# Cryptographic Security Analysis of E-voting Systems: Achievements, Misconceptions, and Limitations

Ralf Küsters[1(✉)] and Johannes Müller[2]

[1] University of Stuttgart, Stuttgart, Germany
ralf.kuesters@sec.uni-stuttgart.de
[2] University of Trier, Trier, Germany
muellerjoh@uni-trier.de

**Abstract.** Rigorous cryptographic security analysis plays an important role in the design of modern e-voting systems by now. There has been huge progress in this field in the last decade or so in terms of formalizing security requirements and formally analyzing e-voting systems. This paper summarizes some of the achievements and lessons learned, which, among others, challenge common believes about the role of and the relationships between central security requirements.

## 1 Introduction

Privacy, verifiability, accountability, and coercion-resistance are fundamental security requirements for modern e-voting systems. *Privacy* ensures that the way a particular voter voted is not revealed to anybody. Intuitively, *verifiability* guarantees that it is possible to verify that the published election result is correct, even if voting machines/authorities are (partially) untrusted. In the literature, one often finds that verifiability is divided into *individual* and *universal verifiability*. *Accountability* is a stronger form of verifiability: accountability does not only require that it is detected if the published result is incorrect, but that misbehaving parties can be singled out and thus held accountable. This notion so far has gained much less attention than verifiability, although rather than aiming for mere verifiability, modern e-voting system should really strive for accountability in order to be useful in practice, as later explained and further emphasized in this paper. *Coercion-resistance* protects voters against vote buying and coercion. A weaker form of coercion-resistance is called *receipt-freeness*.

In order to find out whether a given voting system achieves its desired security properties, informally analyzing its security is not sufficient since critical aspects can easily be overlooked. Therefore, it is necessary to formally analyze the security of voting systems based on reasonable and formal security definitions.

There have been major achievements in the field of rigorous cryptographic analysis of e-voting systems in the last decade or so. Formal definitions for the


R. Krimmer et al. (Eds.): E-Vote-ID 2017, LNCS 10615, pp. 21–41, 2017.
DOI: 10.1007/978-3-319-68687-5_2

central security requirements have been proposed and intensively been studied (see, e.g., [5,12,31,32,35]). Some of these definitions are formulated in general and widely applicable frameworks so that they can be applied to virtually any e-voting protocols. These frameworks and definitions have been applied to perform rigorous security analysis of various existing e-voting systems (see, e.g., [2,10,13,14,29,31,32,34,35,37,38]), often with surprising results, and newly proposed systems more and more come with security proofs right away (see, e.g., [7,25–28]).

The rigorous approach also has helped to reveal some confusions and common misconceptions concerning security requirements and their relationships, and by this aided the deeper understanding of such requirements, providing a solid formal basis for the design and analysis of e-voting systems.

In this paper, some of these confusions and misconceptions will be highlighted and explained. In particular, based on various works from the literature, we point out the following:

- The still popular notions of individual and universal verifiability together are neither sufficient nor necessary to achieve end-to-end (E2E) verifiability, as explained in Sect. 2.
- E2E verifiability alone is typically insufficient for practical purposes. E-voting systems should really be designed with accountability in mind, a notion presented in Sect. 3.
- While it is commonly believed that coercion-resistance implies privacy, surprisingly, this is not true in general. Moreover, improving the level of privacy can lead to a lower level of coercion resistance (see Sect. 4).

Throughout the paper, we also emphasize the importance of widely applicable security definitions. The definitions which we recall in this paper are all based on a common general framework where systems and protocols are formulated as sets of interactive probabilistic polynomial time Turing machines (see Sect. 2.1). By this, virtually any e-voting system can be modeled in such a framework. All definitions presented here are cryptographic game-based definitions. The definitions also allow one to measure the level of security an e-voting system provides. This is crucial as security typically is not perfect, since, for example, only a fraction of voters perform certain checks.

Before we conclude in Sect. 6, we briefly discuss limitations of the cryptographic analysis of e-voting systems in Sect. 5, such as usability aspects, legal requirements, implementation and deployment issues.

## 2 Verifiability

E-voting systems are complex hardware/software systems. In such systems, as in all complex systems, it is almost impossible to avoid programming errors. Even worse, components of e-voting systems, such as voters' devices, voting machines, and voting servers, might have deliberately been tampered with. In fact, it has been demonstrated that numerous e-voting systems suffer from flaws that make it

possible for more or less sophisticated attackers to change the election result (see, e.g., [19,49,51,52]). Such manipulations are often hard or virtually impossible to detect. In some occasions, announced results were incorrect and/or elections had to be rerun (see, e.g., [23]).

Therefore, besides vote privacy, modern e-voting systems strive for what is called *verifiability*, more precisely end-to-end (E2E) verifiability. Roughly speaking, E2E verifiability means that voters and possibly external auditors should be able to check whether the published election result is correct, i.e., corresponds to the votes cast by the voters, even if voting devices and servers have programming errors or are outright malicious.

In the remainder of this section, we first recapitulate the notion of E2E verifiability and its formal definition. We then discuss other notions of verifiability, in particular the prominent notions of individual and universal verifiability. Following [28,35,37], we show that, unlike commonly believed, these two notions fail to provide a solid basis for verifiability. In particular, they are neither necessary nor sufficient to achieve E2E verifiability.

### 2.1 E2E Verifiability

About 30 years ago, Benaloh already provided a first definition of E2E verifiability [4]. As discussed in [12], while Benaloh's definition is fairly simple and captures the essence of verifiability, it requires unrealistically strong properties so that it would reject even reasonable e-voting systems.

In [32], Küsters, Truderung, and Vogt introduced a generic framework (the *KTV framework*) for verifiability and, more precisely, the even stronger notion of accountability (see Sect. 3). They also instantiated the framework to define E2E verifiability; also called *global verifiability* in [32], in contrast to individual and universal verifiability (see Sect. 2.2). This framework and definition since then have been used to analyze several e-voting protocols and mix nets [28,29,32,35,37,38], such as Helios, ThreeBallot, VAV, Wombat Voting, sElect, Chaumian RPC mix nets, and re-encryption RPC mix nets. It can also be applied to other domains, such as auctions and contract signing [32].

Cortier et al. [12] demonstrated that it is possible to cast all formal verifiability definitions from the literature into the generic KTV framework (see also below).

*E2E Verifiability in Short.* In short, Küsters et al. capture E2E verifiability in the KTV framework as follows: The probability that a run is accepted (by a judge or other observers), but the published result of the election does not correspond to the actual votes cast by the voters is small (bounded by some parameter $\delta$). More specifically, the result should contain all votes of the honest voters, except for at most $k$ honest votes (for some parameter $k \geq 0$), and it should contain at most one vote for every dishonest voter.

In what follows, we first briefly recall the generic KTV framework and then its instantiation which captures E2E verifiability (see [32] for details or the presentation of this framework in [12]). In [32], formalizations both in a symbolic as

well as a computational model were presented. Here, as throughout the paper, we concentrate on the computational model.

