SOK:Verifiability Notions for E-Voting Protocols

Cortier, Veronique; Galindo, David; Kuesters, Ralf; Truderung, Tomasz

DOI:
10.1109/SP.2016.52

SoK: Verifiability Notions for E-Voting Protocols

Véronique Cortier*, David Galindo†, Ralf Küsters‡, Johannes Müller‡, Tomasz Truderung§
*LORIA/CNRS
†University of Birmingham
‡University of Trier
§Polyas GmbH
Email: veronique.cortier@loria.fr, D.Galindo@cs.bham.ac.uk, {kuesters, muellerjoh}@uni-trier.de, ttruderung@gmail.com

***Abstract*—There have been intensive research efforts in the last two decades or so to design and deploy electronic voting (e-voting) protocols/systems which allow voters and/or external auditors to check that the votes were counted correctly. This security property, which not least was motivated by numerous problems in even national elections, is called *verifiability*. It is meant to defend against voting devices and servers that have programming errors or are outright malicious. In order to properly evaluate and analyze e-voting protocols w.r.t. verifiability, one fundamental challenge has been to formally capture the meaning of this security property. While the first formal definitions of verifiability were devised in the late 1980s already, new verifiability definitions are still being proposed. The definitions differ in various aspects, including the classes of protocols they capture and even their formulations of the very core of the meaning of verifiability. This is an unsatisfying state of affairs, leaving the research on the verifiability of e-voting protocols in a fuzzy state.**

In this paper, we review all formal definitions of verifiability proposed in the literature and cast them in a framework proposed by Küsters, Truderung, and Vogt (the KTV framework), yielding a uniform treatment of verifiability. This enables us to provide a detailed comparison of the various definitions of verifiability from the literature. We thoroughly discuss advantages and disadvantages, and point to limitations and problems. Finally, from these discussions and based on the KTV framework, we distill a general definition of verifiability, which can be instantiated in various ways, and provide precise guidelines for its instantiation. The concepts for verifiability we develop should be widely applicable also beyond the framework used here. Altogether, our work offers a well-founded reference point for future research on the verifiability of e-voting systems.

***Keywords*-e-voting; verifiability; protocol analysis**

I. Introduction

Systems for electronic voting (e-voting systems) have been and are being employed in many countries for national, state-wide and municipal elections, for example in the US, Estonia, India, Switzerland, France, and Australia. They are also used for elections within companies, organizations, and associations. There are roughly two types of e-voting systems: those where the voter has to go to a polling station in order to cast her vote using a voting machine and those that allow the voter to cast her vote remotely over the Internet, using her own device. When voting at a polling station, the voter either has to fill in a paper ballot, which then is scanned by an optical scan voting system, or the voter enters her vote into a machine directly, a so-called Direct Recording Electronic (DRE) voting system.

For most voting systems used in practice today, voters have no guarantees that their votes have actually been counted: the voters' devices, voting machines, and/or voting servers might have (unintentional or deliberate) programming errors or might have been tampered with in some other way. In numerous elections it has been demonstrated that the employed systems can easily be manipulated (e.g., by replacing hardware components in voting machines) or that they contained flaws that made it possible for more or less sophisticated attackers to change the result of the elections (see, e.g., [29], [14], [2], [3], [52], [53], [48], [25]). In some occasions, announced results were incorrect and/or elections had to be rerun (see, e.g., [1], [4]). Given that e-voting systems are complex software and hardware systems, programming errors are unavoidable and deliberate manipulation of such systems is often hard or virtually impossible to detect.

Therefore, there has been intensive and ongoing research in the last two decades or so to design e-voting protocols and systems[1] which provide what is called *verifiability* (see, e.g., [21], [31], [17], [6], [15], [10], [9], [19], [34], [27], [33]). Roughly speaking, verifiability means that voters and possibly external auditors should be able to check whether the votes were actually counted and whether the published election result is correct, even if voting devices and servers have programming errors or are outright malicious. Several of such systems have already been deployed in real binding elections (see, e.g., [6], [15], [7], [44], [13], [50], [22], [26]).

For the systematic security analysis of such systems and protocols, one challenge has been to formally and precisely capture the meaning of verifiability. While the first attempts at a formal definition stem from the late 1980s [12], new definitions are still being put forward, with many definitions having been proposed in the last few years [16], [35], [32], [37], [19], [34], [33], [47], [49]. The definitions differ in many aspects, including the classes of protocols they capture, the underlying models and assumptions, the notation, and importantly, the formulations of the very core of the meaning of verifiability.

This is an unsatisfying state of affairs, which leaves the research on the verifiability of e-voting protocols and systems in a fuzzy state and raises many questions, such as: What are the advantages, disadvantages, problems, and limitations of the various definitions? How do the security guarantees provided by the definitions compare? Are they similar or fundamentally different? Answering such questions is non-trivial. It requires some common basis on which the definitions can be discussed and compared.

Contribution of this paper. First, we show that the essence of all formal definitions of verifiability proposed in the literature so far can be cast in one framework. We choose the framework proposed by Küsters, Truderung, and Vogt [37] for this purpose.

[1] In what follows, we use the terms protocols and systems interchangeably. We point out, however, that this work is mostly concerned with the protocol aspects of e-voting rather than specific system aspects.

The generic definition of verifiability in this framework is applicable to essentially any kind of protocol, with a flexible way of dealing with various trust assumptions and types of corruption. Most importantly, it allows us to capture many kinds and flavors of verifiability.

The casting of the different definitions in one framework is an important contribution by itself as it yields a uniform treatment of verifiability. This uniform treatment enables us to provide a detailed and systematic comparison of the different formal definitions of verifiability proposed in the literature until now. We present thorough discussions of all relevant definitions and models concerning their advantages, disadvantages, problems, and limitations, resulting in various new insights concerning the definitions itself and their relationships. Among others, it turns out that while the definitions share a common intuition about the meaning of verifiability, the security guarantees that are actually captured and formalized often vary, with many technical subtleties involved. Cast in tailored models, different, sometimes implicit, and often unnecessary assumptions about the protocol structure or the trust assumptions are made. For some definitions, we point out severe limitations and weaknesses.

Finally, we distill these discussions and insights into detailed guidelines that highlight several aspects any verifiability definition should cover. Based on the KTV framework, we provide a solid, general, and flexible verifiability definition that covers a wide range of protocols, trust assumptions, and voting infrastructures. Even if alternative frameworks are used, for example in order to leverage specific proof techniques or analysis tools, our guidelines provide insights on which parameters may be changed and what the implications of such modifications are. This lays down a common, uniform, and yet general basis for all design and analysis efforts of existing and future e-voting protocols. As such, our work offers a well-founded reference point for future research on the verifiability of e-voting systems and protocols.

Structure of this paper. In Section II, we introduce some notation which we use throughout this paper. We briefly recall the KTV framework in Section III. In Sections IV to VIII we then cast various definitions in this framework and based on this we carry out detailed discussions on these definitions. Further definitions are briefly discussed in Section IX, with some of them treated in detail in the appendix. The mentioned definitions and guidelines we distill from our discussions, together with various insights, are presented in Section X. The appendix contains further details, with full details provided in our technical report [20].

II. Notation and Preliminaries

Next, we provide some background on e-voting and introduce notation that we use throughout the paper.

In an e-voting protocol/system, a *voter*, possibly using some *voter supporting device (VSD)* (e.g., a desktop computer or smartphone), computes a ballot, typically containing the voter's choice in an encrypted or encoded form, and casts it. Often this means that the ballot is put on a bulletin board (see also below). The ballots are collected (e.g., from the bulletin board) and tallied by *tellers/voting authorities*. In modern e-voting protocols, the tallying is, for example, done by combining all ballots into one, using homomorphic encryption, and then decrypting the resulting ballot, or by using mix-nets, where the ballots before being decrypted are shuffled. At the beginning of an election, the voting authorities produce the election parameters prm, typically containing keys and a set of valid choices C, the *choice space*. In general, C can be an arbitrary set, containing just the set of candidates, if voters can choose one candidate among a set of candidates, or even tuples of candidates, if voters can choose several candidates or rank them. We emphasize that we consider abstention to be one of the choices in C.

In this paper, we denote the voters by $\mathsf{V}_1, \ldots, \mathsf{V}_n$ and their VSDs (if any) by $\mathsf{VSD}_1, \ldots, \mathsf{VSD}_n$. In order to cast a vote, a voter V_i first picks her choice $\mathsf{c}_i \in \mathsf{C}$. She then runs her voting procedure $\mathsf{Vote}(\mathsf{c}_i)$, which in turn might involve providing her VSD with her choice. The VSD runs some procedure $\mathsf{VoteVSD}$, given certain parameters, e.g., the voters choice. The result of running the voting procedure is a ballot b_i, which, for example, might contain c_i in encrypted form. Some models do not distinguish between the voter and her VSD, and in such a case, we simply denote the voter's voting procedure by Vote.

Often voters have to perform some verification procedure during or at the end of the election in order to prevent/detect malicious behavior by their VSDs or the voting authorities. We denote such a procedure by Verify. This procedure might for example involve checking that the voter's ballot appears on the bulletin board or performing certain cryptographic tasks. Carrying out Verify will often require some trusted device.

We denote the tellers by $\mathsf{T}_1, \ldots, \mathsf{T}_m$. As mentioned, they collect the ballots, tally them, and output the election result Tally, which belongs to what we call the *result space* R (fixed for a given election). The result is computed according to a *result function* $\rho : \mathsf{C}^n \to \mathsf{R}$ which takes the voters' choices $\vec{\mathsf{c}} = (\mathsf{c}_1, \ldots, \mathsf{c}_n)$ as input and outputs the result. (Of course dishonest tellers might try to manipulate the election outcome, which by the verifiability property, as discussed in the next section, should be detected.) The result function should be specified by the election authorities before an election starts.

At the end or throughout the election, *auditors/judges* might check certain information in order to detect malicious behavior. Typically, these checks are based solely on publicly available information, and hence, in most cases their task can be carried out by any party. They might for example check certain zero-knowledge proofs. In what follows, we consider the auditors/judges to be one party J, who is assumed to be honest.

As already noted above, most election protocols assume an append-only *bulletin board* B. An honest bulletin board stores all the input it receives from arbitrary participants in a list, and it outputs the list on request. Typically, public parameters, such as public keys, the election result, voters' ballots, and other public information, such as zero-knowledge proofs generated by voting authorities, are published on the bulletin board. As we will see, in most models (and many protocols) a single honest bulletin board is assumed. However, trust can be distributed [23]. Providing robust and trustworthy bulletin boards, while very important, is mainly considered to be a task orthogonal to the rest of the election protocol. For this reason, we will mostly refer to *the* (honest) bulletin board B, which in practice might involve a distributed solution rather than a single trusted server.

III. The KTV Framework

In this section, we briefly recall the KTV framework [37], which is based on a general computational model and provides a general definition of verifiability. As already mentioned in the introduction, in the subsequent sections we use this framework to cast all other formal definitions of verifiability. Here, we slightly simplify this framework without losing generality. These simplifications help, in particular, to smoothly deal with modeling dynamic corruption of parties (see below).

A. Computational Model

Processes are the core of the computational model. Based on them, protocols are defined.

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

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

Protocol. A *protocol P* is defined by a set of agents Σ (also called *parties* or *protocol participants*), and a program π_{a} which is supposed to be run by the agent. This program is the *honest program* of a. Agents are pairwise connected by channels and every agent has a channel to the adversary (see below).[2]

Typically, a protocol P contains a *scheduler* S as one of its participants which acts as the master program of the protocol process (see below). The task of the scheduler is to trigger the protocol participants and the adversary in the appropriate order. For example, in the context of e-voting, the scheduler would trigger protocol participants according to the phases of an election, e.g., i) register, ii) vote, iii) tally, iv) verify.

If $\pi_{a_1}, \ldots, \pi_{a_n}$ are the honest programs of the agents of P, then we denote the process $\pi_{\mathsf{a}_1} \parallel \ldots \parallel \pi_{\mathsf{a}_n}$ by π_P.

The process π_P is always run with an adversary A. The adversary may run an arbitrary probabilistic polynomial-time program and has channels to all protocol participants in π_P. Hence, a *run r of P* with adversary (adversary program) π_{A} is a run of the process $\pi_P \parallel \pi_A$. We consider $\pi_P \parallel \pi_A$ to be part of the description of r, so that it is always clear to which process, including the adversary, the run r belongs.

The honest programs of the agents of P are typically specified in such a way that the adversary A can corrupt the programs by sending the message $\mathsf{corrupt}$. Upon receiving such a message, the agent reveals all or some of its internal state to the adversary and from then on is controlled by the adversary. Some agents, such as the scheduler or a judge, will typically not be corruptible, i.e., they would ignore $\mathsf{corrupt}$ messages. Also, agents might only accept $\mathsf{corrupt}$ message upon initialization, modeling static corruption. Altogether, this allows for great flexibility in defining different kinds of corruption, including various forms of static and dynamic corruption.

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

Property. A *property* γ *of* P is a subset of the set of all runs of P.[3] By $\neg\gamma$ we denote the complement of γ.

Negligible, overwhelming, δ-bounded. As usual, a function f from the natural numbers to the interval $[0,1]$ is *negligible* if, for every $c > 0$, there exists ℓ_0 such that $f(\ell) \leq \frac{1}{\ell^c}$ for all $\ell > \ell_0$. The function f is *overwhelming* if the function $1 - f$ is negligible. A function f is δ-*bounded* if, for every $c > 0$ there exists ℓ_0 such that $f(\ell) \leq \delta + \frac{1}{\ell^c}$ for all $\ell > \ell_0$.

B. Verifiability

The KTV framework comes with a general definition of verifiability. The definition assumes a *judge* J whose role is to accept or reject a protocol run by writing accept or reject on a dedicated channel $\mathsf{decision_J}$. 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 all zero-knowledge proofs (if any). Intuitively, J accepts a run if the protocol run looks as expected. The judging procedure should be part of the protocol specification. So, formally the judge should be one of the protocol participants in the considered protocol P, and hence, precisely specified. The input to the judge typically is solely public information, including all information and complaints (e.g., by voters) posted on the bulletin board. Therefore the judge can be thought of as a “virtual” entity: the judging procedure can be carried out by any party, including external observers and even voters themselves.

The definition of verifiability is centered around the notion of a *goal* of the protocol. Formally, a goal is simply a property γ of the system, i.e. a set of runs (see Section III-A). Intuitively, such a goal specifies those runs which are “correct” in some protocol-specific sense. For e-voting, intuitively, the goal would contain those runs where the announced result of the election corresponds to the actual choices of the voters.

Now, the idea behind the definition is very simple. The judge J should accept a run only if the goal γ is met, and hence, the published election result corresponds to the actual choices of the voters. More precisely, the definition requires that the probability (over the set of all runs of the protocol) that the goal γ is not satisfied but the judge nevertheless accepts the run is δ-bounded. 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 ballot appears on the bulletin

[2] We note that in [37] agents were assigned sets of potential programs they could run plus an honest program. Here, w.l.o.g., they are assigned only one honest program (which, however, might be corrupted later on).

[3] Recall that the description of a run r of P contains the description of the process $\pi_P \parallel \pi_{\mathsf{A}}$ (and hence, in particular the adversary) from which r originates. Hence, γ can be formulated independently of a specific adversary.

board, giving an adversary A the opportunity to manipulate or drop some ballots without being detected. Therefore, $\delta = 0$ cannot be achieved in general.

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

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

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

is δ-bounded as a function of ℓ.

