successful setup phase and a valid signed `VotingClosed` message (i.e., the output would be tallied by an honest tallier), then in that output and before the first `VotingClosed` message there are less than $n^{\text{hon}}_{\mathsf{voters}} - k$ signed ballots submitted by honest voters. That is, $\mathcal{A}$ is willing to drop at most k ballots submitted by honest voters as $\mathcal{A}$ considers the probability of a complaint due to vote verification to be too high for larger numbers.

C. Privacy of ABBOVE

We prove privacy using the following assumptions and class of adversaries:

(P1) An adversary is admissible if it *(i)* does not corrupt Sch, *(ii)* corrupts at most $n_{\mathsf{talliers}} - 1$ talliers, *(iii)* corrupts at most $n^{\text{dishon}}_{\mathsf{voters}} = n_{\mathsf{voters}} - n^{\text{hon}}_{\mathsf{voters}}$ voters, and *(iv)* is k-risk-avoiding.

(P2) The ElGamal group $\mathbb{G}$ is non-trivial and the DDH problem is hard in $\mathbb{G}$.

(P3) $\pi^{\mathsf{KeyShareGen}}$, π^{Enc}, and π^{DecShare} are NIZKPs and the signature scheme $\mathcal{S}$ is EUF-CMA-secure (we recall formal security definitions of the cryptographic primitives we use in the full version [61]).

Now, we can show the following:

Theorem 3. *Consider ABBOVE for choice space C, number n_{voters} of voters, $n^{hon}_{\mathsf{voters}}$ a lower bound for the number of honest non-abstaining voters, and vote distribution μ_{vote} for those voters. Then ABBOVE achieves ideal privacy for $\mathcal{P}_{\mathsf{ideal}}(n_{\mathsf{voters}}, n^{hon}_{\mathsf{voters}} - k, \mathsf{C}, \mu_{vote})$ against admissible adversaries in the sense of (P1) under Assumptions (P2) and (P3).*

We give a detailed proof in the full version [61] using a sequence of games that transforms the game $\mathbb{E}^{\mathsf{Priv}}(\mathcal{A}, \mathcal{P}, \eta, \mathsf{v}_{\mathsf{obs}})$ to the game $\mathbb{E}^{\mathsf{Priv}}(\mathcal{A}_{\mathsf{ideal}}, \mathcal{P}_{\mathsf{ideal}}, \eta, \mathsf{v}_{\mathsf{obs}})$ for an ideal adversary $\mathcal{A}_{\mathsf{ideal}}$. Theorem 1 then implies:

Corollary 2. *ABBOVE using $\mathsf{Fabric}_{\mathsf{BB}}$ achieves ideal privacy for the same parameters as in Theorem 3.*

While showing Theorem 3, we found a gap in the privacy proof of Helios given in [11] which is of independent interest. We provide details in Appendix B.

Acknowledgements

This research was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), Project-ID 459731562. We thank Mike Graf for helpful discussions and feedback.

References

[1] J. Benaloh, "Simple Verifiable Elections," in *2006 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'06, Vancouver, BC, Canada, August 1, 2006*. USENIX Association, 2006.

[2] V. Iovino, A. Rial, P. B. Rønne, and P. Y. A. Ryan, "Universal Unconditional Verifiability in E-Voting without Trusted Parties," in *33rd IEEE Computer Security Foundations Symposium, CSF 2020, Boston, MA, USA, June 22-26, 2020*. IEEE, 2020, pp. 33–48.

[3] J. Benaloh, M. Naehrig, and O. Pereira, "REACTIVE: Rethinking Effective Approaches Concerning Trustees in Verifiable Elections," Cryptology ePrint Archive, Tech. Rep. 2024/915, 2024.

[4] N. Chondros, B. Zhang, T. Zacharias, P. Diamantopoulos, S. Maneas, C. Patsonakis, A. Delis, A. Kiayias, and M. Roussopoulos, "D-DEMOS: A Distributed, End-to-End Verifiable, Internet Voting System," in *36th IEEE International Conference on Distributed Computing Systems, ICDCS 2016, Nara, Japan, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 711–720.

