Accountability: Definition and Relationship to Verifiability

Ralf Küsters
University of Trier, Germany
kuesters@uni-trier.de

Tomasz Truderung
University of Trier, Germany
truderun@uni-trier.de

Andreas Vogt
University of Trier, Germany
vogt@uni-trier.de

ABSTRACT

Many cryptographic tasks and protocols, such as non-repudiation, contract-signing, voting, auction, identity-based encryption, and certain forms of secure multi-party computation, involve the use of (semi-)trusted parties, such as notaries and authorities. It is crucial that such parties can be held accountable in case they misbehave as this is a strong incentive for such parties to follow the protocol. Unfortunately, there does not exist a general and convincing definition of accountability that would allow to assess the level of accountability a protocol provides.

In this paper, we therefore propose a new, widely applicable definition of accountability, with interpretations both in symbolic and computational models. Our definition reveals that accountability is closely related to verifiability, for which we also propose a new definition. We prove that verifiability can be interpreted as a restricted form of accountability. Our findings on verifiability are of independent interest.

As a proof of concept, we apply our definitions to the analysis of protocols for three different tasks: contract-signing, voting, and auctions. Our analysis unveils some subtleties and unexpected weaknesses, showing in one case that the protocol is unusable in practice. However, for this protocol we propose a fix to establish a reasonable level of accountability.

1. INTRODUCTION

Many cryptographic tasks and protocols, such as non-repudiation [48], contract-signing [4], voting [16, 10], auctions [38], identity-based encryption [19, 20], and certain forms of secure multi-party computation [24], involve the use of (semi-)trusted parties, such as notaries and authorities. It is crucial that such parties can be held accountable in case they misbehave as this is a strong, in some cases maybe the main incentive for such parties to follow the protocol. Unfortunately, there does not exist a general and convincing definition of accountability that would allow to assess the level of accountability a protocol provides. The few existing formulations of accountability are, for the most part, quite ad hoc and protocol specific (see Section 4 for the related work).

The main goal of this paper is therefore to propose a new, general definition of accountability and to demonstrate its applicability to a wide range of cryptographic tasks and protocols. Jumping ahead, it turns out that accountability is closely related to verifiability. This motivated us to also propose a new definition for this prominent security requirement. More precisely, our contributions are as follows.

This work was partially supported by *Deutsche Forschungsgemeinschaft* (DFG) under Grant KU 1434/5-1. An abridged version was published in [34].

Contribution of this Paper. In this paper, we propose a general, model-independent definition of accountability. We provide interpretations of our definition both in symbolic (Dolev-Yao style) and computational (cryptographic) models. While, as usual, analysis in the symbolic model is simpler and more amenable to tool-support, the computational definition gives stronger security guarantees, as it does not abstract from cryptographic details and allows for a more-fine grained measure of accountability. As for the symbolic definition, we discuss and illustrate how existing analysis tools can be used to check accountability in some cases.

Our definition of accountability is applicable to a wide range of cryptographic tasks and protocols, yet it allows to precisely capture the level of accountability a protocol provides. This is demonstrated in three case studies, in which we apply our definition to protocols for three important cryptographic tasks: contract-signing, voting, and auctions. Our analysis of these protocols reveals some subtleties and unexpected, partly severe weaknesses. For example, in the auction protocol that we analyze [38], which was explicitly designed to be of practical use, our analysis shows that if two bidders with two different bids claim to be the winner of the auction, then, even if it is clear that one of the two bidders misbehaved, a judge cannot blame a specific bidder. It even remains open whether the auctioneer was honest and who actually won the auction. We propose a fix for this problem and prove that it in fact solves the problem.

As mentioned, it turns out that accountability is closely related to verifiability. Therefore, we also introduce a new definition of verifiability, again with a symbolic and computational interpretation. This definition is interesting in its own right: It is again applicable to a wide range of cryptographic tasks and protocols. Also, unlike other definitions and informal descriptions, our definition takes a global view on verifiability, centered around the overall goal of a protocol, rather than focussing on what, in the context of e-voting, is called individual and universal verifiability; although these forms of verifiability can also be captured by our definition (see Sections 3 and 4).