A protocol *P* could trivially satisfy verifiability with a judge who never accepts a run. Therefore, one of course would also require a *soundness* or *fairness* condition. That is, one would except at the very least that if the protocol runs with a benign adversary, which, in particular, would not corrupt parties, then the judge accepts a run. Formally, for a benign adversary π_{A} we require that $\mathsf{Pr}[\pi^{(\ell)} \mapsto (\mathsf{J}\colon \mathsf{accept})]$ is overwhelming. One could even require that the judge accepts a run as soon as a certain subset of protocol participants are honest, e.g., the voting authorities (see, e.g., [37] for a more detailed discussion). These kinds of fairness/soundness properties can be considered to be sanity checks of the judging procedure and are typically easy to check. Most definitions of verifiability in the literature do not explicitly mention this property. For brevity of presentation, we therefore mostly ignore this issue here as well. In the subsequent sections, we, however, mention and briefly discuss fairness conditions unless addressed by a definition.

Definition 1 captures the essence of the notion of verifiability in a very simple way, as explained above. In addition, it provides great flexibility and it is applicable to arbitrary classes of e-voting protocols. This is in contrast to most other definitions of verifiability, as we will see in the subsequent sections, which are mostly tailored to specific classes of protocols. This flexibility in fact lets us express the other definitions in terms of Definition 1. There are two reasons for the flexibility. First, the notion of a protocol *P* used in Definition 1 is very general: a protocol is simply an arbitrary set of interacting Turing machines, with one of them playing the role of the judge. Second, the goal γ provides great flexibility in expressing what an e-voting protocol is supposed to achieve in terms of verifiability.

As mentioned in the introduction, in the following sections, we present all relevant definitions of verifiability from the literature, discuss them in detail, and then express their essence in terms of Definition 1. The latter, in particular, allows for a uniform treatment of the various definitions from the literature, and by this a better understanding of the individual definitions and their relationships to other definitions. Advantages and disadvantages of the definitions can be clearly seen in terms of the classes of protocols that are captured by the definitions and the security guarantees that they give. It seems particularly interesting to see which goals γ (in the sense defined above) these definitions consider. In Section X, among others, we use these insights to distill precise guidelines for important aspects of definitions of verifiability and propose goals γ applicable to a broad class of e-voting protocols, and hence, we provide a particularly useful instantiation of Definition 1 given what we have learned from all definitions from the literature.

The following sections, in which we present and discuss the various definitions of verifiability from the literature, are ordered in such a way that definitions that are close in spirit are discussed consecutively. All sections follow the same structure. In every section, we first briefly sketch the underlying model, then present the actual definition of verifiability, followed by a discussion of the definition, and finally the casting of the definition in Definition 1. We emphasize that the discussions about the definitions provided in these sections reflect the insights we obtained by casting the definitions in the KTV framework. For simplicity and clarity of the presentation, we, however, present the (informal) discussions before casting the definitions.

IV. A Specific Verifiability Goal by Küsters et al.

In [37], Küsters et al. also propose a specific family of goals for e-voting protocols that they used in [37] as well as subsequent works [40], [39], [38]. We present this family of goals below as well as the way they have instantiated the model when applied to concrete protocols. Since this is a specific instantiation of the KTV framework, we can omit the casting of their definition in this framework.

A. Model

When applying the KTV framework in order to model specific e-voting protocols, Küsters et al. model static corruption of parties. That is, it is clear from the outset whether or not a protocol participant (and in particular a voter) is corrupted. An honest voter V runs her honest program π_{V} with her choice $\mathsf{c} \in \mathsf{C}$ provided by the adversary. This choice is called the *actual choice* of the voter, and says how the voter intends to vote.

B. Verifiability

In [37], Küsters et al. propose a general definition of accountability, with verifiability being a special case. Their verifiability definition, as mentioned, corresponds to Definition 1. Their definition, however, also captures the fairness condition which we briefly mentioned in Section III-B. To this end, Küsters et al. consider Boolean formulas with propositional variables of the form $\mathsf{hon}(\mathsf{a})$ to express constraints on the honesty of protocol participants. Roughly speaking, given a Boolean formula φ, their fairness condition says that if in a run parties are honest according to φ, then the judge should accept the run.

While just as in Definition 1, the verifiability definition proposed by Küsters et al. does not require to fix a specific goal, for e-voting they propose a family $\{\gamma_k\}_{k\geq 0}$ of goals, which has been applied to analyze various e-voting protocols and mix nets [37], [40], [39], [38].

Roughly speaking, for $k \geq 0$, the goal γ_k contains exactly those runs of the voting protocol in which *all but up to k* votes of the honest voters are counted correctly *and* every dishonest voter votes at most once.

Before recalling the formal definition of γ_k from [37], we first illustrate γ_k by a simple example. For this purpose, consider an election with five eligible voters, two candidates, with the result of the election simply being the number of votes for each candidate. Let the result function ρ (see Section II) be defined accordingly. Now, let *r* be a run with three honest and two dishonest voters such that A, A, B are the actual choices of the honest voters in r and the published election result in r is the following: one vote for A and four votes for B. Then, the goal γ_1 is satisfied because the actual choice of one of the honest voters choosing A can be changed to B and at the same time the choice of each dishonest voter can be B. Hence, the result is equal to $\rho(A,B,B,B,B)$, which is the published result. However, the goal γ_0 is not satisfied in r because in this case, all honest voters' choices (A,A,B) have to be counted correctly, which, in particular, means that the final result has to contain at least two votes for A and at least one vote for B. In particular, a final result with only two votes for A but none for B would also not satisfy γ_0, but it would satisfy γ_1. (Recall from Section II that abstention is a possible choice.)

Definition 2 (Goal γ_k)**.** *Let r be a run of an e-voting protocol. Let n_h be the number of honest voters in r and $n_d = n - n_h$ be the number of dishonest voters in r. Let $\mathsf{c}_1, \ldots, \mathsf{c}_{n_h}$ be the actual choices of the honest voters in this run, as defined above. Then* γ_k is satisfied in r *if there exist valid choices $\tilde{\mathsf{c}}_1, \ldots, \tilde{\mathsf{c}}_n$ such that the following conditions hold true:*

(i) The multiset $\{\tilde{\mathsf{c}}_1, \ldots, \tilde{\mathsf{c}}_n\}$ contains at least $n_h - k$ elements of the multiset $\{\mathsf{c}_1, \ldots, \mathsf{c}_{n_h}\}$.

(ii) The result of the election as published in r (if any) is equal to $\rho(\{\tilde{\mathsf{c}}_1, \ldots, \tilde{\mathsf{c}}_n\})$.

If no election result is published in r, then γ_k is not satisfied in r.

With this goal, Definition 1 requires that if more than k votes of honest voters were dropped/manipulated or the number of votes cast by dishonest voters (which are subsumed by the adversary) is higher than the number dishonest voters (ballot stuffing), then the judge should not accept the run. More precisely, the probability that the judge nevertheless accepts the run should be bounded by δ.

We note that the definition of γ_k does not require that choices made by dishonest voters in r need to be extracted from r in some way and that these extracted choices need to be reflected in $\{\tilde{\mathsf{c}}_1, \ldots \tilde{\mathsf{c}}_n\}$: the multiset $\{\tilde{\mathsf{c}}_1, \ldots, \tilde{\mathsf{c}}_n\}$ of choices is simply quantified existentially. It has to contain $n_h - k$ honest votes but no specific requirements are made for votes of dishonest voters in this multiset. They can be chosen fairly independently of the specific run r (except for reflecting the published result and the requirement that there is at most one vote for every dishonest voter). This is motivated by the fact that, in general, one cannot provide any guarantees for dishonest voters, since, for example, their ballots might be altered or ignored by dishonest authorities without the dishonest voters complaining (see also the discussion in [37]).

C. Discussion

The goal γ_k makes only very minimal assumptions about the structure of a voting system. Namely, it requires only that, given a run r, it is possible to determine the actual choice (intention) of an honest voter and the actual election result. Therefore, the goal γ_k can be used in the analysis of a wide range of e-voting protocols.

One drawback of the goal γ_k is that it assumes static corruption. Another disadvantage of γ_k (for $k > 0$) is the fact that it does not distinguish between honest votes that are dropped and those that are turned into different valid votes, although the impact on the final result by the second kind of manipulation is stronger than the one by the first kind. To illustrate this issue, consider two voting protocols P_1 and P_2 (with the result function ρ being the counting function). In P_1 the adversary might not be able to turn votes by honest voters into different valid votes, e.g., turn a vote for candidate A into a vote for B. This can be achieved if voters sign their ballots. In this case, the adversary can only drop ballots of honest voters. In P_2 voters might not sign their ballots, and hence, the adversary can potentially manipulate honest votes. Now, P_1 obviously offers stronger verifiability because in P_1 votes of honest voters can only be dropped, but not changed: while in P_2 the adversary could potentially turn five honest votes, say for candidate A, into five votes for B, in P_1 one could at most drop the five honest votes, which is less harm. Still both protocols might achieve the same level of verifiability in terms of the parameters γ_k and δ. If γ_k distinguished between dropping of votes and manipulation, one could distinguish the security levels of P_1 and P_2.

In Section X we propose a new goal which solves the mentioned problems.

V. Verifiability by Benaloh

In this section, we study the verifiability definition by Benaloh [12]. This definition constitutes the first formal verifiability definition proposed in the literature, and hence, the starting point for the formal treatment of verifiability. This definition is close in its essence to the one discussed in Section IV.

A. Model

Following [12], an *l-threshold m-teller n-voter election system* (or simply (l,m,n)*-election system*) E is a synchronous system of communicating processes (probabilistic Turing machines) consisting of m tellers $\mathsf{T}_1, \ldots, \mathsf{T}_m$, n voters $\mathsf{V}_1, \ldots, \mathsf{V}_n$ and further participants. Each process of an election system controls one *bulletin board*. Each bulletin board can be read by every other process, but only be written by the owner.

The intended (honest) behavior of the system participants is specified by an election schema. An (l,m,n)*-election schema* S consists of a collection of programs to be used by the participants of an (l,m,n)-election system and an efficiently computable function check, which, given the security parameter ℓ and the messages posted to the public bulletin boards, returns either "good" or "bad". The election schema S describes a program π_T for each teller process and two possible programs for each voter: π_{yes} to be used to cast a "yes" vote and program π_{no} to be used to cast a "no" vote. At the end of the election, each teller T_k releases a value τ_k.

Any process which follows (one of) its program(s) prescribed by S is said to be *proper*. We say that a voter *casts a valid "yes" vote*, if the messages it posts are consistent with the program π_{yes}, and similarly for a "no" vote. Note that a proper voter, by definition, always casts a valid vote; an improper voter may or may not cast a valid vote, and if it does not cast a valid vote, that fact may or may not be detectable by others.

The *tally* of an election is the pair (t_{yes}, t_{no}) where t_{yes} and t_{no} are the numbers of voters who cast valid "yes" and "no" votes, respectively. Note that this pair expresses the expected result corresponding to the cast valid votes. The tally of the election is said to be *correct* if $\rho(\tau_1, \ldots, \tau_m) = (t_{yes}, t_{no})$, where ρ is a pre-determined function. The expression $\rho(\tau_1, \ldots, \tau_m)$ describes the actual tally, that is the result of the election as jointly computed by the tellers (and combined using the function ρ).

B. Verifiability

Now, in [12], verifiability is defined as follows.

Definition 3 (Verifiability)**.** *Let δ be a function of ℓ. The (l,m,n)-election schema* S *is said to be* verifiable with confidence $1-\delta$ *if, for any election system* E, check *satisfies the following properties for random runs of* E *using security parameter ℓ:*

(1) If at least l tellers are proper in E*, then, with probability at least $1-\delta(\ell)$,* check *returns good and the tally of the election is correct.*

(2) The joint probability that check *returns good and the election tally is not correct is at most $\delta(\ell)$.*

The election schema S *is said to be* verifiable *if δ is negligible.*

Condition (1) of Definition 3 expresses a fairness condition (see Section III-B), where to guarantee the successful (and correct) run of a protocol, it is enough to only assume that l tellers are honest.

Condition (2) of Definition 3 is the core of Definition 3. Roughly speaking, it corresponds to Definition 1 with the goal γ_0 defined by Küsters et al. (see Section IV-B). As discussed below, there are, however, subtle differences, resulting in a too strong definition.

C. Discussion

As mentioned before, Benaloh's definition constitutes the first formal verifiability definition, mainly envisaging an entirely computer-operated process based on trusted machines and where, for example, voters were not asked to perform any kind of verification. Given this setting, the definition has some limitations from a more modern point of view.

Similarly to the definition in Section IV, this definition is fairly simple and general, except that only yes/no-votes are allowed, tellers are explicitly required in this definition, and *every* participant has his/her own bulletin board. These restrictions, however, are not necessary in order to define verifiability as illustrated in Section IV. This definition also focuses on static corruption. The main problem with this definition is that it is too strong in settings typically considered nowadays, and hence, it would exclude most e-voting protocols, even those that intuitively should be considered verifiable.

As already mentioned, Condition (2) of Definition 3 is related to the goal γ_0. The goal γ_0 is, however, typically too strong because, for example, not all honest voters perform the verification process, e.g., check whether their ballots actual appear on the bulletin board. Hence, there is a non-negligible chance that the adversary is not caught when dropping or manipulating ballots. This is why Küsters et al. (Section IV) considered goals γ_k for $k \geq 0$.

Moreover, the goal considered here is even stronger (see also Section V-D). Condition (2) in Definition 3 is concerned not only with honest voters, but also with dishonest ones who post messages consistent with honest programs. Now, the problem is that a dishonest voter could simply cast a vote just like an honest one. The dishonest voter may, however, never complain even if dishonest tellers (who might even team up with the dishonest voter) drop or manipulate the ballot of the dishonest voter. Hence, it cannot be guaranteed that votes of such dishonest voters are counted, unlike what Condition (2) in Definition 3 requires. So, Definition 3 would deem almost all e-voting protocols in settings typically considered nowadays insecure, even completely reasonable ones.

Also, Condition (1) of Definition 3 may be too strong in many cases. It says that the threshold of l tellers is enough to guarantee that a protocol run is correct, i.e., in terms of the KTV framework, the judge would accept the run. It might not always be possible to resolve disputes, for example, when voters complain (possibly for no reason). For the sake of generality of the definition, it would therefore be better to allow for a more flexible fairness condition, as the one sketched in Section IV.

D. Casting in the KTV Framework

We now cast Definition 3 in the KTV Framework. To this end, we have to define the class of protocols considered in [12] in terms of the KTV Framework and the goal γ.

Protocol P_B. The set of agents Σ consists of the voters, the tellers, the judge J, one bulletin board for each of these participants, and the remaining participants. Since static corruption is considered, the agents accept a $\mathsf{corrupt}$ message only at the beginning of an election run. The bulletin boards and the judge do not accept $\mathsf{corrupt}$ message at all. As usual, we consider an additional honest party, the scheduler. The honest programs are defined as follows:

- The scheduler behaves in the expected way: it triggers all the parties in every protocol step. The judge is triggered in the final phase, after the tellers are supposed to output their (partial) tallying.
- The honest behavior of the bulletin boards is as described in Section II, with the only difference that a bulletin board owned by some party accepts messages posted only by this party; it serves its content to all parties, though.
- When a voter V runs her honest program π_{V}, she first expects "yes" or "no" as input (if the input is empty, she stops). If the input is "yes", she runs π_{yes}, and otherwise π_{no}. She sends the result to her bulletin board $\mathsf{B}(\mathsf{V})$; π_{V} might later be triggered again to perform verification steps.
- When the judge J runs π_{J} and is triggered in the final phase, it reads the content of all the bulletin boards and computes the result of the function check on this content. If check evaluates to "good", it outputs "accept", and otherwise "reject".
- The honest program π_{T} of T depends on the concrete election system that is used.

The goal. We define the goal γ_0^* to be γ_0 (see Definition 2), with the difference that, instead of considering the multiset $\mathsf{c}_1,\ldots,\mathsf{c}_{n_h}$ of choices of honest voters only, we now consider the multiset of choices of all voters who cast a valid vote. This, as explained, includes not only honest voters, but might also include some dishonest voters.