[5] L. Hirschi, L. Schmid, and D. A. Basin, "Fixing the Achilles Heel of E-Voting: The Bulletin Board," in *34th IEEE Computer Security Foundations Symposium, CSF 2021, Dubrovnik, Croatia, June 21-25, 2021*. IEEE, 2021, pp. 1–17.

[6] R. Küsters, T. Truderung, and A. Vogt, "Clash Attacks on the Verifiability of E-Voting Systems," in *33rd IEEE Symposium on Security and Privacy (S&P 2012)*. IEEE Computer Society, 2012, pp. 395–409.

[7] V. Cortier, J. Lallemand, and B. Warinschi, "Fifty Shades of Ballot Privacy: Privacy against a Malicious Board," in *IEEE 33rd Computer Security Foundations Symposium, CSF 2020, 22-25 July, 2020*. IEEE Computer Society, 2020.

[8] S. Hauser and R. Haenni, "Modeling a Bulletin Board Service Based on Broadcast Channels with Memory," in *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 10958. Springer, 2018, pp. 232–246.

[9] A. Kiayias, A. Kuldmaa, H. Lipmaa, J. Siim, and T. Zacharias, "On the Security Properties of e-Voting Bulletin Boards," in *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, ser. Lecture Notes in Computer Science, vol. 11035. Springer, 2018, pp. 505–523.

[10] M. Graf, R. Küsters, D. Rausch, S. Egger, M. Bechtold, and M. Flinspach, "Accountable Bulletin Boards: Definition and Provably Secure Implementation," in *37th IEEE Computer Security Foundations Symposium (CSF 2024)*. IEEE, 2024, to appear.

[11] D. Bernhard, O. Pereira, and B. Warinschi, "How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios," in *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, ser. Lecture Notes in Computer Science, X. Wang and K. Sako, Eds., vol. 7658. Springer, 2012, pp. 626–643.

[12] V. Cortier, C. C. Dragan, F. Dupressoir, and B. Warinschi, "Machine-Checked Proofs for Electronic Voting: Privacy and Verifiability for Belenios," in *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*. IEEE Computer Society, 2018, pp. 298–312.

[13] W. Lueks, I. Querejeta-Azurmendi, and C. Troncoso, "VoteAgain: A scalable coercion-resistant voting system," in *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*. USENIX Association, 2020, pp. 1553–1570.

[14] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi, "SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions," in *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 2015, pp. 499–516.

[15] D. Bernhard, V. Cortier, O. Pereira, B. Smyth, and B. Warinschi, "Adapting Helios for Provable Ballot Privacy," in *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*, ser. Lecture Notes in Computer Science, vol. 6879. Springer, 2011, pp. 335–354.

[16] V. Cortier, D. Galindo, S. Glondu, and M. Izabachène, "Distributed ElGamal à la Pedersen: Application to Helios," in *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*. ACM, 2013, pp. 131–142.

[17] D. A. Basin, S. Radomirovic, and L. Schmid, "Alethea: A Provably Secure Random Sample Voting Protocol," in *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*. IEEE Computer Society, 2018, pp. 283–297.

[18] V. Cortier, D. Galindo, R. Küsters, J. Müller, and T. Truderung, "SoK: Verifiability Notions for E-Voting Protocols," in *IEEE 37th Symposium on Security and Privacy (S&P 2016)*. IEEE Computer Society, 2016, pp. 779–798.

[19] B. Smyth, "Ballot secrecy with malicious bulletin boards," Cryptology ePrint Archive, Tech. Rep. 2014/822, 2014.

[20] V. Cortier and J. 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, Toronto, ON, Canada, October 15-19, 2018*. ACM, 2018, pp. 53–66.

[21] B. Adida, "Helios: Web-based Open-Audit Voting," in *Proceedings of the 17th USENIX Security Symposium*, P. C. van Oorschot, Ed. USENIX Association, 2008, pp. 335–348.