*Protocol Model of the KTV Framework.* A protocol is simply modeled as a set of probabilistic polynomial-time interactive Turing machines (ITMs) where the ITMs are connected via named tapes. We also refer to such a set as a *process*. By this, arbitrary protocols can be modeled.

More specifically, a *protocol* $P$ is defined by a set of agents/parties $\Sigma$ and an ITM $\pi_{\mathsf{a}}$ for each agent $\mathsf{a}$ in $\Sigma$. The set $\Sigma$ may contain voters, voting devices, bulletin board(s), various tellers, auditors, etc. Note that one can easily model voters and voting devices as separate entities (ITMs) in this framework. The program $\pi_{\mathsf{a}}$ is called the *honest program* of $\mathsf{a}$. By $\pi_P$ we denote the process consisting of all of these (connected) ITMs. This process is always run with an adversary $\mathsf{A}$ which may run an arbitrary (probabilistic polynomial-time) program $\pi_{\mathsf{A}}$ and which is connected to all other parties. The adversary can model the network and/or dishonest parties. Also, $\mathsf{A}$ may statically or dynamically corrupt parties (by sending a corrupt messages to these parties); parties who should not be corruptable would simply ignore corruption messages by the adversary. A *run of* $P$ with adversary $\pi_{\mathsf{A}}$ is a run of the process $\pi_P \| \pi_{\mathsf{A}}$ (the union of the ITMs in $\pi_P$ and the ITM $\pi_{\mathsf{A}}$).

*A Generic Verifiability Definition in the KTV Framework.* The KTV framework provides a general definition of verifiability, which in particular can be instantiated to model E2E verifiability (see below). The definition assumes a judge $\mathsf{J}$ whose role is to accept or reject a protocol run by outputting $\mathsf{accept}$ or $\mathsf{reject}$ (on some tape). To make a decision, the judge runs a so-called *judging procedure*, which performs certain checks (depending on the protocol specification), such as verification of zero-knowledge proofs (if any) and taking voter complaints into account. Typically, the judging procedure can be carried out by any party, including external observers and even voters themselves, as the information to be checked is public. Hence, the judge might just be a "virtual" entity.

The generic KTV verifiability definition is centered around a *goal* $\gamma$ of the protocol. Formally, $\gamma$ is a set of protocol runs.[1] The goal $\gamma$ specifies those runs which are correct or desired in some protocol-specific sense. In the context of e-voting and for E2E verifiability, the goal would contain those runs where the announced election result corresponds to the actual choices of the voters.

Now, the idea behind the definition of verifiability in the KTV framework is very simple. Only those runs $r$ should be accepted by the judge in which the goal $\gamma$ is met, i.e., $r \in \gamma$. In the context of e-voting, if in a run the published result does not correspond to the actual choices of the voters, then the judge should reject the run. More precisely, the definition requires that for all adversaries the probability (over the set of all protocol runs) that a run is accepted by the judge but the goal is not met is bounded by some constant $\delta$ (plus a negligible function). Although $\delta = 0$ is desirable, this would be too strong for almost all e-voting protocols. For example, typically not all voters check whether their

[1] Note that a single run is determined by the random coins used by the parties involved in the run.

ballots appear on the bulletin board. This give the adversary the opportunity to manipulate or drop some votes without being detected. Therefore, $\delta = 0$ cannot be achieved in general. The parameter $\delta$ is called the *verifiability tolerance* of the protocol.

By $\Pr(\pi^{(\ell)} \mapsto \neg\gamma,\ (\mathsf{J}\colon \mathsf{accept}))$ we denote the probability that the process $\pi$, with security parameter $1^\ell$, produces a run which is not in $\gamma$ but nevertheless accepted by $\mathsf{J}$.

**Definition 1 (Verifiability).** *Let $P$ be a protocol with the set of agents $\Sigma$. Let $\delta \in [0,1]$ be the tolerance, $\mathsf{J} \in \Sigma$ be the judge, and $\gamma$ be a goal. Then, we say that* the protocol P is $(\gamma,\delta)$-verifiable by the judge $\mathsf{J}$ *if for all adversaries $\pi_\mathsf{A}$ and $\pi = (\pi_P \| \pi_\mathsf{A})$, the probability*

$$\Pr(\pi^{(\ell)} \mapsto \neg\gamma,\ (\mathsf{J}\colon \mathsf{accept}))$$

*is $\delta$-bounded*[2] *as a function of $\ell$.*

We note that the original definition in [32] also captures soundness/fairness: if the protocol runs with a benign adversary, which, in particular, would not corrupt parties, then the judge accepts all runs. This kinds of fairness/soundness can be considered to be a sanity check of the protocol, including the judging procedure, and is typically easy to check.

We note that Definition 1 does not (need to) assume any specific protocol structure, and hence, is widely applicable. It also takes into account real-world uncertainties. As mentioned before and shown in [12], all definitions of verifiability from the literature can be captured by appropriate choices of the goal $\gamma$. The specific protocol structures often assumed in such definitions can also easily be captured.

*E2E Verifiability in the KTV Framework.* In [32], Küsters et al. proposed an instantiation of the generic verifiability definition to capture E2E verifiability. To this end, they introduce a family of goals $\{\gamma_k\}_{k\geq 0}$:[3] the goal $\gamma_k$ contains exactly those runs of the voting protocol in which $(i)$ all but up to $k$ votes of the honest voters are counted correctly, and $(ii)$ every dishonest voter votes at most once (see the technical report [33] of [32] or [12] for the formal definition). For example, consider a run of an e-voting protocol with three honest voters and two dishonest voters. Assume that there are two candidates/choices $A$ and $B$, and that the tallying function returns the number of votes for each candidate. Now, if all honest voters vote for, say, $A$ and the final result is $(A,B) = (2,2)$, then $\gamma_k$ is achieved for all $k \geq 1$ but $\gamma_0$ is not achieved: one vote of an honest voter is missing (dropped or flipped to a vote for $B$), and there is at most one vote for every dishonest voter; $\gamma_0$ is not satisfied because it requires that all votes of honest voters are counted, which is not the case here.

With this definition of goals, Definition 1 captures E2E verifiability: the probability that the judge accepts a run where more than $k$ votes of honest voters

[2] Bounded by $\delta$, plus some negligible function in the security parameter $\ell$.

[3] In [12] (Subsect. 10.2), these goals have been refined.

were manipulated or dishonest voters could cast too many votes, is bounded by $\delta$. In security statements about concrete e-voting protocols, $\delta$ will typically depend on various parameters, such as $k$ and the probability that voters performs certain checks. While $k = 0$ is desirable, this is in most cases impossible to achieve because, for example, voters might not always perform the required checks, and hence, there is a chance that manipulation of votes goes undetected.

Importantly, this definition of E2E verifiability allows one to *measure* the level of E2E verifiability an e-voting protocol provides.

### 2.2 Individual and Universal Verifiability

Sako and Kilian [45] introduced the notions of *individual* and *universal verifiability*. These requirements (and subsequent notions, such as *cast-as-intended*, etc.) have become very popular and are still used to design and analyze e-voting systems. According to Sako and Kilian, an e-voting system achieves individual verifiability if "a sender can verify whether or not his message has reached its destination, but cannot determine if this is true for the other voters". Universal verifiability guarantees that it is possible to publicly verify that the tallying of the ballots is correct. That means that the final election result exactly reflects the content of those ballots that have been accepted to be tallied.