Verifiability. Now, it should be clear that the notion of verifiability defined by Benaloh can be characterized in terms of Definition 1 as (γ_0^*,δ)-verifiability.[4] As discussed before, the goal γ_0^* is too strong for several reasons.

[4] Recall that here we do not consider the fairness conditions.

VI. E2E Verifiability by Kiayias et al.

In this section, we study the end-to-end verifiability definition by Kiayias et al. [34], [33].

A. Model

According to Kiayias et al., an e-voting scheme Π is a tuple (Setup, Cast, Tally, Result, Verify) of probabilistic polynomial-time (ppt) algorithms where Cast and Tally are interactive. The entities are the election authority EA, the bulletin board B, the tellers $\mathsf{T}_1,\ldots,\mathsf{T}_m$ and the voters. The algorithm Cast is run interactively between B and a voter V_i where the voter operates a *voter supporting device* VSD on the following inputs: public parameters $\mathsf{prm}_{\mathsf{pub}}$, a choice c_i, and her credentials cred_i. Upon successful termination, V_i obtains a receipt α_i. The algorithm Tally is run between EA, the tellers and B. This computation updates the public transcript τ. The algorithm $\mathsf{Verify}(\tau,\alpha_i)$ denotes the individual verification of the public transcript τ by voter V_i, while $\mathsf{Verify}(\tau,\mathsf{st}_i)$ denotes the verification of τ by teller T_i on her private state st_i; the output of Verify is a bit. The algorithm Setup is run for setting up an election, and the algorithm Result, given τ, outputs the result of the election, if any.

B. E2E Verifiability

The E2E-verifiability definition by Kiayias et al. [34], [33] is given in Figure 1. The adversary can corrupt voters and tellers, and he controls the EA and the VSDs of voters. The bulletin board is assumed to be honest, but the adversary can determine the content τ of it. The set $\mathsf{V}_{\mathsf{cast}}$ contains all voters who successfully terminated their protocol, and hence, obtained a receipt. However, they might not have verified their receipts. The adversary wins the game if (i) $|\mathsf{V}_{\mathsf{cast}}| \geq \theta$, i.e., not too few voters successfully terminated, and (ii) would all of these voters verify their receipt, then they would verify successfully, and (iii) the published result of the election $\mathsf{Result}(\tau)$ deviates by at least k from the actual result $\rho(\mathsf{c}_1,\ldots,\mathsf{c}_n)$ obtained according to the actual votes of voters. More specifically, for the last condition, i.e., Condition (iii), Kiayias et al. postulates the existence of a vote extractor algorithm Extr (not necessarily running in polynomial-time) which is supposed to determine the votes of all voters not in $\mathsf{V}_{\mathsf{cast}}$, where Extr is given the transcript and the receipt of voters in $\mathsf{V}_{\mathsf{cast}}$ as input. Note that the adversary wins the game if Extr fails to return these votes (Condition (iii-b)).

Definition 4 (E2E-verifiability)**.** *Let $0<\delta<1$ and $n,w,k,t,\theta \in \mathbb{N}$ with $k>0$ and $0<\theta\leq n$. The election protocol Π w.r.t. election function achieves E2E verifiability with error δ, for a number of at least θ honest successful voters and tally deviation k, if there exists a vote-extractor* Extr *such that for any adversary* A *controlling less than $n-\theta$ voters and t tellers, the EA and all* VSD*'s holds:* $\Pr\left[G^{\mathsf{A},\mathsf{Extr},k,\theta}(1^\ell,w,n,t)=1\right]\leq\delta$.

We note that [34] considers a fairness condition (named *perfect correctness)* similarly to the one in Section III-B.

C. Discussion

We first note that the definition is too specific in some situations due to the use of the extractor in the definition. Indeed, it does not seem to apply to voting protocols where ballots published on the bulletin board hide the choices of voters information-theoretically, such as [24]. In this case, the adversary could, for example, corrupt some voters but just follow the protocol honestly. For these voters and those in $\mathsf{V}_{\mathsf{cast}}$ the extractor could not determine their votes, and hence, it would be very likely that the adversary wins the game in Figure 1: if the extractor outputs votes, then it would be very likely that Condition (iii-a) is satisfied, and otherwise Condition (iii-b) would be satisfied.

This problem can be fixed by providing the extractor with the votes of the voters in $\mathsf{V}_{\mathsf{cast}}$, not only with their receipts. In this case, the extractor could simply compute $\mathsf{Result}(\tau)$ and choose $(\mathsf{c}_i)_{\mathsf{V}_i\notin\mathsf{V}_{\mathsf{cast}}}$ such that $\mathsf{d}_1(\mathsf{Result}(\tau),\rho(\mathsf{c}_1,\ldots,\mathsf{c}_n))$ is minimal. This would be the best extractor, i.e., the one that makes it the hardest for the adversary to win the game. Note that this extractor does not have to actually extract votes from τ, or even look closely at τ, except for computing $\mathsf{Result}(\tau)$.

Conditions (iii-a) and (iii-b) could therefore be replaced by the following one:

(iii)* For any combination of choices $(\mathsf{c}_i)_{\mathsf{V}_i\notin\mathsf{V}_{\mathsf{cast}}}$:

$$\mathsf{d}_1(\mathsf{Result}(\tau),\rho(\mathsf{c}_1,\ldots,\mathsf{c}_n))\geq k.$$

This is then similar to Definition 2 where votes of dishonest voters are quantified existentially. (Note that (iii)* talks about when verifiability is broken, while Definition 2 talks about the goal, i.e., what verifiability should achieve, hence the switch from existential quantification in Definition 2 to universal quantification in (iii)*). As explained in Section IV, the existential quantification is very reasonable because, for several reasons, it is often not possible to extract votes of dishonest voters.

E2E Verifiability Game $G^{\mathsf{A},\mathsf{Extr},k,\theta}(1^\ell,w,n,t)$

1) A chooses a list of choices $\mathsf{C}=\{\mathsf{c}_1,\ldots,\mathsf{c}_w\}$, a set of voters $\{\mathsf{V}_1,\ldots,\mathsf{V}_n\}$, and a set of tellers $\{\mathsf{T}_1,\ldots,\mathsf{T}_t\}$. It provides the challenger Ch with these sets along with information $\mathsf{prm}_{\mathsf{pub}}$ and voter credentials $\{\mathsf{cred}_i\}_{1\leq i\leq n}$. Throughout the game, Ch plays the role of B.
2) A and Ch engage in an interaction where A schedules the Cast protocols of all voters. For each voter V_i, A can either completely control the voter or allow Ch operate on V_i's behalf, in which case A provides a choice c_i to Ch. Then, Ch engages in the Cast protocol with the adversary A, so that A plays the roles of EA and VSD. Provided the protocol terminates successfully, Ch obtains a receipt α_i on behalf of V_i.
3) Finally, A posts the election transcript τ to B.

The game returns a bit which is 1 if the following conditions hold true:

i) $|\mathsf{V}_{\mathsf{cast}}|\geq\theta$, (i.e., at least θ honest voters terminated)
ii) $\forall \mathsf{V}_i\in\mathsf{V}_{\mathsf{cast}}\colon \mathsf{Verify}(\tau,\alpha_i)=1$ (i.e. the honest voters that terminated verified successfully)

and either one of the following two conditions:

(iii-a). If $\perp\neq(\mathsf{c}_i)_{\mathsf{V}_i\notin\mathsf{V}_{\mathsf{cast}}}\leftarrow\mathsf{Extr}(\tau,\{\alpha_i\}_{\mathsf{V}_i\in\mathsf{V}_{\mathsf{cast}}})$, then $\mathsf{d}_1(\mathsf{Result}(\tau),\rho(\mathsf{c}_1,\ldots,\mathsf{c}_n))\geq k$ (d_1 is a metric).

(iii-b). $\perp\leftarrow\mathsf{Extr}(\tau,\{\alpha_i\}_{\mathsf{V}_i\in\mathsf{V}_{\mathsf{cast}}})$

Fig. 1: E2E-verifiability by Kiayias et al.

Our second observation is that the definition (even the version with the fix above) is too weak in the following sense. Consider runs where honest voters cast their votes successfully, and hence, obtain a receipt, but do not verify their receipt, and where the verification would even fail. Because of Condition (ii), the adversary would right away loose the game in these runs. However, these runs are realistic threats (since often voters do not verify), and hence, guarantees should be given even for such runs. The game in Figure 1 simply discards such runs. Therefore, instead of Condition (ii) one should simply require that the judge (looking at τ and waiting for complaints from voters, if any) accepts the run. Note that if the judge does not accept the run, then the election is invalid.

D. Casting in the KTV Framework

Protocol P_{KZZ}**.** The set of agents Σ consists of the voters, the bulletin board B, the voting authority EA, the judge J, the tellers $\mathsf{T}_1, \ldots, \mathsf{T}_m$ and the remaining participants.

When a voter V runs her honest program π_{V} in the casting phase, she expects a choice c, a credential and the public parameters of the election (if her input is empty, she stops). Then, she runs Cast in interaction with B, and expects a receipt α (if she does not receive a receipt, she stops). When the voter is triggered by the judge in the verification phase, the voter reads the election transcript τ from the bulletin board B (if she does not receive τ, she outputs "reject") and runs $\mathsf{Verify}(\tau, \alpha)$. If $\mathsf{Verify}(\tau, \alpha)$ evaluates to "false" or "true", respectively, she sends "reject" or "accept" to the judge J. The definition of Kiayias et al. is not explicit about whether voters always verify when triggered or not. So here one could also model that they decide whether they verify according to some probability distribution.

When a teller T runs its honest program π_{T} in the setup phase, it interacts with the remaining tellers, the EA and B. It expects as output its secret state st (otherwise, it stops). In the tally phase, on input st and the contents of B (if any input is empty, it stops), it runs Tally in interaction with B and EA, and outputs a partial tally ta that is sent to EA.

When the election authority EA runs its honest program π_{EA}, it expects a security parameter 1^ℓ in the setup phase (if the input is empty, it stops). Then, it runs Setup in interaction with B and the tellers, and outputs the election parameters, which are published in B, and the voters' credentials $(\mathsf{cred}_1, \ldots, \mathsf{cred}_n)$, which are sent to the corresponding voters $(\mathsf{V}_1, \ldots, \mathsf{V}_n)$. In the tally phase, EA runs Tally in interaction with B and the tellers, and publishes the partial tally data $ta_1, \ldots, ta_m$ produced by each teller at the end of the interaction.

When the judge J runs its honest program π_{J} and is triggered in the verification phase, it reads the election transcript τ. It performs whatever check prescribed by the protocol. If one of these checks fails, J outputs "reject". Otherwise, J iteratively triggers all voters and asks about their verification results (if any). If one of the voters rejects, J outputs "reject", and otherwise, "accept".

E2E verifiability. We define the goal $\gamma_{\theta,k,\mathsf{Extr}}$, which is parameterized by θ, k, and Extr as in Figure 1, to be the set of runs of P_{KZZ} (with some adversary A) such that at least one of the Conditions (i), (ii), (iii-a) or (iii-b) in Figure 1 is not satisfied. With this, Definition 4, corresponds to the notion of $(\gamma_{\theta,k,\mathsf{Extr}}, \delta)$-verifiability according to Definition 1 when the same extractors are used and one quantifies over the same set of adversaries.

As already discussed above, this definition on the one hand is too specific (due to the use of the extractor) and on the other hand too weak (due to Condition (ii)). Therefore, as mentioned, the definition would be improved if Conditions (iii-a) and (iii-b) were replaced by (iii)* and Condition (ii) was replaced by the condition that the judge accepts the run. If one set $\theta = 0$ in addition, then Definition 4 would closely resemble γ_k from Definition 2.

VII. Computational Election Verifiability by Cortier et al.

In this section, we study the definition of verifiability by Cortier et al. [19], which can be seen as an extension of a previous verifiability definition by Catalano et al. [32], whereby the bulletin board may act maliciously, and thus it could potentially perform ballot stuffing (i.e. stuff itself with self-made ballots on behalf of voters who did not vote) or erase ballots previously cast by voters.

A. Model

Cortier et al. [19] model an e-voting scheme Π as a tuple (Setup, $\mathsf{Credential}$, Vote, $\mathsf{VerifyVote}$, Valid, Board, Tally, Verify) of ppt algorithms where $\mathsf{VerifyVote}$ and Verify are non-interactive. The entities are the registrar Reg, the bulletin board B, the teller T and the voters. The algorithm $\mathsf{Setup}(\ell)$ is run by the teller T, and outputs the public parameters of the election $\mathsf{prm}_{\mathsf{pub}}$ and the secret tallying key sk. The procedure $\mathsf{Credential}$ is run by Reg with the identity id_i of voter V_i, and outputs a public/secret credential pair $(\mathsf{upk}_i, \mathsf{usk}_i)$. The algorithms discussed next implicitly take $\mathsf{prm}_{\mathsf{pub}}$ as input. The algorithm Vote is run interactively between B and a voter V_i, on inputs $\mathsf{prm}_{\mathsf{pub}}$, a choice c_i and her credentials $(\mathsf{upk}_i, \mathsf{usk}_i)$. Upon successful termination, a ballot b_i is appended to the public transcript τ of the election. The procedure $\mathsf{Valid}(\mathsf{b})$ outputs 1 or 0 depending on whether b is well-formed. Board denotes the algorithm that B must run to update τ. The algorithm Tally is run at the end of the election by T, given the content of B and the secret key sk as input, and outputs tallying proofs P and the final election result Result. $\mathsf{VerifyVote}(\tau, \mathsf{upk}_i, \mathsf{usk}_i, \mathsf{b})$ is an algorithm run by voter V_i that checks whether ballot b appears in τ. The algorithm $\mathsf{Verify}(\tau, \mathsf{Result}, \mathsf{P})$ denotes the verification of the result of the election, while $\mathsf{VerifyVote}(\tau, \mathsf{upk}_i, \mathsf{b}_i)$ denotes the verification that ballot b_i from voter V_i was included in the final transcript of the election as published by B.

B. Verifiability Against Malicious Bulletin Board

In the e-voting system Helios [6], a dishonest bulletin board B may add ballots, since it is the sole entity checking the eligibility of voters. If B is corrupted, then it might stuff the ballot box with ballots on behalf of voters that in fact did not vote. This problem, as already mentioned in Section IV-B, is called ballot stuffing. The work in [19] gives a definition of verifiability in the computational model to account for a malicious bulletin board. To defend voters against a dishonest B, a registration authority Reg is required. Depending on whether both B and Reg are required to be honest, [19] defines *weak verifiability* (both are honest) or *strong verifiability* (not simultaneously dishonest).

In Figure 2 we give a snapshot of the cryptographic game used in [19] to define verifiability in case B is dishonest. The adversary has oracles to register voters, corrupt voters, and

Experiment $\mathsf{Exp}^{\mathsf{verb}}_{\mathsf{A},\Pi}$

Adversary A has access to the following oracles:

- $\mathcal{O}\mathsf{reg}(id)$: creates voters' credentials via $(\mathsf{upk}_{id}, \mathsf{usk}_{id}) \leftarrow \mathsf{Credential}(id)$, stores them as $\mathcal{U} = \mathcal{U} \cup \{(id, \mathsf{upk}_{id}, \mathsf{usk}_{id})\}$, and returns upk_{id} to the attacker.
- $\mathcal{O}\mathsf{corrupt}(id)$: firstly, checks if an entry $(i,*,*)$ appears in $\mathcal{U}$; if not, stops. Else, gives $(\mathsf{upk}_{id}, \mathsf{usk}_{id})$ to A, updates a list of corrupted voters $\mathcal{CU} = \mathcal{CU} \cup \{(i, \mathsf{upk}_{id})\}$ and updates the list of honest cast ballots HVote by removing any occurrence $(id,*,*)$.
- $\mathcal{O}\mathsf{vote}(id, \mathsf{c})$: if $(i,*,*) \notin \mathcal{U}$, or $(i,*) \in \mathcal{CU}$, aborts; else returns $b = \mathsf{Vote}(i, \mathsf{upk}_{id}, \mathsf{usk}_{id}, \mathsf{c})$ and replaces any previous entry $(id,*,*)$ in HVote with (i, c, b). The latter list is used to record the voter's intention.