[22] V. Cortier, P. Gaudry, and S. Glondu, "Belenios: A Simple Private and Verifiable Electronic Voting System," in *Foundations of Security, Protocols, and Equational Reasoning - Essays Dedicated to Catherine A. Meadows*, ser. Lecture Notes in Computer Science, vol. 11565. Springer, 2019, pp. 214–238.

[23] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a Secure Voting System," in *2008 IEEE Symposium on Security and Privacy (S&P 2008)*. IEEE Computer Society, 2008, pp. 354–368.

[24] C. Culnane and S. A. Schneider, "A Peered Bulletin Board for Robust Use in Verifiable Voting Systems," in *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*. IEEE Computer Society, 2014, pp. 169–183.

[25] J. Feigenbaum, A. D. Jaggard, and R. N. Wright, "Open vs. Closed Systems for Accountability," in *Proceedings of the 2014 Symposium and Bootcamp on the Science of Security, HotSoS 2014, Raleigh, NC, USA, April 08 - 09, 2014*. ACM, 2014, p. 4.

[26] R. Küsters, T. Truderung, and A. Vogt, "Accountability: Definition and Relationship to Verifiability," in *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2010)*. ACM, 2010, pp. 526–535, the full version is available at http://eprint.iacr.org/2010/236.

[27] M. Graf, R. Küsters, and D. Rausch, "AUC: Accountable Universal Composability," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 1148–1167.

[28] R. Künnemann, I. Esiyok, and M. Backes, "Automated Verification of Accountability in Security Protocols," in *32nd IEEE Computer Security Foundations Symposium, CSF 2019, Hoboken, NJ, USA, June 25-28, 2019*. IEEE, 2019, pp. 397–413.

[29] N. Asokan, V. Shoup, and M. Waidner, "Asynchronous Protocols for Optimistic Fair Exchange," in *Proceedings of the IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society, 1998, pp. 86–99.

[30] C. Chang and Y. Chang, "Efficient Anonymous Auction Protocols with Freewheeling Bids," *Comput. Secur.*, vol. 22, no. 8, pp. 728–734, 2003.

[31] Z. Zhao, M. Naseri, and Y. Zheng, "Secure Quantum Sealed-bid Auction with Post-confirmation," *Optics Communications*, vol. 283, no. 16, pp. 3194–3197, 2010.

[32] C. Hong, J. Katz, V. Kolesnikov, W. Lu, and X. Wang, "Covert Security with Public Verifiability: Faster, Leaner, and Simpler," in *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, ser. Lecture Notes in Computer Science, vol. 11478. Springer, 2019, pp. 97–121.

[33] Y. Aumann and Y. Lindell, "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries," in *Proceedings of the 4th Theory of Cryptography Conference,(TCC 2007)*, ser. Lecture Notes in Computer Science, S. P. Vadhan, Ed., vol. 4392. Springer, 2007, pp. 137–156.

[34] G. Asharov and C. Orlandi, "Calling Out Cheaters: Covert Security with Public Verifiability," in *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, ser. Lecture Notes in Computer Science, vol. 7658. Springer, 2012, pp. 681–698.

[35] C. Baum, I. Damgård, and C. Orlandi, "Publicly Auditable Secure Multi-Party Computation," in *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, ser. Lecture Notes in Computer Science, vol. 8642. Springer, 2014, pp. 175–196.

[36] M. Rivinius, P. Reisert, D. Rausch, and R. Küsters, "Publicly Accountable Robust Multi-Party Computation," in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 2430–2449.

[37] M. Rivinius, "MPC with publicly identifiable abort from pseudorandomness and homomorphic encryption," in *Eurocrypt 2025*, 2025, (To appear).

[38] H. Leibowitz, A. Herzberg, and E. Syta, "Provable Security for PKI Schemes," Cryptology ePrint Archive, Tech. Rep. 2019/807, 2019.

[39] S. Matsumoto and R. M. Reischuk, "Certificates-as-an-Insurance: Incentivizing accountability in SSL/TLS," in *Proceedings of the NDSS Workshop on Security of Emerging Network Technologies (SENT'15)*, 2015.