We show that verifiability can be interpreted as a restricted form of accountability. While, given our definitions, this relationship is easy to see, in the literature, accountability and verifiability have not been formally connected before. The relationship offers a deeper understanding of the two notions and allows to derive statements for verifiability from statements for accountability, as illustrated by our case studies. We believe that accountability is the property protocol designers should aim for, not just verifiability, which on its own is often too weak a property in practice: If a protocol participant (rightly) complains that something went wrong, then it should be possible to (rightly) hold specific protocol participants accountable for their misbehavior, and by this, resolve the dispute.

Structure of the Paper. Accountability is defined in Section 2. In Section 3 we provide our definition of verifiability, along with the proposition that shows that verifiability is implied by accountability. Related work is discussed in Section 4. Our case studies are presented in Sections 5 (voting), 6 (auction), and 7 (contract signing). More details can be found in the appendix.

2. ACCOUNTABILITY

In this section, we provide our definition of accountability. As mentioned in the introduction, we present two variants: a symbolic and a computational one, which conceptually are closely related. We start with a definition of protocols.

2.1 Protocols

In this section, we present a generic definition of a protocol, suitable for the definition of accountability (and verifiability).

We do not fix any specific symbolic or computational model as our definitions do not depend on details of such models. We only require that the model provides us with a notion of a *process* which can perform internal computation and can communicate with other processes by sending messages via *(external) input/output* channels. We also assume that processes can be composed to form new processes; however, the composition may be subject to certain conditions. If π and π' are processes, then we write $\pi \parallel \pi'$ for the composition of π and π'. Moreover, in the symbolic setting, we assume that a process defines a set of runs; we assume a set of runs, rather than a single run, as processes may be nondeterministic. In the computational setting, a process defines a family of probability distributions over runs, indexed by the security parameter. The representation of a single run should contain a description of the corresponding process. In the computational setting, a single run also contains the security parameter and all random coins. We will consider only *complete* runs that cannot be extended, which in the symbolic setting can include infinite runs. Possible symbolic instances of our framework include the applied π-calculus [2] and models based on I/O-automata, see, e.g., [28]. In a computational model, processes would typically be modeled as probabilistic polynomial-time systems of probabilistic polynomial-time interactive Turing machines (ppt ITMs), see, e.g., [18]. Our case studies provide concrete examples (see Sections 5 to 7).

For sets I and O of channel names, we denote by $\Pi(I,O)$ the set of all processes with external input channels in I and external output channels in O.

DEFINITION 1 (**PROTOCOL**). A *protocol* is a tuple $P = (\Sigma, \mathsf{Ch}, \mathrm{In}, \mathrm{Out}, \{\Pi_a\}_{a\in\Sigma}, \{\hat{\Pi}_a\}_{a\in\Sigma})$, where:

- $\Sigma = \{\mathsf{a}_1, \dots, \mathsf{a}_n\}$ and Ch are finite sets, called the set of *agents* and *channels* of P, respectively.
- In and Out are functions from Σ to 2^{Ch} such that $\mathrm{Out}(a)$ and $\mathrm{Out}(b)$ are disjoint for all $a \neq b$ and $\mathrm{In}(a)$ and $\mathrm{In}(b)$ are disjoint for all $a \neq b$. The sets $\mathrm{In}(a)$ and $\mathrm{Out}(a)$ are called the set of (external) *input* and *output channels of agent a*, respectively. We assume that a special channel $\mathsf{decision}_a \in \mathsf{Ch}$ is an element of $\mathrm{Out}(a)$, for every $a \in \Sigma$, but that it is not an input channel for any agent.
- $\Pi_a \subseteq \Pi(\mathrm{In}(a), \mathrm{Out}(a))$, for every $a \in \Sigma$, is called the set of *programs of a*. This set contains all programs a can possibly run, modeling both honest and potential dishonest behavior.
- $\hat{\Pi}_a \subseteq \Pi_a$, for every $a \in \Sigma$, is called the set of *honest programs of a*, i.e., the set of programs that a runs if a is honest. Often this set is a singleton, but sometimes it is convenient to consider non-singleton sets.