Let Checked $\subseteq$ HVote contain those id's who checked that their ballot appears in τ at the end of the election. The experiment outputs a bit as follows, with 1 meaning that the attacker was successful:

(1) $(\tau, \mathsf{Result}, \mathsf{P}) \leftarrow \mathsf{A}^{\mathcal{O}\mathsf{reg}, \mathcal{O}\mathsf{corrupt}, \mathcal{O}\mathsf{vote}}$

(2) if $\mathsf{Verify}(\tau, \mathsf{Result}, \mathsf{P}) = 0$ return 0

(3) if $\mathsf{Result} = \perp$ return 0

(4) if $\exists\ (i^A_1, \mathsf{c}^A_1, *), \ldots, (i^A_{n_A}, \mathsf{c}^A_{n_A}, *) \in \mathsf{HVote} \backslash \mathsf{Checked}$

$\exists\ \mathsf{c}^B_1, \ldots, \mathsf{c}^B_{n_B} \in \mathsf{C}$ s.t. $0 \le n_B \le |\mathcal{CU}|$ s.t.

$\mathsf{Result} = \rho\left(\{\mathsf{c}^E_i\}^{n_E}_{i=1}\right) \star_{\mathsf{R}} \rho\left(\{\mathsf{c}^A_i\}^{n_A}_{i=1}\right) \star_{\mathsf{R}} \rho\left(\{\mathsf{c}^B_i\}^{n_B}_{i=1}\right)$

return 0 else return 1

where $\mathsf{Checked} = \{(i^E_1, \mathsf{c}^E_1, b^E_1), \ldots, (i^E_{n_E}, \mathsf{c}^E_{n_E}, b^E_{n_E})\}$

Fig. 2: Verifiability against bulletin board by Cortier *et al.* [19]

let honest voters vote. The condition for winning the game is explained below. Note that Cortier et al. assume that the result function admits *partial counting*, namely $\rho(S_1 \cup S_2) = \rho(S_1) \star_{\mathsf{R}} \rho(S_2)$ for any two lists S_1, S_2 containing sequences of elements $c \in \mathsf{C}$ and where $\star_{\mathsf{R}} : \mathsf{R} \times \mathsf{R} \rightarrow \mathsf{R}$ is a commutative operation. For example, the standard result function that counts the number of votes per candidate admits partial counting.

Definition 5 (Verifiability against malicious bulletin board)**.** *An election scheme achieves* verifiability against the bulletin board *if the success rate* $\mathsf{Succ}(\mathsf{Exp}^{\mathsf{verb}}_{\mathsf{A},\Pi}) = \Pr[\mathsf{Exp}^{\mathsf{verb}}_{\mathsf{A},\Pi}(\ell) = 1]$ *of any ppt adversary* A *is negligible as a function of* ℓ*, where* $\mathsf{Exp}^{\mathsf{verb}}_{\mathsf{A},\Pi}$ *is defined as in Figure 2.*

Roughly speaking, this definition declares a protocol verifiable if, in the presence of a malicious bulletin board (which can erase previous cast ballots and/or cast ballots on behalf of absentee voters), voters who check that their ballot has not been removed are guaranteed that their choice has been counted in the final result. Also some of the votes of honest voters who did not check might also be contained in the final result. However, their votes may as well have been dropped (but not altered to other votes). Voters under adversarial control can only vote once, with choices belonging to the choice space. The bulletin board cannot stuff itself with additional ballots without getting caught.

Experiment $\mathsf{Exp}^{\mathsf{verg}}_{\mathsf{A},\Pi}$

Adversary A has access to the oracle $\mathcal{O}\mathsf{vote}, \mathcal{O}\mathsf{corrupt}$ as before, and additionally $\mathcal{O}\mathsf{cast}$, that allows A to cast ballots to B on behalf of corrupted voters, as follows:

- $\mathcal{O}\mathsf{cast}(id, \mathsf{b})$: run $\mathsf{Board}(\tau, \mathsf{b})$.

Let HVote and Checked be the lists defined before. Recall that HVote contains entries of the form $(id, \mathsf{c}, \mathsf{b})$, that stand for honest voters' choices. The experiment outputs a bit as follows:

(1) $(\mathsf{Result}, \mathsf{P}) \leftarrow \mathsf{A}^{\mathcal{O}\mathsf{cast}, \mathcal{O}\mathsf{corrupt}, \mathcal{O}\mathsf{vote}}$

(2) if $\mathsf{Verify}(\tau, \mathsf{Result}, \mathsf{P}) = 0$ return 0

(3) if $\mathsf{Result} = \perp$ return 0

(4) if $\exists\ (i^A_1, \mathsf{c}^A_1, *), \ldots, (i^A_{n_A}, \mathsf{c}^A_{n_A}, *) \in \mathsf{HVote} \backslash \mathsf{Checked}$

$\exists\ \mathsf{c}^B_1, \ldots, \mathsf{c}^B_{n_B} \in \mathsf{C}$ s.t. $0 \le n_B \le |\mathcal{CU}|$ s.t.

$\mathsf{Result} = \rho\left(\{\mathsf{c}^E_i\}^{n_E}_{i=1}\right) \star_{\mathsf{R}} \rho\left(\{\mathsf{c}^A_i\}^{n_A}_{i=1}\right) \star_{\mathsf{R}} \rho\left(\{\mathsf{c}^B_i\}^{n_B}_{i=1}\right)$

return 0 else return 1

where $\mathsf{Checked} = \{(i^E_1, \mathsf{c}^E_1, b^E_1), \ldots, (i^E_{n_E}, \mathsf{c}^E_{n_E}, b^E_{n_E})\}$

Fig. 3: Verifiability against registrar by Cortier *et al.* [19]

C. Verifiability Against Malicious Registrar

In Helios, the bulletin board B accepts only ballots cast by eligible voters. The bulletin board B can tell apart eligible from ineligible voters generally by using some kind of authentication mechanism. In this situation, one might hope to enjoy verifiability against a dishonest registrar Reg, which is defined in Figure 3.

Definition 6 (Verifiability against malicious registrar)**.** *An election scheme achieves* verifiability against the registrar *if the success rate* $\mathsf{Succ}(\mathsf{Exp}^{\mathsf{verg}}_{\mathsf{A},\Pi}) = \Pr[\mathsf{Exp}^{\mathsf{verg}}_{\mathsf{A},\Pi}(\ell) = 1]$ *of any ppt adversary* A *is negligible as a function of* ℓ*, where* $\mathsf{Exp}^{\mathsf{verg}}_{\mathsf{A},\Pi}$ *is defined as in Figure 3.*

The intuition behind and the guarantees provided by Definition 6 are similar to those of Definition 5 except that instead of a malicious bulletin board a malicious registrar is considered, which thus can handle credentials for voters in a malicious way, i.e. provide invalid credentials or make several users share the same credentials.

D. Strong Verifiability

A protocol is said to have *strong verifiability* if it enjoys verifiability against a dishonest registrar and verifiability against a dishonest bulletin board. Intuitively, this allows one to give verifiability guarantees under a weaker trust assumption than that used in Section VI, since for strong verifiability we do not need the bulletin board and the registrar to be honest simultaneously; in Section V, it was assumed that every party has its own bulletin board, and in Section IV, no specific trust assumptions were fixed or assumed.

We note Cortier *et al.* also consider a fairness (correctness) condition similar to the ones mentioned above: the result corresponds to the votes of honest voters whenever all the parties $(\mathsf{Reg}, \mathsf{T}, \mathsf{B})$, including the voters, are honest.

Experiment $\mathsf{Exp}^{\mathsf{verw}}_{\mathsf{A},\Pi}$

Adversary A has access to the oracles $\mathcal{O}\mathsf{vote}, \mathcal{O}\mathsf{corrupt}, \mathcal{O}\mathsf{reg}$ and $\mathcal{O}\mathsf{cast}$ defined before in this section. Let HVote the list containing the intended choices of the honest voters. The experiment outputs a bit as follows:

(1) $(\mathsf{Result},\mathsf{P}) \leftarrow \mathsf{A}^{\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{corrupt},\mathcal{O}\mathsf{vote},\mathcal{O}\mathsf{cast}}$

(2) if $\mathsf{Verify}(\tau,\mathsf{Result},\mathsf{P}) = 0$ return 0

(3) if $\mathsf{Result} = \perp$ return 0

(4) if $\exists\ (i^A_1, \mathsf{c}^A_1, *), \dots, (i^A_{n_A}, \mathsf{c}^A_{n_A}, *) \in \mathsf{HVote}$
$\exists\ \mathsf{c}^B_1, \dots, \mathsf{c}^B_{n_B} \in \mathsf{C}$ s.t. $0 \le n_B \le |\mathscr{CU}|$
s.t. $\mathsf{Result} = \rho\left(\{\mathsf{c}^A_i\}^{n_A}_{i=1}\right) \star_{\mathsf{R}} \rho\left(\{\mathsf{c}^B_i\}^{n_B}_{i=1}\right)$

return 0 else return 1

Fig. 4: Weak verifiability by Cortier *et al.* [19]

E. Weak Verifiability

For weak verifiability, the trust assumptions are stronger: both the registrar Reg and the board B are assumed to be honest. This means, in particular, that B does not remove ballots, nor stuffs itself; and that Reg faithfully distributes credentials to the eligible voters. The formal definition is given in Figure 4.

Intuitively, weak verifiability guarantees that all votes that have been successfully cast are counted and that dishonest voters can only vote once; additionally only choices belonging to the choice space can be cast and counted.

F. Tally Uniqueness

As part of their definitional framework for verifiability, Cortier et al. [19] and Juels et al. [32], require the notion of *tally uniqueness*. Roughly speaking, tally uniqueness of a voting protocol ensures that the tally of an election is unique, even if all the players in the system are malicious.

More formally, the goal of the adversary against tally uniqueness is to output public election parameters $\mathsf{prm}_{\mathsf{pub}}$, a public transcript τ, two results $\mathsf{Result} \neq \mathsf{Result}'$, and corresponding proofs of valid tallying P and P′ such that both pass verification, i.e. $\mathsf{Verify}(\tau,\mathsf{Result},\mathsf{P}) = \mathsf{Verify}(\tau,\mathsf{Result}',\mathsf{P}') = 1$. A voting protocol Π has *tally uniqueness* if every ppt adversary A has a negligible advantage in this game.

Following [19], tally uniqueness ensures that, given a tally, there is at most one plausible instantiation (one-to-one property).

G. Discussion

Strong verifiability explicitly captures the situation where key players in an electronic election, such as the bulletin board or the registrar, might be corrupted and willing to alter the legitimate operation of the election. This is notably the case for Helios without identifiers (i.e. the transcript τ does not contain voters' identifiers), where a malicious B can stuff itself with ballots on behalf of absentee voters. Additionally, strong verifiability provides stronger guarantees, compared to previous definitions, to honest voters: ballots from honest voters that do not verify successfully at the end of the election can at worst be removed from the election's announced result, but *never changed*. In [19], sufficient properties for proving strong verifiability have been established.

A downside of the above definitions is that the voter's intent *is not* captured by the oracle $\mathcal{O}\mathsf{vote}(id,\mathsf{c})$, as this oracle simply performs the honest voting algorithm. In reality, voters typically use some VSD, which might be corrupted. Additionally, since Cortier et al. require that the adversary wins the game (i.e., successfully cheats) with at most negligible probability, ballot audit checks, such as Benaloh's audits[5] [11], are deemed non-verifiable as these checks may fail with non-negligible probability. Another weak point, although less important than the previous ones, is that this framework assumes that the result function ρ admits partial tallying, which is commonly the case, but it is, for example, not applicable to voting protocols which use the majority function as the result function.

H. Casting in the KTV Framework

Protocol P_{CGGI}. The set of agents Σ consists of the voters, the bulletin board B, the registrar Reg, the teller T, judge J, the scheduler, and the remaining participants. As usually, we assume that the judge and the scheduler cannot be corrupted (they ignore the corrupt message). As in the definition of Cortier et al., Reg and B can be corrupted statically, i.e., they accept the corrupt message at the beginning of a run only. Voters can be corrupted dynamically.

When the voter V runs her honest program π_{V}, she expects a candidate c, a credential pair upk, usk as input (if the input is empty, she stops). After that, she reads the election parameters $\mathsf{prm}_{\mathsf{pub}}$ and C from the bulletin board B (if she cannot find any election paramaters on B, she stops). Then, she runs $\mathsf{Vote}(\mathsf{prm}_{\mathsf{pub}},\mathsf{c},\mathsf{upk},\mathsf{usk})$ and sends the result b to the bulletin board B. Once the election is closed, she reads the content of the bulletin board and checks whether her ballot has been properly handled by the ballot box by running $\mathsf{VerifyVote}(\tau,\mathsf{upk},\mathsf{usk},\mathsf{b})$. If not, the voters send her complaint to the judge. The program of the judge accepts a run, if it does not receive any complaint from a voter and the procedure $\mathsf{Verify}(\tau,\mathsf{Result},\mathsf{P})$ returns 1.

When the registrar Reg runs the honest program π_R, it generates and distributes secret credentials to voters and registers the corresponding public credentials in the bulletin board.

When the teller T runs its honest program π_{T}, it reads the public transcript τ and runs $(\mathsf{Result},\mathsf{P}) \leftarrow \mathsf{Tally}(\tau,\mathsf{sk})$, with the election private key sk. The transcript is updated to $\tau' = \tau||\mathsf{Result}||\mathsf{P}$.

Strong verifiability. We define the goal γ_{SV} to be the set of all runs of P_{CGGI} in which either (a) both Reg and B are corrupted, (b) the result is not output, or (c) the result r of the election is defined and satisfies:

$$r = \rho\left(\{\mathsf{c}^E_i\}^{n_E}_{i=1}\right) \star_{\mathsf{R}} \rho\left(\{\mathsf{c}^A_i\}^{n_A}_{i=1}\right) \star_{\mathsf{R}} \rho\left(\{\mathsf{c}^B_i\}^{n_B}_{i=1}\right)$$

for some n_E, n_A, n_B and some $\mathsf{c}^E_i, \mathsf{c}^A_i, \mathsf{c}^B_i$ such that

- $\mathsf{c}^E_1, \dots, \mathsf{c}^E_{n_E}$ are the choices read by honest voters that successfully checked their ballots at the end of the election (and report it to the judge).
- $w_1, \dots, w_{m_A}$ are the candidates read by honest voters that did not check their ballots and $\{\mathsf{c}^A_i\}^{n_A}_{i=1} \subseteq \{w_j\}^{m_A}_{j=1}$;
- $\mathsf{c}^B_1, \dots, \mathsf{c}^B_{n_b} \in \mathsf{C}$ and n_b is smaller then the number of voters running a dishonest program.

[5] In these audits the voter can decide to cast or to audit a ballot created by her VSD. If she decides to audit the ballot, she can check whether it actually encodes her choice.

Note that, according to the above definition, if both the registrar and the bulletin board are corrupted, then the goal is trivially achieved, as we do not expect to provide any guarantees in this case.

For the protocol P_{CGGI}, strong verifiability by Cortier *et al.* can essentially be characterized by the fact that it is (γ_{SV}, δ)-verifiable by the judge J in the sense of Definition 1, for $\delta = 0$.

Let us emphasize that this goal ensures that votes of honest voters who do not verify at the end of the election are at most dropped, but not changed. This is in contrast to the goals we have seen so far. In these goals, votes of honest voters who do not verify might have been tampered with.