[40] T. H. Kim, L. Huang, A. Perrig, C. Jackson, and V. D. Gligor, "Accountable Key Infrastructure (AKI): a Proposal for a Public-key Validation Infrastructure," in *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 679–690.

[41] M. Graf, R. Küsters, and D. Rausch, "Accountability in a Permissioned Blockchain: Formal Analysis of Hyperledger Fabric," in *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*. Los Alamitos, CA, USA: IEEE, 2020, pp. 236–255.

[42] G. O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, and S. Capkun, "Misbehavior in Bitcoin: A Study of Double-Spending and Accountability," *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 1, pp. 2:1–2:32, 2015.

[43] V. Buterin and V. Griffith, "Casper the Friendly Finality Gadget," *CoRR*, vol. abs/1710.09437, 2017.

[44] R3, "R3 Corda Master Documentation," https://docs.corda.net/docs/corda-os/4.4.html, 2020, (Accessed on 04/24/2020).

[45] Ethereum Foundation, "Ethereum Enterprise," https://www.ethereum.org/enterprise/, 2019, (Accessed on 11/13/2019).

[46] A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey, "Towards Better Availability and Accountability for IoT Updates by Means of a Blockchain," in *2017 IEEE European*

Symposium on Security and Privacy Workshops, EuroS&P Workshops 2017, Paris, France, April 26-28, 2017. IEEE, 2017, pp. 50–58.

[47] G. D'Angelo, S. Ferretti, and M. Marzolla, "A Blockchain-based Flight Data Recorder for Cloud Accountability," in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, CRYBLOCK@MobiSys 2018, Munich, Germany, June 15, 2018*. ACM, 2018, pp. 93–98.

[48] E. Funk, J. Riddell, F. Ankel, and D. Cabrera, "Blockchain technology: a data framework to improve validity, trust, and accountability of information exchange in health professions education," *Academic Medicine*, vol. 93, no. 12, pp. 1791–1794, 2018.

[49] A. Shamis, P. Pietzuch, B. Canakci, M. Castro, C. Fournet, E. Ashton, A. Chamayou, S. Clebsch, A. Delignat-Lavaud, M. Kerner *et al.*, "IA-CCF: Individual Accountability for Permissioned Ledgers," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 467–491.

[50] R. Küsters, J. Liedtke, J. Müller, D. Rausch, and A. Vogt, "Ordinos: A Verifiable Tally-Hiding E-Voting System," in *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*. IEEE, 2020, pp. 216–235.

[51] C. C. Dragan, F. Dupressoir, K. Gjøsteen, T. Haines, P. B. Rønne, and M. R. Solberg, "Machine-Checked Proofs of Accountability: How to sElect Who is to Blame," in *Computer Security - ESORICS 2023 - 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25-29, 2023, Proceedings, Part III*, ser. Lecture Notes in Computer Science, vol. 14346. Springer, 2023, pp. 471–491.

[52] J. Heather and D. Lundin, "The Append-Only Web Bulletin Board," in *Formal Aspects in Security and Trust, 5th International Workshop, FAST 2008, Malaga, Spain, October 9-10, 2008, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 5491. Springer, 2008, pp. 242–256.

[53] V. Cortier and B. Smyth, "Attacking and fixing Helios: An analysis of ballot secrecy," *Journal of Computer Security*, vol. 21, no. 1, pp. 89–148, 2013.

[54] R. Küsters, T. Truderung, and A. Vogt, "Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study," in *32nd IEEE Symposium on Security and Privacy (S&P 2011)*. IEEE Computer Society, 2011, pp. 538–553.

[55] R. Küsters, J. Müller, E. Scapin, and T. Truderung, "sElect: A Lightweight Verifiable Remote Voting System," in *IEEE 29th Computer Security Foundations Symposium (CSF 2016)*. IEEE Computer Society, 2016, pp. 341–354.