Let $P = (\Sigma, \mathsf{Ch}, \mathrm{In}, \mathrm{Out}, \{\Pi_a\}_{a\in\Sigma}, \{\hat{\Pi}_a\}_{a\in\Sigma})$ be a protocol. An *instance of P* is a process of the form $\pi = (\pi_{\mathsf{a}_1} \parallel \dots \parallel \pi_{\mathsf{a}_n})$ with $\pi_{\mathsf{a}_i} \in \Pi_{a_i}$. We say that a_i is *honest* in such an instance, if $\pi_{\mathsf{a}_i} \in \hat{\Pi}_{\mathsf{a}_i}$. A *run of P* is a run of some instance of P. We say that a_i is honest in a run r, if r is a run of an instance of P with honest a_i. A *property* γ *of P* is a subset of the set of all runs of P. By $\neg\gamma$ we denote the complement of γ.

2.2 Symbolic and Computational Accountability

We now provide a symbolic and a computational definition of accountability.

Our definition of accountability is w.r.t. an agent J of the protocol who is supposed to blame protocol participants in case of misbehavior. The agent J, which we sometimes refer to as a *judge*, can be a "regular" protocol participant or an (external) judge, possibly provided with additional information by other protocol participants; however, J may not necessarily trust these other protocol participants since they may be dishonest and may provide J with bogus information.

In order to understand the subtleness of accountability, it is instructive to look at a first (flawed) definition of accountability and its possible interpretations, inspired by informal statements about accountability in the literature.

(i) (*fairness*) J (almost) never blames protocol participants who are honest, i.e., run their honest program.

(ii) (*completeness*) If in a protocol run participants "misbehave", then J blames those participants.

While the fairness condition is convincing and clear, this is not the case for the completeness condition. First, the question is what "misbehavior" means. It could be interpreted as a behavior that does not correspond to any honest behavior. However, this interpretation is much too strong. No protocol would satisfy it, because this includes misbehavior that is impossible to be observed by any other party and misbehavior that is completely "harmless" and "irrelevant". For example, if, in addition to the messages a party A is supposed to send to another party B, A also sends some harmless message "hello", say, then B can observe this misbehavior, but cannot convince J of any misbehavior. This example also shows that interpreting "misbehavior" as dishonest behavior observable by honest parties, and hence, misbehavior that, at least to some extent, affects these parties, does not work either. In fact, a completeness condition based on this notion of "misbehavior" would again deem basically all non-trivial protocols insecure w.r.t. accountability. More importantly, this completeness condition misses the main point: Misbehavior that cannot be observed by any honest party may still be very relevant and harmful. We therefore advocate an interpretation that circles around the desired goals of a protocol.

Informally speaking, our definition of accountability reads as follows:

(i) (*fairness*) J (almost) never blames protocol participants who are honest, i.e., run their honest program.

(ii) (*completeness*, goal centered) If, in a run, some desired goal of the protocol is not met—due to the misbehavior of one or more protocol participants—then J blames those participants who misbehaved, or at least some of them (see below).

For example, for voting protocols a desired goal could be that the published result of the election corresponds to the actual votes cast by the voters. The completeness condition now guarantees that if in a run of the protocol this is not the case (a fact that must be due to the misbehavior of one or more protocol participants), then one or more participants are held accountable by J; by the fairness condition they are *rightly* held accountable. In case of auctions, a desired goal could be that the announced winner is in fact the winner of the auction; if this is not so in a run, by the completeness condition some participant(s), who misbehaved, will be blamed. Desired goals, as the above, will be a parameter of our definition.