Weak verifiability. We define the goal γ_{WV} to be the set of all runs of P_{CGGI} in which either (a) either Reg or B is corrupted, (b) the result is not output, or (c) the result r of the election is defined and satisfies:

$$r = \rho\left(\{\mathsf{c}_i^A\}_{i=1}^{n_A}\right) \star_{\mathsf{R}} \rho\left(\{\mathsf{c}_i^B\}_{i=1}^{n_B}\right)$$

for some n_A, n_B and some $\mathsf{c}_i^A, \mathsf{c}_i^B$ such that

- $\mathsf{c}_1^A, \ldots, \mathsf{c}_{n_A}^A$ are the candidates read by honest voters that cast their votes;
- $\mathsf{c}_1^B, \ldots, \mathsf{c}_{n_b}^B \in \mathsf{C}$ and n_b is smaller then the number of voters running a dishonest program.

For the protocol P_{CGGI}, weak verifiability by Cortier *et al.* can essentially be characterized by the fact that it is (γ_{WV}, δ)-verifiable in the sense of Definition 1.

Note that Item (c) of the goal γ_{WV} is stronger than the corresponding item of γ_{SV} (since all honest cast votes shall be counted). However, the latter is called *weak verifiability* in [19] because the trust assumptions (Item (a)) are stronger (both the ballot box and the registrar shall be honest).

VIII. Computational Election Verifiability by Smyth et al.

This section focuses on the definitions of individual, universal and election verifiability by Smyth et al. [47]. Smyth et al. consider two different verifiability settings, one for election schemes with *external* and the other one for election schemes with *internal* authentication (such as Helios and Civitas, respectively). For the sake of brevity, we focus on election schemes with external authentication because the issues discussed in Section VIII-E apply to both of them. We present the casting of the definitions of Smyth et al. in the KTV framework in Appendix C.

A. Model

According to Smyth et al., an election scheme Π is a tuple (Setup, Vote, Tally, Verify) of probabilistic polynomial-time algorithms. The algorithms Setup and Vote are defined as usual. The algorithm Tally is run by the tellers and receives the content of the bulletin board B and the parameters prm as input, and outputs the tally along with a non-interactive proof P for the correctness of the tally. The algorithm Verify describes the verification of the election result and receives the content of the bulletin board B, the public parameters $\mathsf{prm_{pub}}$, the tally, denoted by tally, and a proof P, and outputs a bit. The algorithm Verify is deterministic.

Experiment $\mathsf{ExpIV}(\Pi, \mathsf{A})$

(1) $(\mathsf{prm_{pub}}, \mathsf{c}, \mathsf{c}') \longleftarrow \mathsf{A}$
(2) $\mathsf{b} \longleftarrow \mathsf{Vote}(\mathsf{c}, \mathsf{prm_{pub}})$
(3) $\mathsf{b}' \longleftarrow \mathsf{Vote}(\mathsf{c}', \mathsf{prm_{pub}})$
(4) if $\mathsf{b} = \mathsf{b}'$ and $\mathsf{b} \neq \perp$ and $\mathsf{b}' \neq \perp$ then
return 1 else return 0

Fig. 5: Individual verifiability experiment by Smyth et al. [47]

Experiment $\mathsf{ExpUV}(\Pi, \mathsf{A})$

(1) $(\mathsf{B}, \mathsf{prm_{pub}}, \mathsf{tally}', \mathsf{P}') \longleftarrow \mathsf{A}$
(2) $\mathsf{tally} \longleftarrow \mathsf{correct\ tally}(\mathsf{B}, \mathsf{prm_{pub}})$
(3) if $\mathsf{tally} \neq \mathsf{tally}'$ and $\mathsf{Verify}(\mathsf{B}, \mathsf{prm_{pub}}, \mathsf{tally}', \mathsf{P}')$ then
return 1 else return 0

Fig. 6: Universal verifiability experiment by Smyth et al. [47]

B. Individual Verifiability

According to Smyth et al., an election scheme achieves individual verifiability if, for any two honest voters, the probability that their ballots are equal is negligible, which formally is expressed as follows.

Definition 7 (Individual verifiability)**.** *An election scheme* $\Pi =$ (Setup, Vote, Tally, Verify) *achieves* individual verifiability *if the success rate* $\mathsf{Succ}(\mathsf{ExpIV}(\Pi, \mathsf{A}))$ *of any ppt adversary* A *in Experiment* $\mathsf{ExpIV}(\Pi, \mathsf{A})$ *(Fig. 5) is negligible as a function of* ℓ.

C. Universal Verifiability

According to Smyth et al., an election scheme achieves universal verifiability if no ppt adversary A can simulate a tallying phase such that, on the one hand, the verification algorithm Verify accepts the output (e.g., all zero-knowledge proofs are successful), and, on the other hand, the given output of the tallying phase does not agree with what Smyth et al. call the *correct tally*.

The function correct tally, defined as follows, extracts the actual votes from the ballots on the bulletin board.

Definition 8 (Correct Tally)**.** *The function* correct tally *maps each tuple* $(\mathsf{B}, \mathsf{prm_{pub}})$ *to a vector in* $\{0, \ldots, \mathsf{n_{ballots}}\}^{\mathsf{n_{cand}}}$ *such that for every choice* $\mathsf{c} \in \{1, \ldots, \mathsf{n_{cand}}\}$ *and every number* $l \in \{0, \ldots, \mathsf{n_{ballots}}\}$ *we have that* $\mathsf{correct\ tally}(\mathsf{B}, \mathsf{prm_{pub}})[\mathsf{c}] = l$ *if and only if there are exactly* l *different ballots* b $(\neq \perp)$ *on the bulletin board* B *and for each of them there exists a random bit string* r *such that* $\mathsf{b} = \mathsf{Vote}(\mathsf{c}, \mathsf{prm_{pub}}; r)$.

Now, universal verifiability is defined as follows according to Smyth et al.

Definition 9 (Universal verifiability)**.** *An election scheme* (Setup, Vote, Tally, Verify) *achieves* universal verifiability *if the success rate* $\mathsf{Succ}(\mathsf{ExpUV}(\Pi, \mathsf{A}))$ *of every ppt adversary* A *in Experiment* $\mathsf{ExpUV}(\Pi, \mathsf{A})$ *(Fig. 6) is negligible as a function of* ℓ.

D. Election Verifiability

The notion of verifiability proposed by Smyth et al. now combines the notions of individual and universal verifiability.

Definition 10 (Election Verifiability)**.** *An election scheme* (Setup, Vote, Tally, Verify) *satisfies* election verifiability *if for every ppt adversaries* A*, there exists a negligible function* μ *such that for all security parameters* ℓ*, we have that*

$$\mathsf{Succ}(\mathsf{ExpIV}(\Pi,\mathsf{A}))+\mathsf{Succ}(\mathsf{ExpUV}(\Pi,\mathsf{A}))\le\mu(\ell).$$

Smyth et al. also consider some soundness properties, including fairness and correctness, similar to the ones mentioned in previous sections.

E. Discussion

This definition has two main shortcomings. First, as stated by the authors, their "definitions of verifiability have not addressed the issue of voter intent, that is, whether the ballot constructed by the Vote algorithm corresponds to the candidate choice that a voter intended to make." (Page 12, [47]). In general, it is not clear that the combination of the proposed definitions of verifiability along with additional soundness properties implies any form of end-to-end verifiability. More precisely, if all the verification procedures succeed, it is unclear whether the final outcome of an election corresponds to the voters' choices at least with reasonable probability.[6] We think, however, that capturing such overall correctness and the voter's intent is at the very core of a meaningful notion of verifiability.

Second, the definition considers a restricted class of protocols (the authors themselves provide a list of protocols excluded by their definition), some of these restrictions, as pointed out before, also apply to some of the other definitions discussed in this paper: (1) The model captures "single-pass" protocols only: voters send a single ballot to the election server, without any further interaction. (2) The authors assume that the whole ballot is published. (3) The authors assume that the vote can be recovered directly from the ballot, which excludes protocols using information-theoretically hiding commitments. (4) There is no revote. (5) The bulletin board publishes the list of ballots, as received. And hence, voting schemes such as ThreeBallot [45] cannot be modeled.

As mentioned before, the casting of the Smyth et al. definitions in the KTV framework is presented in Appendix C.

IX. Further Related Work

Since the focus of this paper is on verifiability notions that have been formally defined, we excluded those verifiability notions from our analysis which do not fulfill this requirement ([46], [30], [51], [43], [41], [42]). An important paper is the one by Sako and Kilian [46] who were the first to propose the concept of individual and universal verifiability. This then motivated other researchers to regard end-to-end verifiability as the sum of certain verifiability subproperties; we discuss this issue in Section X.

Due to space constraints, a few formal definitions of verifiability are discussed and cast in the KTV framework in the appendix or in the full version [20] of this paper only. We briefly discuss them here.

Kremer et al. [35] (Appendix A) and Cortier et al. [18] (Appendix B) define verifiability in symbolic models, where messages are modeled by terms. Kremer et al. propose a definition that corresponds to γ_0 but under the trust assumption that every voter is honest and verifies the final result, which is clearly too strong. Cortier et al. [18] devise formulations for *individual verifiability*, *universal verifiability*, and *no clash* (two honest ballots should never collude), and they show that these three properties imply what they call *end-2-end verifiability*, the latter being close to the goal γ_{SV} (introduced in Section VII), except that ballot stuffing is not prohibited.

In the full version of this paper [20], we also analyze the definition by Baum et al. [8] (Szepieniec et al. [49] proposed a closely related definition), the one by Chevallier-Mames et al. [16], and by Hosp et al. [28]. The definition by Baum et al. (auditable correctness) can be applied to arbitrary multi-party computation (MPC) protocols and is based on an ideal functionality in the Universal Composability (UC) framework. In the context of e-voting protocols, the goal of this definition is γ_0. Baum et al. also consider a very (in fact too) strong fairness condition: auditors have to always accept a protocol run if the goal γ_0 is achieved, regardless of whether, for example, zero-knowledge proofs are valid or not. As for the definition by Chevallier-Mames et al., it captures universal verifiability, and hence, a subproperty of end-to-end verifiability only. Hosp et al. propose information-theoretic measures for the verifiability of voting systems, by comparing these systems to perfect voting systems which always output the correct result, independently of voters being honest or dishonest. This definition is even much stronger than what is required by γ_0, and therefore, does not seem to be applicable to any practical voting protocol.

X. Summary and Conclusion

In the previous sections, we have studied the formal definitions of verifiability for e-voting system proposed in the literature. We have presented the original definitions and cast them in the KTV framework. This casting has demonstrated that the essence of these notions can be captured within a uniform framework and enabled us to identify their relative and recurrent merits and weaknesses as well as their specific (partly severe) limitations and problems.

In Section X-A, we distill these discussions and insights into detailed requirements and guidelines that highlight several aspects any verifiability definition should cover. We also summarize from the previous sections how the different existing definitions of verifiability from the literature handle these aspects, with a brief overview for some of the aspects provided in Table I. Finally, in Section X-B, as a viable and concrete embodiment of our guidelines, we instantiate the KTV framework accordingly, obtaining a solid and ready to use definition of verifiability.

A. Guidelines

We now present our requirements and guidelines for the following central aspects, along with a summary of the previous sections concerning these aspects.

Generality. Many verifiability definitions are designed for protocols with specific protocol structures and are tailored to them (see Sections VI, VII, VIII and Appendix A, B). As a

[6] It indeed seems that this is not the case due to some technical issues: Their correctness property requires only that Vote correctly encodes the given choice in the case of the *honest setup*; it does not guarantee anything for the dishonest setup which is considered in the verifiability games. Therefore, if, for instance, Vote always produces $\perp$ (an invalid ballot) for some dishonestly generated public key, the system can still be proved verifiable according to the definition of Smyth et al., although it clearly produces a wrong election outcome. This particular technicality seems to be easy to fix, but it nevertheless demonstrates that there is some gap between the given combination of disconnected properties and an overarching and meaningful verifiability notion.

TABLE I: Overview of Verifiability Notions

Notion (Section & Paper)	**Verifiability goal** (Intuition)	**Verifiability tolerance**	**General trust assumptions**	**Protocol classes**
Verifiability (IV, [37])	Flexible γ, with γ_k ($k \geq 0$) being one example	Yes	Flexible	No specific structure required.
Verifiability (V, [12])	γ_0^* (The votes of all eligible (honest and dishonest) voters who submit valid ballots are counted.)	Yes	$\mathsf{hon}(\mathsf{B})$	Assumes personal bulletin board for each protocol participant. Only "yes" or "no" choices possible. Otherwise no specific protocol structure.
E2E verifiability (VI, [34])	$\gamma_{\theta,k,\mathsf{Extr}}$ ($\theta, k \geq 0$) (Either (i) the published result differs on less than k positions from the correct result (as extracted by Extr), or (ii) less than θ many honest voters successfully cast their ballots, or (iii) at least one of these honest voters complains if she verifies the final result.)	Yes	$\mathsf{hon}(\mathsf{B})$	Assumes specific (Setup, Cast, Tally, Result, Verify) protocol structure. Requires extraction property.
Strong verifiability (VII, [19])	γ_{SV} (If Reg or B are honest, then (i) the votes of all honest voters who check are counted, and (ii) further honest votes can only be dropped (not manipulated), and (iii) only votes of eligible voters are counted.)	No	Flexible	Assumes specific (Setup, $\mathsf{Credential}$, Vote, $\mathsf{VerifyVote}$, Valid, Board, Tally, Verify) protocol structure. Tally function with partial tallying property.
Weak verifiability (VII, [19])	γ_{WV} (If Reg and B are honest, then γ_0 is achieved.)	No		
Individual verifiability (VIII, [47])	γ_{IV} (All honest voters' valid ballots are pairwise different.)	No	$\mathsf{hon}(\mathsf{B})$	Assumes specific (Setup, Vote, Tally, Verify) protocol structure. Requires extraction property.
Universal verifiability (VIII, [47])	γ_{UV} (($\mathsf{Tally}, \mathsf{P}$) published on B and $\mathsf{Tally} = \mathsf{correct\ tally}(\mathsf{B})$)	No		
Election verifiability (VIII, [47])	$\gamma_{IV} \wedge \gamma_{UV}$	No		
Individual and universal verifiability (A, [35])	γ_{IUV} (γ_0 and all honest voters' ballots are pairwise different.)	No	$\mathsf{hon}(\mathsf{B}) \wedge \bigwedge_{i=1}^{n} \mathsf{hon}(\mathsf{V}_i)$	Requires bulletin board B. Assumes that all voters verify. Otherwise no specific protocol structure.
Individual verifiability (B, [18])	γ_{IV} (The ballots of all honest voter who check are on B.)	No	$\mathsf{hon}(\mathsf{B})$	Assumes specific structure and extraction property for the definition of individual and universal verifiability. Requires bulletin board B for the E2E verifiability definition, otherwise no specific protocol structure.
Universal verifiability (B, [18])	γ_{UV} (The votes of all honest voters whose ballots are on B are counted. Non-eligible voters are allowed.)	No		
E2E verifiability (B, [18])	γ_{E2E} (The votes of all honest voter who check are counted. Non-eligible voters are allowed.)	No		

Table description: We group associated definitions. For each goal we have extracted from the definition, we include a short informal description. The third column ("Verifiability tolerance") states whether or not the associated verifiability definition allows for some tolerance: "Yes" if $\delta \geq 0$ is allowed, "No" if $\delta = 0$ is required, with δ as in Definition 1. The fourth column ("General trust assumptions") describes which protocol participants are assumed to be always honest (besides the judge and the scheduler) and in the fifth column ("Protocol classes") requirements on the protocol structure are listed, where *extraction property* is the requirement that single ballots and their content (i.e. the plain vote) can be extracted from the bulletin board B.

result, for new classes of protocols often new definitions are necessary.