The notions of individual and universal verifiability have later been formalized by Chevallier-Mames et al. [8] (only universal verifiability), Cortier et al. [10], and Smyth et al. [48]. As mention in [32] and demonstrated in [12], these notions can also be captured in the KTV framework.

*A Common Misconception.* Unfortunately, it is often believed that individual together with universal verifiability implies E2E verifiability, which is the security property that e-voting systems should achieve. However, in [32,37], and [28], Küsters et al. have demonstrated that individual and universal verifiability are *neither sufficient nor necessary* for E2E verifiability.

In short, there are e-voting systems, such as ThreeBallot and VAV [42] as well as variants of Helios, that arguably provide individual and universal verifiability but whose verifiability is nevertheless broken, i.e., they do not provide E2E verifiability. Conversely, there are e-voting systems, such as sElect [28], which provide E2E verifiability without having to rely on universal verifiability.

In what follows, we explain these results in more detail.

### 2.3 Not Sufficient

We recall several attacks that break the E2E verifiability of e-voting systems, even though these systems provide individual and universal verifiability. The first class of attacks uses that (dishonest) voters possibly with the help of malicious authorities might cast malformed ballots. In the second class of attacks (so-called *clash attacks*), the same receipt is shown to different voters who voted for the same candidate, allowing malicious voting devices and authorities to drop or manipulate ballots.

*An Illustrative Example: A Modification of Helios.* Helios [1] is one of the most prominent remote e-voting systems which, on a high level, works as follows. Trustees share a secret key $sk$ which belongs to a public/private ElGamal key pair $(pk, sk)$. Voters encrypt the candidate of their choice under the public key $pk$ and submit the resulting ciphertext to the bulletin board. Then all ciphertexts are publicly multiplied so that, by the homomorphic property of the ElGamal public-key encryption scheme, the resulting ciphertext encrypts the number of votes for each candidate. Finally, the trustees perform distributed and verifiable decryption of this ciphertext and publish the resulting plaintext as the outcome of the election.

In order to guarantee the integrity of the final result, several zero-knowledge proofs (ZKP) are used. Among others, a voter has to prove that her ciphertext encrypts a valid choice, and, for privacy reasons, that she knows which choice it encrypts.

It has been formally proven that under certain assumptions Helios is E2E verifiable (see, [11,37]). Furthermore, assuming that the voting devices are honest, Helios provides individual verifiability because each voter can check whether her ballot appears on the bulletin board. Universal verifiability follows from the fact that the multiplication of the ciphertexts on the bulletin board is public and that the tellers perform verifiable decryption. Thus, Helios provides E2E verifiability as well as individual and universal verifiability.

To see that individual and universal verifiability together do not imply E2E verifiability consider a modification of Helios in which voters do not have to prove that their votes are correct, i.e., dishonest voters may cast malformed ballots without being detected. Then a (single!) dishonest voter could completely spoil the election result by encrypting an invalid choice. Such a malformed ballot might contain negative votes for certain candidates, and hence, effectively subtracting votes from candidates, or the malformed ballot might contain many more votes for a candidate then allowed. So, such a system certainly does not provide E2E verifiability. At the same time, such a system can still be considered to provide individual and universal verifiability. Voters can still check that their ballots appear on the bulletin board (individual verifiability), and ballots on the bulletin board can still be tallied in a universally verifiable way. But dishonest voters might have spoiled the election result completely and this is not detected.[4]

This simple example demonstrates that, even if an e-voting system achieves individual and universal verifiability, its overall verifiability can nevertheless completely and trivially be broken.

*Another Example: ThreeBallot.* The attack illustrated above conceptually also applies to the ThreeBallot voting system [42] (also to VAV), but the details of the attack differ. We start by briefly describing how ThreeBallot works.

In ThreeBallot, a voter is given a multi-ballot consisting of three simple ballots. On every simple ballot, the candidates, say $A$ and $B$, are printed in the same fixed order, say $A$ is listed first and $B$ is listed second. In the secrecy

[4] Note that the arguments hold true even when assuming that only eligible voters (honest or dishonest) may vote.

of a voting booth, the voter is supposed to fill out all three simple ballots in the following way: she marks the candidate of her choice on exactly *two* simple ballots and every other candidate on exactly *one* simple ballot. Assume, for example, that a voter votes for candidate $A$. Then

$$\binom{x}{o}, \binom{x}{o}, \binom{o}{x} \text{ or } \binom{x}{x}, \binom{o}{o}, \binom{x}{o}$$

would be valid multi-ballots to vote for $A$. After this, the voter feeds all three simple ballots to a voting machine (a scanner) and indicates the simple ballot she wants to get as a receipt. The machine checks the well-formedness of the multi-ballot, prints secretly (pairwise independent) random numbers on each simple ballot, and provides the voter with a copy of the chosen simple ballot, with the random number printed on it. Note that the voter does not get to see the random numbers of the remaining two simple ballots. The scanner keeps all simple ballots (now separated) in a ballot box.

In the tallying phase, the voting machine posts on the bulletin board (electronic copies of) all the cast simple ballots in random order. From the ballots shown on the bulletin board, the result can easily be computed: The number of votes for the $i$th candidate is the number of simple ballots with the $i$th position marked minus the total number of votes (since every voter marks every candidate at least ones).

ThreeBallot offers (some level of) individual verifiability because each voter may check whether the simple ballot she has taken as a receipt appears on the bulletin board. Thus, it should be risky for any party to remove or alter simple ballots. Additionally, ThreeBallot offers universal verifiability because the tallying is completely public. However, as Küsters et al. [35] have pointed out, ThreeBallot does not offer E2E verifiability. One variant of the attack presented in [35] assumes that the scanner is dishonest. To illustrate the attack, assume that an honest voter votes for, say, candidate $A$ by submitting a multi-ballot of one of the forms shown above. Now, a dishonest voter which collaborates with the dishonest scanner could create a malformed ballot of the form

$$\binom{o}{x}, \binom{o}{x}, \binom{o}{x},$$

which, together with the ballot of the honest voter (no matter which one of the two kinds shown above), yields two (valid!) votes for candidate $B$ and no vote for candidate $A$. Clearly, E2E verifiability is broken: a vote for $A$ and one invalid ballot result in two valid votes for $B$. But no honest voter would complain because none of their single/multi-ballots were manipulated. So, this attack neither invalidates individual verifiability nor universal verifiability, showing again that these notions together do not imply E2E verifiability, and are really insufficient.

*Clash Attacks.* The idea of individual and universal verifiability not only fails due to undetected malformed ballots. Another problem are *clash attacks* [37], which might break E2E verifiability, while individual and universal verifiability

together again do not detect such attacks. As demonstrated in [37], several e-voting system are vulnerable to clash attacks, including several variants of Helios.