The informal completeness condition above leaves open who exactly should be blamed. This could be fixed in a specific way. However, this would merely provide a black and white picture, and either set the bar too high or too low for many protocols. For example, it is desirable that the judge, whenever a desired goal of a protocol is not met, blames *all* misbehaving parties. This, as explained above, is usually not possible (e.g., if for a dishonest party the deviation from the protocol consists in sending a harmless "hello" message). So, this sets the bar too high for practically every protocol. Alternatively, one could require that at least some misbehaving parties can be blamed individually (*individual accountability*). Being able to rightly blame individual parties, rather than, say, just a group of parties among which at least one misbehaved, is important in practice, since only this might have actual consequences for a misbehaving party. However, as illustrated by our case studies, protocols often fail to achieve individual accountability. One could set the bar lower and only require that a group of parties is blamed among which at least one misbehaved. But this is often unsatisfying in practice. Altogether, rather than fixing the level of accountability protocols are supposed to provide up front, it is more reasonable to have a language in which this can be described precisely, allowing to compare protocols and tell apart weak protocols from strong ones.

To this end, below we introduce what we call accountability properties, which are sets of what we call accountability constraints. We also allow the judge to state quite detailed "verdicts".

Formally, a *verdict* is a positive boolean formula ψ built from propositions of the form $\mathsf{dis}(a)$, for an agent a, where $\mathsf{dis}(a)$ is intended to express that a misbehaved (behaved dishonestly), i.e., did not follow the prescribed protocol. Let us look at some examples. If the judge states $\mathsf{dis}(a) \vee \mathsf{dis}(b)$, then this expresses the judge's belief that a or b misbehaved. (In case of a fair judge, this implies that at least one of the two parties indeed misbehaved.) Another example: In a voting protocol, with a voting machine M and auditors $A_1,\dots,A_r$, if the judge states, say, $\mathsf{dis}(M) \wedge \mathsf{dis}(A_1) \wedge \dots \wedge \mathsf{dis}(A_r)$, then this expresses the judge's belief that the voting machine and all auditors misbehaved; the judge would state $\mathsf{dis}(M) \vee (\mathsf{dis}(A_1) \wedge \dots \wedge \mathsf{dis}(A_r))$ if she is not sure whether the voting machine or all auditors misbehaved. Our case studies demonstrate the usefulness of such expressive forms of verdicts. We will denote by $\mathscr{F}_{\mathsf{dis}}$ the set of all verdicts. A party J can *state a verdict* ψ, by sending ψ on its dedicated output channel $\mathsf{decision}_J$. Note that, in one run, J may state many different verdicts $\psi_1,\dots,\psi_k$, which is equivalent to stating the verdict $\psi_1 \wedge \dots \wedge \psi_k$.

Formally, for a protocol P and an instance π of P, a verdict ψ *is true in* π, written $\pi \models \psi$, iff the formula ψ evaluates to true with the proposition $\mathsf{dis}(a)$ set to false, if a is honest in π, and set to true otherwise.

We now introduce accountability constraints and accountability properties which allow to precisely describe the level of accountability a protocol provides.

An *accountability constraint* of a protocol P is a tuple $(\alpha,\psi_1,\dots,\psi_k)$, written $(\alpha \Rightarrow \psi_1 \mid \dots \mid \psi_k)$, where α is a property of P and $\psi_1,\dots,\psi_k \in \mathscr{F}_{\mathsf{dis}}$. We say that a constraint $(\alpha \Rightarrow \varphi_1 \mid \dots \mid \varphi_k)$ *covers* a run r, if $r \in \alpha$.

Intuitively, in a constraint $C = (\alpha \Rightarrow \psi_1 \mid \dots \mid \psi_k)$, the set α contains runs in which some desired goal of the protocol is not met (due to the misbehavior of some protocol participant). The formulas $\psi_1,\dots,\psi_k$ are the possible (minimal) verdicts that are supposed to be stated by J in such a case; J is free to state stronger verdicts (by the fairness condition these verdicts will be true). Formally, for a run r, we say that *J ensures C in r*, if either $r \notin \alpha$ or J states in r a verdict ψ that implies one of $\psi_1,\dots,\psi_k$ (in the sense of propositional logic).