Clearly, it is desirable for a verifiability definition to be applicable to as many protocols as possible. It provides not only reusability, but also comparability: by applying the same definition to different protocols and protocol classes we can clearly see the differences in the level and nature of verifiability they provide. A very minimal set of assumptions on the protocol structure is sufficient to express a meaningful notion of verifiability, as illustrated by the definition in Section IV and also by the instantiation of the KTV framework given below.

Note, however, that some additional assumptions on the protocol structure allow one to express some specific properties, such as universal verifiability, which, as discussed in the previous sections, on their own do not capture end-to-end verifiability, but may be seen as valuable additions.

Static versus dynamic corruption. We observe that most of the studied verifiability definitions focus on static corruption, except the definitions in Sections VI and VII, which capture the dynamic corruption of voters. In general, modeling dynamic corruption can yield stronger security guarantees. In the context of verifiability, one could, for example, provide guarantees not only to honest voters but also to certain corrupted voters. If a voter is corrupted only late in the election, e.g., when the voting phase, one might still want to guarantee that her vote is counted. None of the existing definitions provide this kind of guarantee so far. We briefly discuss how this can be captured in the KTV framework in Section X-B.

Binary versus quantitative verifiability. As discussed in Section III-B, the probability δ (see Definition 1) that under realistic assumptions some cheating by an adversary remains undetected may be bigger than 0 even for reasonable protocols: often some kind of partial and/or probabilistic checking is carried out, with Benaloh audits (see Section VII-G) being an example. These checks might fail to detect manipulations with some non-negligible probability. Still, as we have seen when casting the different verifiability notions in the KTV framework, most of the studied definitions assume the verifiability tolerance to be $\delta = 0$. This yields a binary notion of verifiability which, as explained, outright rejects reasonable protocols.

In contrast, the definitions studied in the KTV framework (including Section IV) as well as the ones in Sections V and VI , allow for measuring the level of verifiability. This gives more expressiveness and allows one to establish meaningful verifiability results for (reasonable) protocols which do not provide perfect verifiability.

Goals. As pointed out in Section IV, the goal γ_0, which, among others, requires that all the ballots cast by honest voters are correctly tallied and make it to the final result is very strong and typically too strong. In order to satisfy this goal very strong trust assumptions are necessary, for instance, the assumptions taken in the definition of weak verifiability in Section VII.

From the previous sections, two main and reasonable approaches for defining a goal emerged, which one could characterize as quantitative and qualitative, respectively:

Quantitative. In Section IV, a family of goals γ_k, $k \geq 0$, together with a non-zero tolerance level δ is considered; a similar approach is taken in Section VI, but see the discussion in this section. This approach, among others, captures that the probability that more than k votes of honest voters can be changed without anybody noticing should be small, i.e., bounded by δ. To be more precise and allow for stronger guarantees, this approach could be combined with an aspect of the goal defined for strong verifiability, namely the distinction between votes that are manipulated and those that are "just" dropped (see Section VII).

Qualitative. In Section VII ("strong verifiability"), the protocol goal (as cast in the KTV framework), among others, stipulates that votes of voters who verify their receipt are contained in the final result. To be general, this approach should also be combined with a non-zero tolerance level δ (which, however, was not captured in the original definition). The reason is that checks (such as Benaloh challenges) might not be perfect, i.e., manipulation might go undetected with some probability.

In both cases, votes of dishonest voters were restricted to be counted at most once (no ballot stuffing).

The quantitative approach, on the one hand, provides overall guarantees about the deviation of the published result from the correct one and measures the probability δ that the deviation is too big (bigger than k) but nobody notices this. On the other hand, it does not explicitly require that voters who check their receipts can be sure (up to some probability) that their votes were counted. But, of course, to prove verifiability of a system w.r.t. this goal, one has to take into account whether or not voters checked, and more precisely, the probabilities thereof. These probabilities also capture the uncertainty of the adversary about whether or not specific voters check, and by this, provides protection even for voters who do not check.

The qualitative approach explicitly provides guarantees for those honest voters who verify their receipts. On the one hand, this has the advantage that one does not need to consider probabilities of voters checking or not, which simplifies the analysis of systems. On the other hand, such probabilities of course play an important role for measuring the overall security of a system, an aspect this simpler approach abstracts away. Nevertheless, it provides a good qualitative assessment of a system.

Interestingly, one could in principle combine both approaches, i.e., consider the intersection of both goals. While this would give voters also in the quantitative approach direct guarantees (in addition to the aspect of making a distinction between manipulating and dropping votes, mentioned above already), it would typically not really change the analysis and its result: as mentioned, in the quantitative analysis one would anyway have to analyze and take into account the guarantees offered when checking receipts.

Below, we provide concrete instantiations for both approaches in the KTV framework.

Ballot stuffing. Not all definitions of verifiabiltiy rule out ballot stuffing, even though ballot stuffing, if unnoticed, can dramatically change the election result. Some definitions go even further and abstract away from this problem by assuming that there are only honest voters (see trust assumptions below).

Clearly, allowing undetected ballot stuffing makes a verifiability definition too weak. We recommend that a verifiability definition should exclude undetected ballot stuffing. It might also be useful to capture different levels of ballot stuffing in order to distinguish systems where it is very risky to add even a small number of ballots from those where adding such a small number is relatively safe. The goals discussed above, as mentioned, both require that no ballot stuffing is possible at all.

Trust assumptions. Some verifiability definitions assume some protocol participants to be always honest, for example the bulletin board (Sections V, VI, VIII, Appendix A, B), or all voters (Appendix A) or all voter supporting devices (Sections VIII, VII), or some disjunctions of participants (Section VII); the definition discussed in Section IV does not make such assumptions. We think that verifiability definitions which rely on the unrealistic assumption that all voters are honest are too weak. The other trust assumptions might be reasonable depending on the threat scenario. A general verifiability definition should be capable of expressing different trust assumptions and make them explicit; embedding trust assumptions into a definition not only makes the definition less general, but also makes the assumptions more implicit, and hence, easy to overlook.

Individual and universal verifiability. In Section VIII and Appendix B, definitions of individual and universal verifiability were presented. We already pointed out that the split-up of end-to-end verifiability into sub-properties is problematic. In fact, Küsters et al. [39] have proven that, in general, individual and universal verifiability (even assuming that only eligible voters vote) do not imply end-to-end verifiability, e.g. for ThreeBallot [45]. For the definitions of individual and universal verifiability presented in Section VII, it was shown in [18] that they imply end-to-end verifiability under the assumption that there are no clashes [39]. However, the notion of end-to-end verifiability considered there is too weak since it allows ballot stuffing. For the definitions of individual and universal verifiability in Section VIII no such proof was provided, and therefore, it remains unclear whether it implies end-to-end verifiability. (In fact, technically these definitions, without some fixes applied, do not provide end-to-end verifiability as pointed out in Section VIII.)

The (combination of) notions of individual and universal verifiability (and other properties and subproperties, such as eligibility verifiability, cast-as-intended, recorded-as-cast, and counted-as-recorded) should not be used as a replacement for end-to-end verifiability per se since they capture only specific aspects rather than the full picture. Unless formally proven that their combination in fact implies end-to-end verifiability they might miss important aspects, as discussed above. Therefore, the security analysis of e-voting systems should be based on the notion of end-to-end verifiability (as, for example, concretely defined below). Subproperties could then possibly be used as useful proof techniques.

B. Exemplified Instantiation of the Guideline

We now demonstrate how the guidelines given above can be put into practice, using, as an example, the KTV framework. By this, we obtain a solid, ready-to-use definition of verifiability. More specifically, we propose two variants, one for qualitative and one for quantitative reasoning, as explained next.

The distillation of our observations given in Section X-A reviews different aspects of verifiability and, in most cases, it clearly identifies the best and favorable ways they should be handled by verifiability definitions. When it comes to the distinction between qualitative and quantitative approaches to define verifiability goals, we have, however, found out that both approaches have merits and both can yield viable definitions of verifiability. This is why we propose two instantiations of the KTV framework, one following the qualitative approach and one for the quantitative approach.

To instantiate the KTV framework, one only has to provide a definition of a goal (a family of goals) that a protocol is supposed to guarantee. Note that, as for the second parameter of Definition 1, δ, one should always try, for a given goal, to establish an as small δ as possible. In other words, the value of δ is the result of the analysis of a concrete system, rather than something fixed up front.

In the following, we define two goals corresponding to the two variants of verifiability discussed above: goal $\gamma_{ql}(\varphi)$ for the qualitative variant and goal $\gamma_{qn}(k,\varphi)$ for the quantitative one. We explain the meaning of the parameters below. Here we only remark that the common parameter φ describes the trust assumptions (i.e., it determines which parties are assumed to be honest and which can be corrupted and when) under which the protocol is supposed to provide specific guarantees. Recall that, in the KTV framework, the adversary sends a special message $\mathsf{corrupt}$ to a participant in order to corrupt it (a participant can then accept or reject such a message). This allows for modeling various forms of static and dynamic corruption. Note also that it is easily visible, given a run, if and when a party is corrupted.

In the following, for a given run r of an e-voting protocol with n eligible voters, we denote by n_h the number of honest and by n_d the number of dishonest voters in r. Recall that we say a party is honest in a run r if it has not received a $\mathsf{corrupt}$ message or at least has not accepted such a message throughout the whole run. We denote by $\mathsf{c}_1,\ldots,\mathsf{c}_{n_h}$ the actual choices of the honest voters in this run (which might include abstention), as defined in Section IV-A.

Qualitative goal. The goal $\gamma_{ql}(\varphi)$ we define here corresponds to the strong verifiability goal γ_{SV} from Section VII. In contrast to γ_{SV}, $\gamma_{ql}(\varphi)$ has the parameter φ for the trust assumptions, which were fixed in γ_{SV}. Informally, this goal requires that, if the trust assumption φ holds true in a protocol run, then (i) the choices of all honest voters who successfully performed their checks are included in the final result, (ii) votes of those honest voters who did not performed their check may be dropped, but not altered, and (iii) there is only at most one ballot cast for every dishonest voter (no ballot stuffing). If the trust assumptions φ are not met in a protocol run, we do not expect the protocol to provide any guarantees in this run. For example, if in a setting with two bulletin boards, φ says that at least one of the bulletin boards should be honest in a run, but in the run considered both have been corrupted by the adversary, then no guarantees need to be provided in this run.

Formally, the goal $\gamma_{ql}(\varphi)$ is satisfied in r (i.e., $r \in \gamma_{ql}(\varphi)$) if either (a) the trust assumption φ does not hold true in r, or if (b) φ holds true in r and there exist valid choices $\tilde{\mathsf{c}}_1,\ldots,\tilde{\mathsf{c}}_n$ for which the following conditions are satisfied:

(i) An election result is published in r and it is equal to $\rho(\tilde{\mathsf{c}}_1,\ldots,\tilde{\mathsf{c}}_n)$.
(ii) The multiset $\{\tilde{\mathsf{c}}_1,\ldots,\tilde{\mathsf{c}}_n\}$ consists of all the actual choices of honest voters who successfully performed their check, plus a subset of actual choices of honest voters who did not perform their check (successfully), and plus at most n_d additional choices.

If the checks performed by voters do not fully guarantee that their votes are actually counted, because, for example, Benaloh checks were performed (and hence, some probabilistic checking), then along with this goal one will obtain a $\delta > 0$, as there is some probability for cheating going undetected. Also, the requirement that votes of honest voters who do not checked can at most be dropped, but not altered, might only be achievable under certain trust assumptions. If one wants to make weaker trust assumptions, one would have to weaken $\gamma_{ql}(\varphi)$ accordingly.

Quantitative goal. The goal $\gamma_{qn}(k,\varphi)$ of the quantitative verifiability definition is a refinement of the goal γ_k from Section IV (note that now, φ can specify trust assumption with dynamic corruption). Similarly to Section VI, we use a distance function on election results. Roughly, the goal $\gamma_{qn}(k,\varphi)$ requires that the distance between the produced result and the "ideal" one (obtained when the actual choices of honest voters are counted and one choice for every dishonest voter) is bounded by k, where, for $\gamma_{qn}(k,\varphi)$, we consider a specific distance function d. In order to define d, we first define a function $f_{\mathsf{count}}\colon \mathsf{C}^l \to \mathbb{N}^{\mathsf{C}}$ which, for a vector $(\mathsf{c}_1,\ldots,\mathsf{c}_l) \in \mathsf{C}^l$ (representing a multiset of voters' choices), counts how many times each choice occurs in this vector. For example, $f_{\mathsf{count}}(B,C,C)$ asigns 1 to B, 2 to C, and 0 to all the remaining choices. Now, for two vectors of choices $\vec{\mathsf{c}},\vec{\mathsf{c}}'$ the distance function d is defined by

$$d(\vec{\mathsf{c}},\vec{\mathsf{c}}') = \sum_{\mathsf{c}\in\mathsf{C}} |f_{\mathsf{count}}(\vec{\mathsf{c}})[\mathsf{c}] - f_{\mathsf{count}}(\vec{\mathsf{c}}')[\mathsf{c}]|.$$

For instance, $d((B,C,C),(A,C,C,C)) = 3$.

Now, the goal $\gamma_{qn}(k,\varphi)$ is satisfied in r if either (a) the trust assumption φ does not hold true in r, or if (b) φ holds true in r and there exist valid choices $\mathsf{c}'_1,\ldots,\mathsf{c}'_{n_d}$ (representing possible choices of dishonest voters) and $\tilde{\mathsf{c}}_1,\ldots,\tilde{\mathsf{c}}_n$, such that:

(i) an election result is published and it is equal to $\rho(\tilde{\mathsf{c}}_1,\ldots,\tilde{\mathsf{c}}_n)$, and
(ii) $d((\mathsf{c}_1,\ldots,\mathsf{c}_{n_h},\mathsf{c}'_1,\ldots,\mathsf{c}'_{n_d}),(\tilde{\mathsf{c}}_1,\ldots,\tilde{\mathsf{c}}_n)) \leq k$.

Note that when an adversary *drops* one honest vote, this increases the distance in Condition (ii) by one, but when he *replaces* an honest voter's choice by another one, this increases the distance by two. This corresponds to the real effect of a manipulation on the final result (goal γ_k does not distinguish between these two types of manipulations).

As already explained, since not all voters will check their receipts, some manipulation will go undetected. And hence, for this goal $\delta = 0$ is typically not achievable. The security analysis carried out on a concrete protocol will have to determine the optimal (i.e., minimal) δ, given the parameter k.

We finally note that both of the above goals could be refined by providing guarantees for those voters who have been corrupted sufficiently late in the protocol. For this, one merely has to change what it means for a voter to be honest: voters corrupted late enough would still be considered honest for the purpose of the above goal definitions. For such voters, one would then also provide guarantees. However, such refinements are protocol dependent, whereas the above goals are applicable to a wide range of protocols.

Acknowledgements. This work was partially supported by *Deutsche Forschungsgemeinschaft* (DFG) under Grant KU 1434/6-3 within the priority programme 1496 "Reliably Secure Software Systems – RS^3". The research leading to these results has also received funding from the *European Research Council* under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 258865 (ProSecure).

References

[1] http://www.economist.com/node/8382578, December 7th 2006.

[2] http://www.computerworld.com/s/article/9118204/Princeton_report_rips_N.J._e_voting_machines_as_easily_hackable_, October 27th 2008.

[3] http://ucsdnews.ucsd.edu/newsrel/science/08-09ElectronicVoting.asp, August 10th 2009.

[4] https://freedom-to-tinker.com/blog/appel/nj-election-cover/, September 13th 2011.

[5] M. Abadi and C. Fournet. Mobile Values, New Names, and Secure Communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL 2001)*, pages 104–115. ACM Press, 2001.

[6] Ben Adida. Helios: Web-based Open-Audit Voting. In Paul C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

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

[8] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. Cryptology ePrint Archive, Report 2014/075, 2014. http://eprint.iacr.org/.