[56] N. Huber, R. Küsters, T. Krips, J. Liedtke, J. Müller, D. Rausch, P. Reisert, and A. Vogt, "Kryvos: Publicly tally-hiding verifiable e-voting," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 1443–1457. [Online]. Available: https://doi.org/10.1145/3548606.3560701

[57] R. Küsters, T. Truderung, and A. Vogt, "Formal Analysis of Chaumian Mix Nets with Randomized Partial Checking," in *35th IEEE Symposium on Security and Privacy (S&P 2014)*. IEEE Computer Society, 2014, pp. 343–358.

[58] R. Küsters and T. Truderung, "Security Analysis of Re-Encryption RPC Mix Nets," in *IEEE 1st European Symposium on Security and Privacy (EuroS&P 2016)*. IEEE Computer Society, 2016, pp. 227–242.

[59] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. 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*. ACM, 2016, pp. 1614–1625.

[60] S. Delaune, S. Kremer, and M. Ryan, "Coercion-Resistance and Receipt-Freeness in Electronic Voting," in *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW'06)*. IEEE Computer Society Press, 2006, pp. 28–39.

[61] D. Rausch, N. Huber, and R. Küsters, "Verifiable E-Voting with a Trustless Bulletin Board," Cryptology ePrint Archive, Tech. Rep. 841, 2025, full version of this paper, available at https://eprint.iacr.org/2025/841.

[62] R. Canetti, "Universally Composable Security," *J. ACM*, vol. 67, no. 5, pp. 28:1–28:94, 2020.

[63] R. Küsters, M. Tuengerthal, and D. Rausch, "The IITM Model: a Simple and Expressive Model for Universal Composability," *Journal of Cryptology*, vol. 33, no. 4, pp. 1461–1584, 2020.

[64] J. Camenisch, S. Krenn, R. Küsters, and D. Rausch, "iUC: Flexible Universal Composability Made Simple," in *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, ser. Lecture Notes in Computer Science, vol. 11923. Springer, 2019, pp. 191–221, the full version is available at http://eprint.iacr.org/2019/1073.

[65] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," in *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*. ACM, 2018, pp. 30:1–30:15.

[66] S. Kremer and P. B. Rønne, "To Du or Not to Du: A Security Analysis of Du-Vote," in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*. IEEE, 2016, pp. 473–486.

[67] V. Cortier and B. Smyth, "Attacking and Fixing Helios: An Analysis of Ballot Secrecy," in *Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF, 2011*, 2011, pp. 297–311.

[68] D. Mestel, J. Müller, and P. Reisert, "How efficient are replay attacks against vote privacy? A formal quantitative analysis," *J. Comput. Secur.*, vol. 31, no. 5, pp. 421–467, 2023.

Appendix A
Supplementary Figures

Here we provide supplementary Figures 1 to 3.

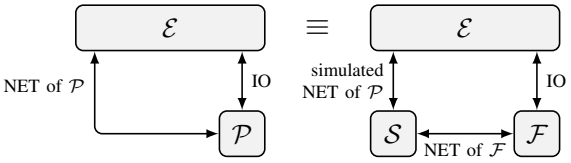

Fig. 1. UC security definition. Real world/protocol on the left, ideal world/protocol on the right. "IO" denotes input/output interfaces provided to higher-level protocols, "NET" are interfaces for sending/receiving network messages, ≡ is computational indistinguishability of the output of $\mathcal{E}$.

Appendix B
Further Insights Related to Privacy

Equivocation Attacks for Threshold Encryption. Consider a homomorphic aggregation-based voting system that uses a threshold $t < n_{\mathsf{talliers}}$ for decryption (unlike ABBOVE, which uses full threshold, i.e., $t = n_{\mathsf{talliers}}$). This is a common technique to improve the robustness of the tallying phase. In such systems, a malicious BB can run the following type of attack using equivocation.

If $t \leq \frac{n_{\mathsf{talliers}}}{2}$, the malicious BB controlled by an adversary $\mathcal{A}$ can split the talliers into two or more distinct groups. It can show each group a different set of ballots, say, one set that contains all submitted ballots and one set that contains all ballots except for the one from $\mathsf{v}_{\mathsf{obs}}$ that $\mathcal{A}$ wants to learn. As both sets of ballots appear reasonable to the respective talliers, both groups would compute and publish the election result based on the respective sets of ballots they see. Hence, the adversary controlling the BB learns (at least) two election results (rather than only one as in an ideal voting protocol) and can even derive the vote of $\mathsf{v}_{\mathsf{obs}}$ directly from both results.