To illustrate the attack, consider the Helios voting system, where the voting devices might be dishonest and where the ballots of the voters are published on the bulletin board without voter names or pseudonyms attached to them. Now, if two voters vote for the same candidate, the voting devices might use the same randomness to create the ballots, and hence, the two ballots are identical. However, instead of putting both ballots on the bulletin board, authorities might add only one of them to the bulletin board and the other ballot might be replaced by one for another candidate. The two voters can check individually that "their" ballot appears on the bulletin board (individual verifiability); they do not realize that they are looking at the same ballot, i.e., they do not realize the "clash". Universal verifiability is obviously guaranteed as well. Still, the system does not provide E2E verifiability: a vote of an honest voter was replaced in an undetectable way by another vote.

*Adding More Subproperties?* Now that we have seen that individual and universal verifiability do not imply the desired security property E2E verifiability, it might be tempting to search for more subproperties that would then, eventually, yield a sufficiently strong verifiability notion.

In [12], it has been demonstrated that all verifiability notions proposed in the literature so far that are split up into additional subproperties, such as individual and universal verifiability, do not provide E2E verifiability, even if more subproperties are added. In [10], for example, a subproperty was introduced that rules out clash attacks but the resulting verifiability notion is still too weak (see [12], Appendix B, for details).

When existing systems are analyzed w.r.t. verifiability or new systems are proposed, one should always check for E2E verifiability as introduced above, as E2E verifiability is the kind of verifiability modern e-voting systems ultimately should aim for. While subproperties, such as individual and universal verifiability, can guide the design of e-voting systems, unless formally proven that their combination in fact implies E2E verifiability, such properties alone might miss important aspects and can therefore not replace E2E verifiability.

### 2.4 Not Necessary

The examples and attacks above illustrate that the notions of individual and universal verifiability are not sufficient to provide E2E verifiability. Following [28], we now demonstrate that they are not necessary to achieve E2E verifiability either. More specifically, in [28] the remote e-voting system sElect was proposed, and it was shown that it provides E2E verifiability (under reasonable assumptions). But sElect is not universally verifiable.

*sElect.* sElect [28] is a conceptually simple remote voting system which is based on a Chaumian mix net.

A *Chaumian mix net* consists of mix servers $\mathsf{M}_1, \ldots, \mathsf{M}_n$ where each one of them holds a public/secret key pair $(pk_i, sk_i)$ of a (CCA2-)secure public-key encryption scheme. The input to the mix net is a set of ciphertexts $c_1, \ldots, c_l$ where each ciphertext $c_i$ is a nested encryption of a plaintext $m_i$ under the public keys of the mix servers in reverse order, i.e.,

$$c_i = \mathsf{Enc}\left(\ldots \mathsf{Enc}\left(m_i, pk_n\right) \ldots, pk_1\right).$$

When the mix net is executed, the first mix server decrypts the outer encryption layer with its secret key $sk_1$, shuffles the result,[5] and forwards it to the second mix server, which decrypts the next encryption layer with $sk_2$, shuffles the resulting ciphertexts, and so on. Finally, the output of the mix net is a random permutation $\pi$ of the input plaintexts $m_1, \ldots, m_l$. As long as one of the mix servers is honest, the permutation $\pi$ remains secret. That is, it is not possible to connect the input ciphertexts to their corresponding plaintexts.

Note that there are no ZKPs for correct shuffling or correct decryption, which means that Chaumian mix nets are not universally verifiable.

Now, roughly speaking, sElect works as follows. A voter uses her voting device (a browser) to select the candidate of her choice $m_i$. Then, the voting device creates a random nonce $n_i$ (which can be done jointly with the voter to decrease trust in the voting device). Afterwards, the device encrypts $(m_i, n_i)$ under the public keys of the mix servers as explained above. For verification purposes, the voter memorizes or writes down the nonce $n_i$. In addition, the voting device stores this information and the random coins that were used for encryption. In the tallying phase, all input ciphertexts are processed by the mix net as explained above, and the final result is, as well as all intermediate ciphertexts, published on the bulletin board. Each voter is finally invited to use her voting device in order to check whether her candidate $m_i$ appears next to her personal nonce $n_i$. In addition, the voting device performs a *fully automated verification procedure*. In particular, if the voter's vote and nonce do not appear together in the final result, the voting device can provably single out the mix servers that misbehaved because it has stored all information needed to follow the trace of the voter's ballot through the mix net (and because the mix servers signed certain information).

*E2E Verifiability Without Universal Verifiability.* It has been formally proven [28] that sElect provides a reasonable level of E2E verifiability (and even accountability) because it is extremely risky for an adversary to manipulate or drop even only a few votes. At the same time, sElect does not rely on universal verifiability. The Chaumian mix net is not verifiable by itself: it takes the voters to perform a simple check. Therefore, the example of sElect shows that universal verifiability is not necessary for E2E verifiability.

[5] In order to protect against replay attacks [13], duplicates are removed, keeping one copy only (see [28] for details.).

## 3 Accountability

In e-voting systems, and for many other cryptographic tasks and protocols (e.g., secure multi-party computation, identity-based encryption, and auctions), it is extremely important that (semi-)trusted parties can be held accountable in case they misbehave. This fundamental security property is called *accountability*,[6] and it is a stronger form of verifiability: it not only allows one to verify whether a desired property is guaranteed, for example that the election outcome is correct, but it also ensures that misbehaving parties can be identified if this is not the case.

Accountability is important for several practical reasons. First of all, accountability strengthens the incentive of all parties to follow their roles because they can be singled out in case they misbehave and then might have to face, for example, severe financial or legal penalties, or might lose their reputation. Furthermore, accountability can resolve disputes that occur when it is only known that some party misbehaved but not which one. This can, for instance, help to increase the robustness of cryptographic protocols because misbehaving parties, such as a dishonest trustee in an e-voting protocol, can be excluded and the protocol can be re-run without the parties that misbehaved.

Unfortunately, despite its importance, accountability is often not taken into account (at least not explicitly), neither to design e-voting protocols nor to analyze their security (see, e.g., [1,7,9,11,15,25–27,43,44]).

In [32], Küsters et al. provided a general formal definition of accountability and emphasized its importance. This formal definition has since been used to analyze different e-voting protocols (Helios, sElect, Bingo Voting), mix nets (re-encryption and Chaumian mix nets with random partial checking), auction schemes (PRST [41]), and contract signing protocols (ASW [3]). These analyses brought forward several accountability issues, e.g., for different versions of Helios [37]. In what follows, we give a brief summary of the accountability definition, for details see the original paper [32].

*A Formal Accountability Definition.* The accountability definition by Küsters et al. [32] is based on the same generic and expressive protocol model as the verifiability definition (see Sect. 2), and can therefore be applied to all classes of voting protocols and also to other domains.