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

[10] Jonathan Ben-Nun, Niko Fahri, Morgan Llewellyn, Ben Riva, Alon Rosen, Amnon Ta-Shma, and Douglas Wikström. A New Implementation of a Dual (Paper and Cryptographic) Voting System. In Kripp et al. [36], pages 315–329.

[11] Josh Benaloh. Simple verifiable elections. In Dan S. Wallach and Ronald L. Rivest, editors, *2006 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'06, Vancouver, BC, Canada, August 1, 2006*. USENIX Association, 2006.

[12] Josh Daniel Cohen Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, 1987.

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

[14] J. A. Calandrino, A. J. Feldman, J. A. Halderman, D. Wagner, H. Yu, and W. P. Zeller. Source Code Review of the Diebold Voting System, 2007. Report commissioned as part of the California Secretary of State's Top-To-Bottom Review of California voting systems. http://www.eecs.berkeley.edu/~daw/papers/dieboldsrc-ttbr.pdf.

[15] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, and A. T. Sherman. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. In *USENIX/ACCURATE Electronic Voting Technology (EVT 2008)*. USENIX Association, 2008. See also http://www.scantegrity.org/elections.php.

[16] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On Some Incompatible Properties of Voting Schemes. In David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida, editors, *Towards Trustworthy Elections, New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 191–199. Springer, 2010.

[17] 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)*, pages 354–368. IEEE Computer Society, 2008.

[18] Véronique Cortier, Fabienne Eigner, Steve Kremer, Matteo Maffei, and Cyrille Wiedling. Type-Based Verification of Electronic Voting Protocols. In *Proceedings of the 4th Conference on Principles of Security and Trust (POST'15)*, Lecture Notes in Computer Science, London, UK, April 2015. Springer.

[19] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachene. Election Verifiability for Helios under Weaker Trust Assumptions. In *Proceedings of the 19th European Symposium on Research in Computer Security (ESORICS'14)*, LNCS, Wroclaw, Poland, September 2014. Springer.

[20] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. SoK: Verifiability Notions for E-Voting Protocols. Technical report, Cryptology ePrint Archive, 2016. Available at http://eprint.iacr.org/.

[21] R. Cramer, R. Gennaro, and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *Advances in Cryptology — EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1233 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

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

[23] Chris Culnane and Steve A. Schneider. A Peered Bulletin Board for Robust Use in Verifiable Voting Systems. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 169–183. IEEE, 2014.

[24] Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election Verifiability or Ballot Privacy: Do We Need to Choose? In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, volume 8134 of *Lecture Notes in Computer Science*, pages 481–498. Springer, 2013.

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

[26] David Galindo, Sandra Guasch, and Jordi Puiggali. 2015 Neuchâtel's Cast-as-Intended Verification Mechanism. In Rolf Haenni, Reto E. Koenig, and Douglas Wikström, editors, *E-Voting and Identity - 5th International Conference, VoteID 2015, Bern, Switzerland, September 2-4, 2015, Proceedings*, volume 9269 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015.

[27] Gurchetan S. Grewal, Mark Dermot Ryan, Liqun Chen, and Michael R. Clarkson. Du-Vote: Remote Electronic Voting with Untrusted Computers. In Cédric Fournet, Michael W. Hicks, and Luca Viganò, editors, *IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015*, pages 155–169. IEEE, 2015.

[28] Benjamin Hosp and Poorvi L. Vora. An information-theoretic model of voting systems. In David Chaum, Miroslaw Kutylowski, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Frontiers of Electronic Voting, 29.07. - 03.08.2007*, volume 07311 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[29] David Jefferson, Aviel D. Rubin, Barbara Simons, and David Wagner. Analyzing internet voting security. *Communications of the ACM, Special*

issue: The problems and potentials of voting systems*, 47(10):59–64, 2004.

[30] Hugo Jonker, Sjouke Mauw, and Jun Pang. Privacy and verifiability in voting systems: Methods, developments and trends. *Computer Science Review*, 10:1–30, 2013.

[31] A. Juels, D. Catalano, and M. Jakobsson. Coercion-Resistant Electronic Elections. In *Proceedings of Workshop on Privacy in the Eletronic Society (WPES 2005)*, pages 61–70. ACM Press, 2005.

[32] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida, editors, *Towards Trustworthy Elections, New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 37–63. Springer, 2010.

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

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

[35] Steve Kremer, Mark Ryan, and Ben Smyth. Election Verifiability in Electronic Voting Protocols. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *15th European Symposium on Research in Computer Security (ESORICS2010)*, volume 6345 of *Lecture Notes in Computer Science*, pages 389–404. Springer, 2010.

[36] Manuel J. Kripp, Melanie Volkamer, and Rüdiger Grimm, editors. *5th International Conference on Electronic Voting 2012, (EVOTE 2012), Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC, July 11-14, 2012, Castle Hofen, Bregenz, Austria*, volume 205 of *LNI*. GI, 2012.

[37] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and Relationship to Verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2010)*, pages 526–535. ACM, 2010. Full version available at http://eprint.iacr.org/2010/236/.

[38] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In *32nd IEEE Symposium on Security and Privacy (S&P 2011)*, pages 538–553. IEEE Computer Society, 2011.

[39] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Clash Attacks on the Verifiability of E-Voting Systems. In *33rd IEEE Symposium on Security and Privacy (S&P 2012)*, pages 395–409. IEEE Computer Society, 2012.

[40] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Formal Analysis of Chaumian Mix Nets with Randomized Partial Checking. In *35th IEEE Symposium on Security and Privacy (S&P 2014)*, pages 343–358. IEEE Computer Society, 2014.

[41] Lucie Langer. *Privacy and verifiability in electronic voting*. PhD thesis, Darmstadt University of Technology, 2010.

[42] Thea Peacock, Peter YA Ryan, Steve Schneider, and Zhe Xia. Verifiable voting systems. *Computer and Information Security Handbook*, pages 1103–1125, 2013.

[43] Stefan Popoveniuc, John Kelsey, Andrew Regenscheid, and Poorvi L. Vora. Performance Requirements for End-to-End Verifiable Elections. In Douglas W. Jones, Jean-Jacques Quisquater, and Eric Rescorla, editors, *2010 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '10, Washington, D.C., USA, August 9-10, 2010*. USENIX Association, 2010.

[44] Jordi Puigalli and Sandra Guasch. Cast-as-Intended Verification in Norway. In Kripp et al. [36], pages 49–63.

[45] R. L. Rivest and W. D. Smith. Three Voting Protocols: ThreeBallot, VAV and Twin. In *USENIX/ACCURATE Electronic Voting Technology (EVT 2007)*, 2007.

[46] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques, Proceeding*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer, 1995.

[47] Ben Smyth, Steven Frink, and Michael R. Clarkson. Computational Election Verifiability: Definitions and an Analysis of Helios and JCJ. Number 2015/233, 2015. http://eprint.iacr.org/2015/233.

[48] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. Security Analysis of the Estonian Internet Voting System. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 703–715. ACM, 2014.

[49] Alan Szepieniec and Bart Preneel. New Techniques for Electronic Voting. *USENIX Journal of Election Technology and Systems (JETS)*, 3(2):46 – 69, 2015. Cryptology ePrint Archive, Report 2015/809.

[50] Georgios Tsoukalas, Kostas Papadimitriou, Panos Louridas, and Panayiotis Tsanakas. From Helios to Zeus. In *2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13, 2013*. USENIX Association, 2013.

[51] Douglas Wikström. A Sender Verifiable Mix-Net and a New Proof of a Shuffle. In Bimal K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*, pages 273–292. Springer, 2005.

[52] Scott Wolchok, Eric Wustrow, J. Alex Halderman, Hari K. Prasad, Arun Kankipati, Sai Krishna Sakhamuri, Vasavya Yagati, and Rop Gonggrijp. Security analysis of India's electronic voting machines. In *ACM Conference on Computer and Communications Security (CCS 2010)*, pages 1–14, 2010.

[53] Scott Wolchok, Eric Wustrow, Dawn Isabel, and J. Alex Halderman. Attacking the Washington, D.C. Internet Voting System. In *16th Intl. Conference on Financial Cryptography and Data Security (FC'12)*, 2012.

Appendix A
Symbolic Verifiability by Kremer et al.

In this section, we focus on the verifiability definition by Kremer et al. [35] who divide verifiability into three sub-properties.

- *Individual verifiability*: a voter should be able to check that his vote belongs to the ballot box.
- *Universal verifiability*: anyone should be able to check that the result corresponds to the content of the ballot box.
- *Eligibility verifiability*: only eligible voter may vote.

Since the proposed formal definition for eligibility verifiability is rather long and technical, we focus here on individual and universal verifiability.

A. Model

In symbolic models, messages are represented by terms. Kremer et al. model protocols as processes in the applied-pi calculus [5]. A *voting specification* is a pair (V, A) where V is a process that represents the program of a voter while A is an evaluation context that represents the (honest) authorities and the infrastructure. All voters are (implicitly) assumed to be honest.

B. Individual and Universal Verifiability

We can express the definitions of Kremer et al. independently of the execution model, which slightly extends their definitions.

The symbolic verifiability definition by Kremer et al. [35] assumes that each voter V_i performs an individual test φ_i^{IV}, and that observers perform a universal test φ^{UV}. The individual test φ_i^{IV} takes as input the voter's vote and all his local knowledge (e.g. randomness, credentials, and public election data) as well as a partial view of the ballot box (which should correspond to his ballot). The universal test φ^{UV} takes as input the outcome of the election, the public election data, the ballot box, and possibly some extra data generated during the protocol used for the purposes of verification. These tests should satisfy the following conditions for any execution.

Definition 11 (Individual and Universal Verifiability)**.** *A voting specification* (V,A) *satisfies* individual and universal verifiability *if for all* $n \in \mathbb{N}$,

$$\forall i,j\colon \varphi_i^{IV}(\mathsf{b}) \wedge \varphi_j^{IV}(\mathsf{b}) \quad \Rightarrow \quad i=j \tag{1}$$

$$\varphi^{UV}(\mathsf{B},r) \wedge \varphi^{UV}(\mathsf{B},r') \quad \Rightarrow \quad r \approx r' \tag{2}$$

$$\bigwedge_{1\le i\le n} \varphi_i^{IV}(\mathsf{b}_i) \wedge \varphi^{UV}(\mathsf{B},r) \quad \Rightarrow \quad \mathsf{c} \approx r \tag{3}$$

where $\mathsf{c} = (\mathsf{c}_1,\ldots,\mathsf{c}_n)$ *are the choices of the voters,* b *is an arbitrary ballot,* B *is the (content of the) bulletin board,* r *and* r' *are possible outcomes, and* $\approx$ *denotes equality up to permutation.*

Intuitively, Condition (1) ensures that two distinct voters may not agree on the same ballot, i.e., no clash occurs. Condition (2) guarantees the unicity of the outcome: if the observers successfully check the execution, there is at most one outcome they may accept (up to permutation). Finally, Condition (3) is the key property: if all tests succeed, the outcome should correspond to the voters' intent. Observe that, since all voters are assumed to be honest, the implication $\mathsf{c} \approx r$ in Condition (3) can be described by the goal γ_0 (see below).

C. Discussion

Definition (11) is tailored to a specific tally: the outcome of the election has to be the sequence of the votes. Moreover, the definition assumes that the ballot of the voter can be retrieved from the ballot box, which does not apply to ThreeBallot for example. The main restriction is that all voters are assumed to be honest.

Observe that by Condition (3) the goal γ_0 is guaranteed only for protocol runs in which all voters successfully verify their ballots (and the universal test is positive). For the other runs, the outcome can be arbitrary. However, the assumption that all honest voters verify their ballot is unrealistically strong. Therefore, even though this definition uses the strong goal γ_0, this assumption makes the definition weak.

D. Casting in the KTV Framework

Protocol P_{KRS}. The set of agents Σ consists of the voters, the bulletin board B, the judge J, and the remaining participants. Only static corruption is considered. The voters, the bulletin board and the judge do not accept to be corrupted. The honest programs are defined as follows:

– When a voter V_i runs her honest program π_{V_i} and is triggered in order to cast a ballot, she runs the usual program. When V_i is triggered in order to verify her vote, she performs the individual test $\varphi_i^{IV}(\mathsf{b})$ with her ballot b, and if this evaluates to "true", she outputs "accept", otherwise "reject".
– When the judge J runs its honest program π_{J}, it reads the content from the bulletin board B including the result r (if it does not receive any content, it outputs "reject"). Then the judge performs the universal test $\varphi^{UV}(\mathsf{B},r)$, and if this evaluates to "false", the judge outputs "reject". Otherwise, the judge iteratively triggers each voter V_i in order to verify her ballot. If every voter outputs "accept", the judge outputs "accept", and otherwise "false". (This models the requirement in the definition of Kremer et al. that all voters have to verify successfully in order for the run to be accepted. It also means that if not all voters verify, no guarantees are given.)

End-to-end honest verifiability. Let the goal γ_{IUV} be the sub-goal of γ_0 in which all voters produce pairwise different ballots. Then, *individual and universal verifiability* by Kremer et al. (Definition (11)) can essentially be characterized by the fact that the protocol P_{KRS} is $(\gamma_{IUV},0)$-verifiable by the judge J.

To see this, first observe that the judge J as defined above outputs "accept" if and only if the Condition $\bigwedge_{1\le i\le n} \varphi_i^{IV}(\mathsf{b}_i) \wedge \varphi^{UV}(\mathsf{B},r)$ in Condition (3) evaluates to true. As we already pointed out, the implication $\mathsf{c} \approx r$ in Condition (3) describes the goal γ_0. Condition (1) stating that there are no clashes between the ballots of honest voters is also satisfied in γ_{IUV} by definition. Thus, for a protocol which achieves individual and universal verifiability according to Definition 11, the probability that the judge J in P_{KRS} accepts a protocol run in which γ_{IUV} is not fulfilled, is negligible ($\delta = 0$), i.e., we have $\Pr[\pi^{(\ell)} \mapsto \neg\gamma_{IUV}, (\mathsf{J}\colon \mathsf{accept})] \le \delta = 0$ with overwhelming probability as in Definition 1.

Appendix B
Symbolic Verifiability by Cortier et al.

In this section, we study the symbolic verifiability definition by Cortier et al. [18]. Cortier et al. also define different notions of verifiability: individual, universal, and end-to-end verifiability. They prove that under the assumption of an additional property, called "no clash", individual and universal verifiability imply end-to-end verifiability in their symbolic model.

A. Model

As in [35], the definitions of Cortier et al. are cast in a symbolic model. That is, messages are represented by terms and protocols are defined as symbolic processes. Additionally, Cortier et al. assume that voters reach several successive states denoted as follows:

- $\mathsf{Vote}(i,\mathsf{c},\mathsf{cred})$: the voter with identity i owns some credential cred and is willing to cast a choice c.
- $\mathsf{MyBallot}(i,\mathsf{c},\mathsf{b})$: the voter i has prepared a ballot b corresponding to the choice c.
- $\mathsf{VHappy}(i,\mathsf{c},\mathsf{cred},\mathsf{B})$: the voter i with credential cred has cast a choice c and is happy with the content of the ballot box B.

Cortier et al. [18] also assume a judge that checks whether a result r corresponds to a ballot box B and reaches a state $\mathsf{JHappy}(\mathsf{B},r)$ whenever this is the case.

After the casting and before the tallying, some ballots may be removed because they are invalid (e.g., due to flawed signatures or zero-knowledge proofs) or simply because some voters have voted several times and only the last vote counts. This yields a "sanitized" list of ballots $\mathsf{B}_{\mathsf{san}}$.

B. Individual Verifiability

Intuitively, individual verifiability by Cortier et al. holds true if whenever honest voters perform the checks prescribed by the protocol, then their ballots belong to the ballot box.