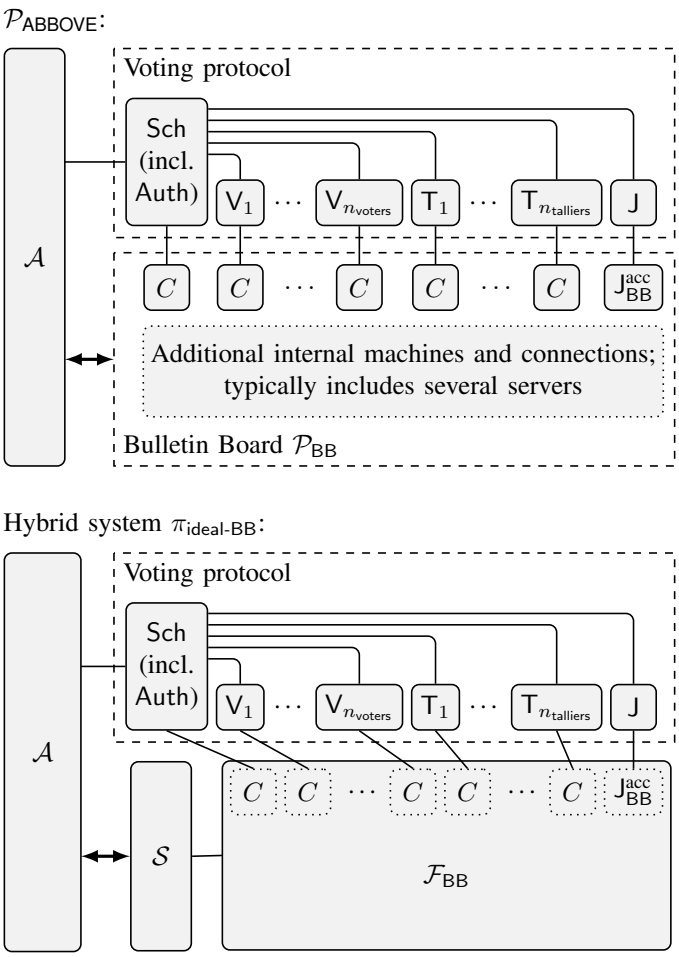

Fig. 2. Top: Illustration of the computational model of our protocol $\mathcal{P}_{\mathsf{ABBOVE}}$ (see Section V-A) running with an adversary $\mathcal{A}$. Bottom: Hybrid system used in the proof of Theorem 2 where the real BB $\mathcal{P}_{\mathsf{BB}}$ is replaced by the ideal BB $\mathcal{F}_{\mathsf{BB}}$ running with a simulator $\mathcal{S}$.

This type of attack is not limited to $t \leq \frac{n_{\mathsf{talliers}}}{2}$ but works for any threshold $t < n_{\mathsf{talliers}}$ if the adversary, in addition to controlling the BB, also controls (up to) $t-1$ malicious talliers. As a result, even if $t = n_{\mathsf{talliers}} - 1$ (such that there are $t-1 = n_{\mathsf{talliers}} - 2$ malicious and 2 honest talliers), the BB might show each honest tallier a different set of ballots to learn one partial decryption for each of them. The malicious talliers can then compute the remaining partial decryptions for both sets.

We note that this is not an impossibility result. It rather demonstrates that, if e-voting protocols using a threshold $t < n_{\mathsf{talliers}}$ are designed based on a real BB that can equivocate, then the protocol has to consider and protect against additional types of attacks on privacy.

A Gap in the Privacy Proof in [11]. While analyzing the privacy of ABBOVE and checking whether we can re-use some arguments from privacy proofs of similar homomorphic-aggregation-based systems, we found a gap in the privacy proof given for Helios in [11].