In contrast to the verifiability definition, the judge now not only accepts or rejects a run, but may output detailed verdicts. A *verdict* is a positive Boolean formula $\psi$ built from propositions of the form $\mathsf{dis}(\mathsf{a})$, for an agent $\mathsf{a}$, where $\mathsf{dis}(\mathsf{a})$ means that (the judge thinks that) agent $\mathsf{a}$ misbehaved, i.e., did not follow the prescribed protocol. For example, in a voting protocol with voters $\mathsf{V}_1, \ldots, \mathsf{V}_n$, a bulletin board $\mathsf{B}$, and trustees $\mathsf{T}_1, \ldots, \mathsf{T}_m$, if the judge $\mathsf{J}$ states, say, $\mathsf{dis}(\mathsf{B}) \wedge \mathsf{dis}(\mathsf{T}_1) \wedge \ldots \wedge \mathsf{dis}(\mathsf{T}_m)$, then this expresses that the judge beliefs that the bulletin board and all trustees misbehaved; the judge would state $\mathsf{dis}(\mathsf{V}_i) \vee \mathsf{dis}(\mathsf{B}) \vee$

[6] In the context of secure MPC, accountability is sometimes called *identifiable abort* [22].

$(\mathsf{dis}(\mathsf{T}_1) \wedge \ldots \wedge \mathsf{dis}(\mathsf{T}_m))$ if she is not sure whether voter $\mathsf{V}_i$. the bulletin board, or all trustees misbehaved.

Who should be blamed in which situation is expressed by a set $\Psi$ of what are called *accountability constraints*. These constrains are of the form

$$C = \alpha \Rightarrow \psi_1 | \cdots | \psi_k,$$

where $\alpha$ is a property of the voting system, similar to the goal $\gamma$ in Sect. 2.1 (a set of runs of the system, where one run is determined by the random coins used by the parties), and $\psi_1, \ldots, \psi_k$ are verdicts. Intuitively, the set $\alpha$ contains runs in which some desired goal $\gamma$ of the protocol is not met (due to the misbehavior of some protocol participant). The formulas $\psi_1, \ldots, \psi_k$ are the possible minimal verdicts that are supposed to be stated by $\mathsf{J}$ in such a case; $\mathsf{J}$ is free to state stronger verdicts (by the fairness condition these verdicts will be true). That is, if a run belongs to $\alpha$, then $C$ requires that in this run the judge outputs a verdict $\psi$ which logically implies one of $\psi_i$.

To illustrate the notion of accountability constraints, let us continue the example from above. Let $\alpha$ contain all runs in which the published election result is incorrect, e.g., $\alpha = \alpha_k = \neg\gamma_k$ with the goal $\gamma_k$ as defined in Sect. 2. Now, consider the following constraints:

$$C_1 = \alpha \Rightarrow \mathsf{dis}(\mathsf{B}) | \mathsf{dis}(\mathsf{T}_1) | \cdots | \mathsf{dis}(\mathsf{T}_m), \tag{1}$$

$$C_2 = \alpha \Rightarrow \mathsf{dis}(\mathsf{V}_1) \vee \cdots \vee \mathsf{dis}(\mathsf{V}_n) \vee \mathsf{dis}(\mathsf{B}) \vee (\mathsf{dis}(\mathsf{T}_1) \wedge \cdots \wedge \mathsf{dis}(\mathsf{T}_m)), \tag{2}$$

$$C_3 = \alpha \Rightarrow \mathsf{dis}(\mathsf{B}) | \mathsf{dis}(\mathsf{T}_1) \wedge \cdots \wedge \mathsf{dis}(\mathsf{T}_m). \tag{3}$$

Constraint $C_1$ requires that if in a run the published election result is incorrect, then at least one (individual) party among $\mathsf{B}, \mathsf{T}_1, \ldots, \mathsf{T}_m$ can be held accountable by the judge $\mathsf{J}$; note that different parties can be blamed in different runs. Constraint $C_2$ states that if the published election result is not correct, then the judge $\mathsf{J}$ can leave it open whether one of the voters, the bulletin board $\mathsf{B}$, or all trustees misbehaved. Constraint $C_3$ requires that it is possible to hold $\mathsf{B}$ or all trustees accountable.

As pointed out in [32], accountability constraints should provide at least *individual accountability*. That is, the postulated minimal verdicts should at least single out one misbehaving party. In the above example, $C_1$ and $C_3$ provide individual accountability, but $C_2$ does not. In fact, $C_2$ is very weak, too weak for practical purposes. If a judge states exactly this verdict, there are no real consequences for any party, since no individual party can be held accountable. This is particular problematic if in such a "fuzzy" verdict not only voting authorities are involved but also voters.

A set $\Phi$ of constraints for a protocol $P$ is called an *accountability property* of $P$. Typically, an accountability property $\Phi$ covers all relevant cases in which a desired goal $\gamma$ for $P$ is not met, i.e., whenever $\gamma$ is not satisfied in a given run $r$ due to some misbehavior of some protocol participant, then there exists a constraint $C$ in $\Phi$ which covers $r$. We write $\Pr(\pi^{(\ell)} \rightarrow \neg(\mathsf{J}\colon \Phi))$ to denote the probability that $\pi$, with security parameter $1^\ell$, produces a run $r$ such that

J does not satisfies all accountability constrains for this run, i.e., there exists $C = \alpha \Rightarrow \psi_1 | \cdots | \psi_k$ with $r \in \alpha$ but the judge outputs a verdict which does not imply some $\psi_i$.

**Definition 2 (Accountability).** *Let $P$ be a protocol with the set of agents $\Sigma$. Let $\delta \in [0,1]$ be the tolerance, $\mathsf{J} \in \Sigma$ be the judge, and $\Phi$ be an accountability property of $P$. Then, we say that* the protocol P is $(\Phi, \delta)$-accountable by the judge J *if for all adversaries $\pi_\mathsf{A}$ and $\pi = (\pi_P \| \pi_\mathsf{A})$, the probability*

$$\Pr\left(\pi^{(\ell)} \rightarrow \neg(\mathsf{J} \colon \Phi)\right)$$

*is $\delta$-bounded as a function of $\ell$.*

Just as for the verifiability definition (Definition 1), the full definition in [32] additionally requires that the judge J is fair, i.e., that she states false verdicts only with negligible probability.

Küsters et al. also showed that verifiability (as defined in Definition 1) can be considered to be a weak form of accountability, and, as mentioned before, verifiability alone is typically too weak for practical purposes.

Instead of explicitly specifying $\Psi$ as necessary in the above definition, there have been attempts to find generic ways to define who actually caused a goal to fail and ideally to blame all of these parties. There has been work pointing into this direction (see, e.g., [16,20,21]). But this problem turns out to be very tricky and has not been solved yet.

## 4 Coercion-Resistance and Privacy

To achieve verifiability, a voter typically obtains some kind of receipt which, together with additional data published in the election, she can use to check that her vote was counted. This, however, potentially opens up the possibility for vote buying and voter coercion. Besides verifiability, many voting systems therefore also intend to provide so-called *coercion-resistance*.

One would expect that privacy and coercion-resistance are closely related: If the level of privacy is low, i.e., there is a good chance of correctly determining how a voter voted, then this should give the coercer leverage to coerce a voter. Some works in the literature (e.g., [17,39]) indeed suggest a close connection. However, Küsters et al. [35] demonstrated that the relationship between privacy and coercion-resistance is more subtle.