Definition 12 (Individual Verifiability)**.** *A protocol guarantees* individual verifiability *if for every execution, and for every voter* V_i*, choice* c*, credentials* cred *and ballot box* B*, whenever the state* $\mathsf{VHappy}(i,\mathsf{c},\mathsf{cred},\mathsf{B})$ *is reached, it follows that*

$$\mathsf{Vote}(i,\mathsf{c},\mathsf{cred}) \wedge \exists \mathsf{b} \in \mathsf{B}\colon \mathsf{MyBallot}(i,\mathsf{c},\mathsf{b}).$$

C. Universal Verifiability

The universal verifiability definition by Cortier et al. depends on certain predicates whose purpose is to formally define what it means that a ballot ”contains” a vote and that the tallying proceeds correctly.

Wrap. To define that a vote is “contained” in a ballot, Cortier et al. introduce a predicate $\mathsf{Wrap}(\mathsf{c},\mathsf{b})$ that is left undefined, but has to satisfy the following properties:

(i) Any well-formed ballot b corresponding to some choice c satisfies the Wrap predicate:

$$\mathsf{MyBallot}(i,\mathsf{c},\mathsf{b}) \Rightarrow \mathsf{Wrap}(\mathsf{c},\mathsf{b})$$

(ii) A ballot b cannot wrap two distinct choices c_1 and c_2:

$$\mathsf{Wrap}(\mathsf{c}_1,\mathsf{b}) \wedge \mathsf{Wrap}(\mathsf{c}_2,\mathsf{b}) \Rightarrow \mathsf{c}_1 = \mathsf{c}_2$$

For a given protocol, the definition of Wrap typically follows from the protocol specification.

Good sanitization. When the ballot box B is sanitized, it is acceptable to remove some ballots but of course true honest ballots should not be removed. Therefore, Cortier et al. define the predicate $\mathsf{GoodSan}(\mathsf{B},\mathsf{B}_{\mathsf{san}})$ to hold true (implicitly *relatively to a run*) if the honest ballots of B are not removed from $\mathsf{B}_{\mathsf{san}}$. This means that (i) $\mathsf{B}_{\mathsf{san}} \subseteq \mathsf{B}$, and (ii) for any $\mathsf{b} \in \mathsf{B}$ such that $\mathsf{MyBallot}(i,\mathsf{c},\mathsf{b})$ holds true for some voter V_i and some choice c, it is guaranteed that $\mathsf{b} \in \mathsf{B}_{\mathsf{san}}$.

Good counting. Cortier et al. define a predicate $\mathsf{GoodCount}(\mathsf{B}_{\mathsf{san}},r)$ in order to describe that the final result r corresponds to counting the votes of $\mathsf{B}_{\mathsf{san}}$. This is technically defined in [18] by introducing an auxiliary bulletin board $\mathsf{B}'_{\mathsf{san}}$ which is a permutation of $\mathsf{B}_{\mathsf{san}}$ and from which the list *rlist* of votes (such that $r = \rho(\mathit{rlist})$ where ρ is the counting function) can be extracted line by line from $\mathsf{B}'_{\mathsf{san}}$. More formally, $\mathsf{GoodCount}(\mathsf{B}_{\mathsf{san}},r)$ holds true if there exist $\mathsf{B}'_{\mathsf{san}}, \mathit{rlist}$ such that (i) $\mathsf{B}_{\mathsf{san}}$ and *rlist* have the same size, and (ii) $\mathsf{B}_{\mathsf{san}}$ and $\mathsf{B}'_{\mathsf{san}}$ are equal as multisets, and (iii) $r = \rho(\mathit{rlist})$, and (iv) for all ballots b with $\mathsf{B}'_{\mathsf{san}}[j] = b$ for some index j, there exists a choice c such that $\mathsf{Wrap}(\mathsf{c},b)$ as well as $\mathit{rlist}[j] = \mathsf{c}$ hold true. Note that the definition of $\mathsf{GoodCount}$ is parameterized by the counting function ρ of the protocol under consideration.

Then, universal verifiability is defined as follows.

Definition 13 (Universal Verifiability)**.** *A protocol guarantees* universal verifiability *if for every execution, and every ballot box* B *and result r, whenever the state* $\mathsf{JHappy}(\mathsf{B},r)$ *is reached, it holds that*

$$\exists \mathsf{B}_{\mathsf{san}}\colon \mathsf{GoodSan}(\mathsf{B},\mathsf{B}_{\mathsf{san}}) \wedge \mathsf{GoodCount}(\mathsf{B}_{\mathsf{san}},r).$$

Intuitively, whenever the judge (some election authority) states that some result r corresponds to a ballot box B, then r corresponds to the votes contained in a *subset* $\mathsf{B}_{\mathsf{san}}$ of B (some ballots may have been discarded because they were ill-formed for example) and this subset $\mathsf{B}_{\mathsf{san}}$ contains at least all ballots formed by honest voters that played the entire protocol (that is, including the final checks).

D. E2E Verifiability

Intuitively, end-2-end verifiability according to Cortier et al. holds if, whenever no one complains (including the judge), then the election result includes all the votes corresponding to honest voters that performed the checks prescribed by the protocol.

Definition 14 (E2E Verifiability)**.** *A protocol guarantees* end-2-end verifiability *if for every execution, and every ballot box* B *and result r, whenever a state is reached such that for some subset of the honest voters (indexed by some set I) with choices* c_i *and credentials* cred_i *($i \in I$) we have*

$$\mathsf{JHappy}(\mathsf{B},r) \wedge \bigwedge_{i \in I} \mathsf{VHappy}(i,\mathsf{c}_i,\mathsf{cred}_i,\mathsf{B}),$$

then there exist rlist such that we have $r = \rho(\mathit{rlist})$ *and* $\{\mathsf{c}_i\}_{i\in I} \subseteq$ *rlist (as multisets).*

E. No Clash

Finally, Cortier et al. define the notion of “no clash” as follows. Intuitively, ”no clash” describes the property that two distinct honest voters may not build the same ballot.

Definition 15 (No Clash)**.** *A protocol guarantees* no clash *if for every execution, whenever a state is reached such that* $\mathsf{MyBallot}(i,\mathsf{c}_i,\mathsf{b}) \wedge \mathsf{MyBallot}(j,\mathsf{c}_j,\mathsf{b})$*, then it must be the case that* $i = j$ *and* $\mathsf{c}_i = \mathsf{c}_j$*.*

F. Discussion

Cortier et al. [18] showed that individual verifiability, universal verifiability, and the ”no clash” property together imply End-to-End verifiability (all as defined above).

In order to be able to define their notions of individual and universal verifiability, Cortier et al. proposed a model in which it is possible to (i) extract single ballots from the bulletin board (implicit in the predicate VHappy), and to (ii) uniquely determine the content, i.e. the plain vote, of each single ballot (Wrap predicate). Therefore, these definitions can only be applied to a class of protocols which fulfill these requirements, and by this, for example, ThreeBallot [45] as well as protocols in which ballots are information theoretically secure commitments (e.g. [24]) can not be analyzed.

The notion of end-2-end verifiability (Definition 14) is rather weak since it only requires that honest votes are counted (for voters that checked). It does not control dishonest votes. In particular, this notion does not prevent ballot stuffing. The authors of [18] introduced this notion because the Helios protocol does not satisfy strong verifiability as defined in [19] for example. Moreover, the verification technique based on typing developed in [18] would probably require some adaption to cover strong verifiability as it would need to *count* the number of votes, which is a difficult task for type-checkers.

G. Casting in the KTV Framework

Protocol P_{CEKMW}**.** The set of agents Σ consists of the honest voters, the bulletin board B, the judge J, and the remaining participants. Only static corruption is considered. The bulletin board and the judge do not accept to be corrupted. The honest programs are defined as follows:

– When a voter V runs her honest program π_{V}, and is triggered to cast her ballot, she expects an identity i and a choice c (if not, she stops). Then, she runs $\mathsf{Vote}(\mathsf{c})$ to build her ballot b and to submit it to the bulletin board. Afterwards, she reaches a state $\mathsf{MyBallot}(i,\mathsf{c},\mathsf{b})$. When the voter is triggered to verify her vote, she reads the content of the bulletin board B and reaches a state $\mathsf{VHappy}(i,\mathsf{c},\mathsf{B})$ if her checks evaluate to true.

– When the judge J runs its honest program π_J and is triggered to verify the election run, it reads the content of the bulletin board B including the final result r (if not possible, J outputs "reject"). If the judge successfully performs some checks (which depend on the concrete voting protocol), then he outputs "accept" and reaches a state $\mathsf{JHappy}(\mathsf{B},r)$.

Individual verifiability. We define the goal γ_{IV} to be the set of all runs of P_{CEKMW} in which whenever an honest voter V_i reaches the state $\mathsf{VHappy}(i,\mathsf{c},\mathsf{B})$ for some choice c and ballot b, then there exists a ballot $\mathsf{b}\in\mathsf{B}$ such that this voter started with (i,c) as her input and reached $\mathsf{MyBallot}(i,\mathsf{c},\mathsf{b})$ as intermediary state. Then, *individual verifiability* by Cortier et al. (Definition 12) can essentially be characterized by the fact that the protocol P_{CEKMW} is $(\gamma_{IV},0)$-verifiable by the judge J.

Universal verifiability. We define the goal γ_{UV} to be the set of all runs of P_{CEKMW} in which whenever a result r is obtained and the final content of the ballot box is B then there exists B_san such that $\mathsf{GoodSan}(\mathsf{B},\mathsf{B}_\mathsf{san})$ and $\mathsf{GoodCount}(\mathsf{B}_\mathsf{san},r)$ hold true (as defined above). Then, *universal verifiability* by Cortier et al. (Definition 13) can essentially be characterized by the fact that the protocol P_{CEKMW} is $(\gamma_{UV},0)$-verifiable by the judge J.

End-to-end verifiability. We define the goal γ_{E2E} to be the set of all runs of P_{CEKMW} in which the result r of the election satisfies $r=\rho(rlist)$ for some $rlist$ that contains (as multiset) all the choices c_i for which some honest voter V_i reached a state $\mathsf{VHappy}(i,\mathsf{c}_i,\mathsf{cred}_i\mathsf{B})$. Then, *end-to-end verifiability* by Cortier et al. (Definition 14) can essentially be characterized by the fact that the protocol P_{CEKMW} is $(\gamma_{E2E},0)$-verifiable by the judge J.

Appendix C
Definition of Smyth et al.: Casting in the KTV Framework

We cast the definitions of individual verifiability, universal verifiability and election verifiability by Smyth et al. [47] in the framework of Definition 1.

Protocol P_{SFC}. The set of agents Σ consists of the voters, the bulletin board B, the judge J, the scheduler, and the remaining participants. Since static corruption is considered, the agents only accept the corrupt message at the beginning of an election run. The bulletin board and the judge do not accept to be corrupted.

When a voter V runs her honest program π_V, she expects a candidate c as input (if the input is empty, she stops). After that, she reads the public election parameters $\mathsf{prm}_\mathsf{pub}$ from the bulletin board B (if she does not receive any election paramaters on B, she stops). Then, she runs $\mathsf{Vote}(\mathsf{c},\mathsf{prm}_\mathsf{pub})$ and sends the resulting ballot b to the bulletin board B. Although this is kept implicit in the discussed paper, we will assume here that V subsequently checks that its ballot is published on B.

When the judge J runs its honest program π_J, it reads the content from the bulletin board B, including the public paramaters $\mathsf{prm}_\mathsf{pub}$, the tally Tally, and the proof P (if the judge does not receive one of these inputs, it outputs "reject"). Then, the judge runs Verify and outputs "accept" or "reject", respectively, if $\mathsf{Verify}(\mathsf{B},\mathsf{prm}_\mathsf{pub},\mathsf{Tally},\mathsf{P})$ evaluates to "true" or "false".

Individual verifiability. We define the goal γ_{IV} to be the set of all runs of P_{SFC} in which all honest voters' ballots are pairwise different (if $\neq\bot$), i.e., no clashes occur. For the protocol P_{SFC}, *individual verifiability* according to Smyth et al. can essentially be characterized by the fact that the protocol P_{SFC} is $(\gamma_{IV},0)$-verifiable by the judge J in the sense of Definition 1.

To see this, observe that, if a protocol achieves individual verifiability according to Definition 7, then for all ppt adversaries π_A the probability $\mathsf{Pr}[\pi(1^\ell)\mapsto\neg\gamma_{IV},\,(\mathsf{J}\colon\mathsf{accept})]\leq\mathsf{Pr}[\pi(1^\ell)\mapsto\neg\gamma_{IV}]$ is negligible for $\pi=\pi_P\parallel\pi_\mathsf{A}$, where the latter probability is negligible, if the protocol satisfies Definition 7.

For the implication in the opposite direction, let us assume that $\mathsf{Pr}[\pi(1^\ell)\mapsto\neg\gamma_{IV},\,(\mathsf{J}\colon\mathsf{accept})]$ is negligible for all adversaries. Now, for each adversary A from the game used in Definition 7, there is a corresponding adversary π_A which always produces correct tally (note that A is not concerned with tallying). For this adversary we have $\mathsf{Pr}[\pi(1^\ell)\mapsto\neg\gamma_{IV},\,(\mathsf{J}\colon\mathsf{accept})]=\mathsf{Pr}[\pi(1^\ell)\mapsto\neg\gamma_{IV}]$ which, by the above assumption, is negligible. This implies individual verifiability (in the sense of Definition 7).

Universal verifiability. We define the goal γ_{UV} to be the set of all runs of P_{SFC} in which first $\mathsf{prm}_\mathsf{pub}$ and then a final result $(\mathsf{Tally},\mathsf{P})$ are published and $\mathsf{Tally}=\mathsf{correct\ tally}(\mathsf{B},\mathsf{prm}_\mathsf{pub})$ (recall that B is the content of the bulletin board that contains voters' ballots).

For the protocol P_{SFC}, *universal verifiability* according to Smyth et al. can essentially be characterized by the fact that the protocol P_{SFC} is $(\gamma_{UV},0)$-verifiable in the sense of Definition 1.

To see this, first observe that, for each adversary A, the condition $\mathsf{Verify}(\mathsf{B},\mathsf{prm}_\mathsf{pub},\mathsf{Tally}',\mathsf{P}')$ in Experiment $\mathsf{ExpUV}(\Pi,\mathsf{A})$ (Fig. 6) is true if an honest judge J outputs "accept" (in the system π with the corresponding adversary), and false otherwise. Second, the adversary A in Experiment $\mathsf{ExpUV}(\Pi,\mathsf{A})$ produces a tuple $(\mathsf{B},\mathsf{prm}_\mathsf{pub},\mathsf{Tally}',\mathsf{P}')$ for which $\mathsf{Tally}'\neq\mathsf{correct\ tally}(\mathsf{B},\mathsf{prm}_\mathsf{pub})$ holds true if and only if we have $\neg\gamma_{UV}$ (in the corresponding run of π).

Thus, essentially, for a voting protocol P achieving universal verifiability according to Definition 9 (which means that the success rate in Experiment $\mathsf{ExpUV}(\Pi,\mathsf{A})$ (Fig. 6) is negligible for every ppt adversary A) is equivalent to the statement that the goal γ_{UV} is 0-verifiable by the judge J according to Definition 1 (which means that the probability $\mathsf{Pr}[\pi(1^\ell)\mapsto\neg\gamma_{UV},\,(\mathsf{J}\colon\mathsf{accept})]$ is negligible in every instance $\pi_P\parallel\pi_\mathsf{A}$).

Election verifiability. According to Smyth et al. the protocol P_{SFC} achieves *election verifiability* if it achieves individual and universal verifiability. Therefore this notion can be expressed in the language of Definition 1 using the goal $\gamma_{IV}\wedge\gamma_{UV}$.