Specifically, as part of their proof in [11], the authors perform a reduction to (a variant of) the CPA-security of

$\underline{\mathcal{P}_{\mathsf{ideal}}(n_{\mathsf{voters}}, n^{\mathsf{hon}}_{\mathsf{voters}}, \mathsf{C}, \mu_{\mathrm{vote}})[\mathbf{v}_b]}$

Parameters:

- Probability distribution μ_{vote} over C
- Numbers of voters n_{voters} and honest voters $n^{\mathsf{hon}}_{\mathsf{voters}}$
- $I \leftarrow \emptyset$ (initially)

Inputs:

- Choice $\mathbf{v}_b$ for the voter under observation $\mathsf{v}_{\mathsf{obs}}$.

$\mathcal{P}_{\mathsf{ideal}}$ uses a scheduler Sch which forwards messages to and from $\mathcal{A}_{\mathsf{ideal}}$ while taking care that the following messages are sent sequentially in the order presented here. It also ensures that the first and last message ((init, honest) and compute) are sent at most once each:

On $(\mathsf{init}, \mathsf{honest})$ from $\mathcal{A}_{\mathsf{ideal}}$ do:

1) $\forall i \in \{1, \ldots, n^{\mathsf{hon}}_{\mathsf{voters}} - 1\}$: store $\mathbf{v}_i \xleftarrow{\mu_{\mathrm{vote}}} \mathsf{C}$
2) Store $\mathbf{v}_{n^{\mathsf{hon}}_{\mathsf{voters}}} \leftarrow \mathbf{v}_b$
3) $I \leftarrow I \cup \{1, \ldots, n^{\mathsf{hon}}_{\mathsf{voters}}\}$
4) Return success (to the adversary $\mathcal{A}_{\mathsf{ideal}}$).

On $(\mathsf{setChoice}, i, \mathbf{v})$ from $\mathcal{A}_{\mathsf{ideal}}$ do:

1) If $i \notin \{n^{\mathsf{hon}}_{\mathsf{voters}} + 1, \ldots, n_{\mathsf{voters}}\}$ or $\mathbf{v} \notin \mathsf{C}$, return $\perp$.
2) Store $\mathbf{v}_i \leftarrow \mathbf{v}$
3) $I \leftarrow I \cup \{i\}$
4) Return success (to the adversary $\mathcal{A}_{\mathsf{ideal}}$).

On compute from $\mathcal{A}_{\mathsf{ideal}}$ do:

1) Return res $\leftarrow \left(\sum_{i \in [n_{\mathsf{voters}}], \mathbf{v}_i = j} 1\right)_{j \in [n_{\mathsf{cand}}]}$ (to the adversary $\mathcal{A}_{\mathsf{ideal}}$).

Fig. 3. The ideal voting protocol $\mathcal{P}_{\mathsf{ideal}}$ for choice space C, number n_{voters} of voters, $n^{\mathsf{hon}}_{\mathsf{voters}} \leq n_{\mathsf{voters}}$ of which are honest voters, and vote distribution μ_{vote}.

ElGamal encryption under the assumption that there is at least one honest tallier $\mathsf{T}_{\mathsf{hon}}$. Intuitively, in the reduction, $\mathsf{T}_{\mathsf{hon}}$ computes its public key share based on the public key shares of all other talliers in a way that forces the overall public encryption key of the election to be the same as the public key in the CPA game. While this works if $\mathsf{T}_{\mathsf{hon}}$ happens to be the last tallier to publish its share, this is not guaranteed. If a malicious tallier waits until all other talliers have published their public key shares and then chooses its share depending on those published shares, say, by hashing all of them and using the result as the secret exponent, the reduction fails.

Fortunately, we found that a different reduction technique from [3] given for the threshold ElGamal version used by Belenios can be adapted to the full threshold ElGamal version used by Helios and ABBOVE to show privacy, as described in the full version [61]. This reduction is more complicated than the one given in [11] but does not require $\mathsf{T}_{\mathsf{hon}}$ to compute its share based on shares of other talliers. Hence, it works irrespective of the order in which talliers publish shares.