Among others, it turns out that improving the level of privacy of a protocol in a natural way (e.g., by changing the way honest voters fill out ballots) can lead to a *lower* level of coercion-resistance. Clearly, in general, one does not expect privacy to imply coercion-resistance. Still, the effect is quite surprising.

A maybe even more important and unexpected finding that comes out of the case studies in [35] is that the level of privacy of a protocol can be much *lower* than its level of coercion-resistance. The reason behind this phenomenon is basically that it may happen that the counter-strategy a coerced voter may

carry out to defend against coercion hides the behavior of the coerced voter, including her vote, better than the honest voting program.

On the positive side, in [35] Küsters et al. proved a theorem which states that under a certain additional assumption a coercion-resistant protocol provides at least the same level of privacy. This is the case when the counter-strategy does not "outperform" the honest voting program in the above sense. The theorem is applicable to a broad class of voting protocols.

In what follows, we explain the subtle relationships between coercion-resistance and privacy in more detail. The findings are based on formal privacy and coercion-resistance definitions proposed in [35] and [31,36], respectively. These definitions build upon the same general protocol model as the one for verifiability, and hence, they are applicable to all classes of voting systems (see, e.g., [28,31,32,34–37]), and they also have been applied to analyze mix nets [29,38]. We only informally introduce the privacy and coercion-resistance definitions in what follows and point to the reader to [31,35,36] for the formal definitions.

Intuitively, the privacy definition in [35] says that no (probabilistic polynomial-time) observer, who may control some parties, such as some authorities or voters, should be able to tell how an honest voter, the voter under observation, voted. More specifically, one considers two systems: in one system the voter under consideration votes for candidate $c$ and in the other system the voter votes for candidate $c'$; all other honest voters vote according to some probability distribution known by the observer. Now, the probability that the observer correctly says with which system he interacts should be bounded by some constant $\delta$ (plus some negligible function in the security parameter). Due to the parameter $\delta$, the definition allows one to *measure* privacy. As discussed in [28], this ability is crucial in the analysis of protocols which provide a reasonable but not perfect level of privacy. In fact, strictly speaking, most remote e-voting protocols do not provide a perfect level of privacy: this is because there is always a certain probability that voters do not check their receipts. Hence, the probability that malicious servers/authorities drop or manipulate votes without being detected is non-negligible. By dropping or manipulating votes, an adversaries obtains some non-negligible advantage in breaking privacy. Therefore, it is essential to be able to precisely tell how much an adversary can actually learn.

For the definition of coercion-resistance (see [31,36]), the voter under observation considered for privacy is now replaced by a *coerced voter* and the observer $\mathsf{O}$ is replaced by the *coercer* $\mathsf{C}$. We imagine that the coercer demands full control over the voting interface of the coerced users, i.e., the coercer wants the coerced voter to run a dummy strategy $\mathsf{dum}$ which simply forwards all messages between the coerced voter and the coercer $\mathsf{C}$. If the coerced voter in fact runs $\mathsf{dum}$, the coercer can effectively vote on behalf of the coerced voter or decide to abstain from voting. Of course, the coercer is not bound to follow the specified voting procedure. Now, informally speaking, a protocol is called *coercion-resistant* if the coerced voter, instead of running the dummy strategy, can run some *counter-strategy* $\mathsf{cs}$ such that $(i)$ by running this counter-strategy, the coerced voter achieves her own goal $\gamma$ (formally, again a set of runs), e.g., successfully votes

for a specific candidate, and (*ii*) the coercer is not able to distinguish whether the coerced voter followed his instructions (i.e., run dum) or tried to achieve her own goal (by running cs). Similarly to the privacy definition, the probability in (*ii*) is bounded by some constant $\delta$ (plus some negligible function). Again, $\delta$ is important in order to be able to measure the level of coercion-resistance a protocol provides: there is always a non-negligible chance for the coercer to know for sure whether the coerced voter followed his instructions or not (e.g., when all voteres voted for the same candidate).

*Improving Privacy Can Lower the Level of Coercion-Resistance.* To illustrate this phenomenon, we consider the following variant of ThreeBallot (for details of ThreeBallot see Sect. 2). An honest voter is supposed to submit, according to her favorite candidate,

$$\text{either } \binom{x}{x}, \binom{x}{o}, \binom{o}{o} \text{ or } \binom{x}{x}, \binom{o}{x}, \binom{o}{o},$$

and always take the first single ballot $\binom{x}{x}$ as her receipt. The scheme is ideal in terms of privacy because the bulletin board and the receipts do not leak any information apart from the pure election result. However, this scheme does not provide any coercion-resistance. Assume that the coerced voter is instructed to cast

$$\binom{o}{x}, \binom{x}{x}, \binom{o}{o}$$

and take the first single ballot as receipt (which is allowed but never done by honest voters). If the coerced voter actually wants to vote for candidate $A$, the voter would have to cast

$$\binom{o}{x}, \binom{x}{o}, \binom{x}{o}.$$

But then, as all the honest voters submit

$$\binom{x}{x}, \binom{x}{o}, \binom{o}{o} \text{ or } \binom{x}{x}, \binom{o}{x}, \binom{o}{o},$$

the coercer could easily detect that he was cheated, by counting the number of ballots of type $\binom{o}{o}$ on the bulletin board.

*Coercion-Resistance Does Not Imply Privacy.* For the original variant of ThreeBallot and the simple variant of VAV, Küsters et al. proved that the level of privacy is much lower than its level of coercion-resistance. The reason behind this phenomenon is basically that the counter-strategy hides the behavior of the coerced voter, including her vote, better than the honest voting program hides the vote. In these voting systems, a receipt an honest voter obtains indeed discloses more information than necessary (for details see [35]).

The following simple, but unlike ThreeBallot and VAV, artificial example, carries this effect to extremes: Consider the ideal voting protocol which collects

all votes and publishes the correct result. Now, imagine a voting protocol in which voters use the ideal voting protocol to cast their vote, but where half of the voters publish how they voted (e.g., based on a coin flip). Clearly, the privacy level this protocol provides is very low, namely $\delta \geq \frac{1}{2}$. However, a coerced voter can be more clever and simply lie about how she voted. This protocol indeed provides a high level of coercion-resistance.

As mentioned at the beginning of Sect. 4, in [35] it is shown that if the counter-strategy does not "outperform" the honest voting program (or conversely, the honest voting program does not leak more information than the counter-strategy), then indeed if a voting system provides a certain level of coercion-resistance, then it provides the same level of privacy. Fortunately, in most systems which are supposed to provide coercion-resistance, the counter-strategy indeed does not outperform the honest program.

## 5 Limitations of Cryptographic Security Analysis

The previous sections were concerned with and highlighted the importance of formally analyzing the security of e-voting systems. However, to obtain a full picture of an e-voting system and to carry out an election, many more aspects have to be taken into account which are beyond formal/cryptographic analysis. Some of these aspects are specific to the field of e-voting, while others apply to virtually all complex systems.

In what follows, we briefly discuss some of these aspects. We start with usability issues and legal requirements, as they are particularly important for e-voting systems.

*Usability and Its Relationship to Security.* E-voting systems are used by human beings, such as voters, administrators, and auditors. Therefore, the security an e-voting system provides in practice crucially depends on whether, or at least to which degree, the involved human parties follow the protocol.

For example, it is, by now, well-known that many voters are not sensitized enough to verify whether their voting devices created a correct ballot, and even if they are, they often fail to do so because the individual verification procedures, such as Benaloh challenges, are too complex (see, e.g., [24,40]). Similarly to these verification issues, many coercion-resistant e-voting protocols (e.g., Civitas [9]) require that coerced voters successfully deceive their coercer, e.g., by creating faked receipts. It is questionable whether average voters are able to do this.

Therefore, *usability* of e-voting systems is not only important to ensure that all voters can participate, but it also determines whether an e-voting system is secure in the real world: if a security procedure is difficult to use, it worsens the security of the system and may render it insecure. However, it is hard to measure usability; instead, certain usability attributes can be measured and empirically be tested, for example, how often users make the same error.

In order to analyze the impact of a system's usability w.r.t. its security, security notions are necessary which allow one to take usability attributes into account. To some degree, this is incorporated in the security definition presented

in the previous sections. For example, Küsters et al. have studied the verifiability levels of Helios, sElect, Bingo Voting, ThreeBallot, and VAV as functions of the probability that a voter (successfully) carries out her verification procedure. For example, for the system sElect [28]. Küsters et al. formally proved that sElect (roughly) provides a verifiability level of $\delta \approx (1-p)^{k+1}$ where $p$ is the probability that an honest voter carries out the verification procedure, i.e., checks whether her vote along with the verification code is in the final result, and where $k$ is the tolerated number of manipulated (honest) votes (see Sect. 2 for details). Hence, the probability that no one complains but more than $k$ votes of honest voters have been manipulated is bounded by $(1-p)^{k+1}$. Using results from usability studies one can now estimate what realistic values for $p$ are, and hence, better assess the security of a system.

*Perceived vs. Provable Security.* In addition to the provable security a system provides, the level of security perceived by regular voters might be just as important and even more important for a system to be accepted. Regular voters simply do not understand what a zero-knowledge proof is and for that reason might not trust it. Therefore simplicity and comprehensibility are very crucial, which, for example, was a driving factor for the system sElect [28]. This system features a simple and easy to understand verification procedure, allows for fully automated verification, and uses asymmetric encryption and signatures as the only cryptographic primitives.

*Legal Requirements.* Since e-voting systems are used in many countries for political elections, they have to provide certain legal requirements which depend on the political system. Unfortunately, it is difficult to formally capture all legal requirements in order to rigorously analyze whether a given e-voting system achieves them. Vice versa, it is also challenging to express formal security definitions in legal terms. There are some approaches that address this problem (see, e.g., [46,47,50]).

*Cryptographic Analysis vs. Code-Level Analysis.* Cryptographic analysis as considered in this paper, typically does not analyze the actual code of a system but a more abstract (cryptographic) model. Hence, implementation flaws can easily go undetected. While carrying out a full-fledged cryptographic security analysis of an e-voting system is already far from trivial, performing such an analysis on the code-level is even more challenging. A first such analysis for a simple e-voting system implemented in Java has been carried out in [30]. In recent years, there has also been successful code-level analysis of cryptographic protocols, such as TLS (see, e.g., [6,18] for some of the most recent work in this direction).

*Implementation and Deployment.* It is possible to model strong adversaries and capture potentially flawed program code in a formal model by weak trust assumptions and various kinds of corruptions. However, at least some parties have to be assumed to be honest in essentially all voting systems to achieve a reasonable security level. With the diverse ways systems can be and are attacked within and outside the domain of e-voting, actually guaranteeing the trust assumptions is highly non-trivial. This is even more true in political elections where e-voting

systems can be targets of extremely powerful adversaries, such as intelligence agencies and hostile states (see, e.g., [49]).

Even without assuming such powerful adversaries, securely deploying an e-voting system in practice is non-trivial and involves a lot of organizational issues which are not captured nor considered by formal analysis. For example, abstract system descriptions assume that trust is distributed among several trustees and that keys are securely generated and distributed. But it might not always be clear in practice, who the trustees should be. Again, it is therefore important to keep e-voting systems as simple as possible to avoid organizational and technical overheads in order to improve the practical security of systems.

## 6 Conclusion

The development of secure e-voting systems that are also easy to use, to understand, and to implement is still a big challenge. Rigorous formal analysis is an important piece of the puzzle. This research area has made huge progress in the last decade or so. Many central security requirements have been formulated by now and their relationships have been studied intensively. As explained in this paper, this helped to obtain a better understanding of desired security properties and to overcome some common misconceptions. This alone is already very important to help thinking about the security of e-voting systems and shaping the design of these systems. For newly proposed systems it is more and more common and expected that they come with a cryptographic security analysis. The general formal frameworks and solid formulations of fundamental security requirements are available for such analyses. While rigorous analysis is highly non-trivial and certainly does not and cannot cover all aspects in the design, implementation, and deployment of e-voting systems, it forms an important and indispensable corner stone.

## References

1. Adida, B.: Helios: web-based open-audit voting. In: USENIX 2008, pp. 335–348 (2008)
2. Arnaud, M., Cortier, V., Wiedling, C.: Analysis of an electronic boardroom voting system. In: Heather, J., Schneider, S., Teague, V. (eds.) Vote-ID 2013. LNCS, vol. 7985, pp. 109–126. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39185-9_7
3. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. IEEE J. Sel. Areas Commun. **18**(4), 593–610 (2000)
4. Benaloh, J.D.C.: Verifiable Secret-Ballot Elections. Ph.D. thesis (1987)
5. Bernhard, D., Cortier, V., Galindo, D., Pereira, O., Warinschi, B.: SoK: a comprehensive analysis of game-based ballot privacy definitions. In: S&P 2015, pp. 499–516 (2015)
6. Bhargavan, K., Blanchet, B., Kobeissi, N.: Verified models and reference implementations for the TLS 1.3 standard candidate. In: S&P 2017, pp. 483–502 (2017)
7. Chaidos, P., Cortier, V., Fuchsbauer, G., Galindo, D.: BeleniosRF: a non-interactive receipt-free electronic voting scheme. In: CCS 2016, pp. 1614–1625 (2016)

8. Chevallier-Mames, B., Fouque, P.-A., Pointcheval, D., Stern, J., Traoré, J.: On some incompatible properties of voting schemes. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (eds.) Towards Trustworthy Elections. LNCS, vol. 6000, pp. 191–199. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12980-3_11
9. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: toward a secure voting system. In: S&P 2008, pp. 354–368 (2008)
10. Cortier, V., Eigner, F., Kremer, S., Maffei, M., Wiedling, C.: Type-based verification of electronic voting protocols. In: Focardi, R., Myers, A. (eds.) POST 2015. LNCS, vol. 9036, pp. 303–323. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46666-7_16
11. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Election verifiability for helios under weaker trust assumptions. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 327–344. Springer, Cham (2014). doi:10.1007/978-3-319-11212-1_19
12. Cortier, V., Galindo, D., Küsters, R., Müller, J., Truderung, T.: SoK: verifiability notions for e-voting protocols. In: S&P 2016, pp. 779–798 (2016)
13. Cortier, V., Smyth, B.: Attacking and fixing helios: an analysis of ballot secrecy. In: CSF 2011, pp. 297–311 (2011)
14. Cortier, V., Wiedling, C.: A formal analysis of the Norwegian E-voting protocol. J. Comput. Secur. **25**(1), 21–57 (2017)
15. Culnane, C., Ryan, P.Y.A., Schneider, S.A., Teague, V.: vVote: a verifiable voting system. ACM Trans. Inf. Syst. Secur. **18**(1), 3:1–3:30 (2015)
16. Datta, A., Garg, D., Kaynar, D.K., Sharma, D., Sinha, A.: Program actions as actual causes: a building block for accountability. In: CSF 2015, pp. 261–275 (2015)
17. Delaune, S., Kremer, S., Ryan, M.: Coercion-resistance and receipt-freeness in electronic voting. In: CSFW-19, pp. 28–42 (2006)
18. Delignat-Lavaud, A., Fournet, C., Kohlweiss, M., Protzenko, J., Rastogi, A., Swamy, N., Béguelin, S.Z., Bhargavan, K., Pan, J., Zinzindohoue, J.K.: Implementing and proving the TLS 1.3 record layer. In: S&P 2017, pp. 463–482 (2017)
19. Epstein, J.: Weakness in depth: a voting machine's demise. IEEE Secur. Priv. **13**(3), 55–58 (2015)
20. Feigenbaum, J., Jaggard, A.D., Wright, R.N.: Towards a formal model of accountability. In: NSPW 2011, pp. 45–56 (2011)
21. Gößler, G., Le Métayer, D.: A general framework for blaming in component-based systems. Sci. Comput. Program. **113**, 223–235 (2015)
22. Ishai, Y., Ostrovsky, R., Zikas, V.: Secure multi-party computation with identifiable abort. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 369–386. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44381-1_21
23. Jones, D., Simons, B.: Broken Ballots: Will Your Vote Count? CSLI Publications (2012)
24. Karayumak, F., Olembo, M.M., Kauer, M., Volkamer, M.: Usability analysis of helios - an open source verifiable remote electronic voting system. In: EVT/WOTE 2011 (2011)
25. Kiayias, A., Zacharias, T., Zhang, B.: DEMOS-2: scalable E2E verifiable elections without random oracles. In: CCS 2015, pp. 352–363 (2015)
26. Kiayias, A., Zacharias, T., Zhang, B.: End-to-end verifiable elections in the standard model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 468–498. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_16
27. Kiayias, A., Zacharias, T., Zhang, B.: An efficient E2E verifiable e-voting system without setup assumptions. In: S&P 2017, vol. 15, no. 3, pp. 14–23 (2017)

28. Küsters, R., Müller, J., Scapin, E., Truderung, T.: sElect: a lightweight verifiable remote voting system. In: CSF 2016, pp. 341–354 (2016)
29. Küsters, R., Truderung, T.: Security analysis of re-encryption RPC mix nets. In: Euro S&P 2016, pp. 227–242. IEEE Computer Society (2016)
30. Küsters, R., Truderung, T., Beckert, B., Bruns, D., Kirsten, M., Mohr, M.: A hybrid approach for proving noninterference of java programs. In: CSF 2015, pp. 305–319 (2015)
31. Küsters, R., Truderung, T., Vogt, A.: A game-based definition of coercion-resistance and its applications. In: CSF 2010, pp. 122–136 (2010)
32. Küsters, R., Truderung, T., Vogt, A.: Accountability: definition and relationship to verifiability. In: CCS 2010, pp. 526–535 (2010)
33. Küsters, R., Truderung, T., Vogt, A.: Accountability: definition and relationship to verifiability. Technical report 2010/236, Cryptology ePrint Archive (2010). http://eprint.iacr.org/
34. Küsters, R., Truderung, T., Vogt, A.: Proving coercion-resistance of scantegrity II. In: Soriano, M., Qing, S., López, J. (eds.) ICICS 2010. LNCS, vol. 6476, pp. 281–295. Springer, Heidelberg (2010). doi:10.1007/978-3-642-17650-0_20
35. Küsters, R., Truderung, T., Vogt, A.: Verifiability, privacy, and coercion-resistance: new insights from a case study. In: S&P 2011, pp. 538–553 (2011)
36. Küsters, R., Truderung, T., Vogt, A.: A game-based definition of coercion-resistance and its applications. In: JCS 2012, pp. 709–764 (2012)
37. Küsters, R., Truderung, T., Vogt, A.: Clash attacks on the verifiability of e-voting systems. In: S&P 2012, pp. 395–409 (2012)
38. Küsters, R., Truderung, T., Vogt, A.: Formal analysis of chaumian mix nets with randomized partial checking. In: S&P 2014, pp. 343–358 (2014)
39. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006). doi:10.1007/11818175_22
40. Olembo, M.M., Bartsch, S., Volkamer, M.: Mental models of verifiability in voting. In: Vote-ID 2013, pp. 142–155 (2013)
41. Parkes, D., Rabin, M., Shieber, S., Thorpe, C.: Practical secrecy-preserving, verifiably correct and trustworthy auctions. In: ICEC 2006, pp. 70–81 (2006)
42. Rivest, R.L., Smith, W.D.: Three voting protocols: threeballot, VAV and twin. In: EVT 2007 (2007)
43. Ryan, P.Y.A., Rønne, P.B., Iovino, V.: Selene: voting with transparent verifiability and coercion-mitigation. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 176–192. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53357-4_12
44. Peter, Y.A., Ryan, D., Heather, J., Schneider, S., Xia, Z.: The prêt à voter verifiable election system, Technical report (2010)
45. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995). doi:10.1007/3-540-49264-X_32
46. Schmidt, A., Heinson, D., Langer, L., Opitz-Talidou, Z., Richter, P., Volkamer, M., Buchmann, J.A.: Developing a legal framework for remote electronic voting. In: VOTE-ID 2009, pp. 92–105 (2009)
47. Schwartz, B., Grice, D.: Establishing a legal framework for e-voting in Canada. Elections Canada (2014)
48. Smyth, B., Frink, S., Clarkson, M.R.: Computational Election Verifiability: Definitions and an Analysis of Helios and JCJ. Number 2015/233 (2015)

49. Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., Halderman, J.A.: Security analysis of the estonian internet voting system, pp. 703–715 (2014)
50. Stein, R., Wenda, G.: The Council of Europe and e-voting: history and impact of REC (2004) 11. In: EVOTE 2014, pp. 1–6 (2014)
51. Wolchok, S., Wustrow, E., Halderman, J.A., Prasad, H.K., Kankipati, A., Sakhamuri, S.K., Yagati, V., Gonggrijp, R.: Security analysis of India's electronic voting machines, pp. 1–14 (2010)
52. Wolchok, S., Wustrow, E., Isabel, D., Halderman, J.A.: Attacking the Washington, D.C. internet voting system. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 114–128. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32946-3_10