EXAMPLE 1. To illustrate the notion of accountability constraints, let us consider the following examples, where, say, J is supposed to blame misbehaving parties, M is a voting machine, $A_1,\dots,A_r$ are auditors, and α contains all runs in which the published result of the election is incorrect:

$$C_1^{ex} = \alpha \Rightarrow \mathsf{dis}(M) \mid \mathsf{dis}(A_1) \mid \dots \mid \mathsf{dis}(A_r) \tag{1}$$
$$C_2^{ex} = \alpha \Rightarrow \mathsf{dis}(M) \vee (\mathsf{dis}(A_1) \wedge \dots \wedge \mathsf{dis}(A_r)) \tag{2}$$
$$C_3^{ex} = \alpha \Rightarrow \mathsf{dis}(M) \mid \mathsf{dis}(A_1) \wedge \dots \wedge \mathsf{dis}(A_r). \tag{3}$$

Constraint C_1^{ex} requires that if in a run the published result of the election is incorrect, then at least one (individual) party among M, $A_1,\dots,A_r$ can be held accountable by J; note that different parties can be blamed in different runs. Party J ensures C_1^{ex} in a run $r \in \alpha$, if, for example, J states $\mathsf{dis}(A_1)$ or J states $\mathsf{dis}(M) \wedge \mathsf{dis}(A_r)$, but not if J only states $\mathsf{dis}(M) \vee \mathsf{dis}(A_1)$. Constraint C_3^{ex} is stronger than C_1^{ex} as it requires that it is possible to hold $\mathsf{dis}(M)$ or *all* auditors accountable. In this case, for J it does not suffice to state $\mathsf{dis}(A_1)$, but stating $\mathsf{dis}(M) \wedge \mathsf{dis}(A_r)$ or $\mathsf{dis}(A_1) \wedge \dots \wedge \mathsf{dis}(A_r)$ does. Constraint C_2^{ex} is weaker than C_3^{ex}, and incomparable to C_1^{ex}. It states that if the published result of the election is incorrect, then J can leave it open whether $\mathsf{dis}(M)$ or all auditors misbehaved.

As mentioned before, we think that in practice, individual accountability is highly desirable to deter parties from misbehaving. So ideally, protocols should satisfy accountability constraints where in case a desired goal is not met, at least one misbehaving party is blamed individually. Formally, we say that $(\alpha \Rightarrow \psi_1 \mid \dots \mid \psi_k)$ provides *individual accountability*, if for every $i \in \{1,\dots,k\}$, there exists a party a such that ψ_k implies $\mathsf{dis}(a)$. In other words, each $\psi_1,\dots,\psi_k$ determines at least one misbehaving party. In Example 1, C_1^{ex} and C_3^{ex} provide individual accountability, but C_2^{ex} does not.

A set Φ of constraints for protocol P is called an *accountability property* of P. Typically, an accountability property Φ covers all relevant cases in which desired goals for P are not met, i.e., whenever some desired goal of P is not satisfied in a given run r due to some misbehavior of some protocol participant, then there exists a constraint in Φ which covers r. We note that considering sets of accountability constraints rather than just a single constraint provides more expressiveness: A set of constraints allows to more precisely link the participants to be blamed with specific violations, and hence, captures more precisely the kind of accountability provided by a protocol (see our case studies for examples.

We are now ready to provide precise symbolic and computational definitions of accountability. As already mentioned, conceptually these two definitions share the same basic idea outlined above.

Symbolic Accountability. Let P be a protocol and J be an agent of P. We say that J is *fair*, if his/her verdicts are never false. Formally, J is *fair in P*, if, for every instance π of P and every run r of π, whenever J states a verdict ψ in r, then $\pi \models \psi$. For instance, if in some run with honest M and A_1, an agent J states $\mathsf{dis}(M) \vee \mathsf{dis}(A_1)$, then J is not fair.

DEFINITION 2 (**Symbolic accountability**). Let P be a protocol with the set of agents Σ, let $J \in \Sigma$, and Φ be an accountability property of P. We say that *J ensures Φ-accountability for protocol P* (or *P is Φ-accountable w.r.t. J*) if

(i) (*fairness*) J is fair in P and

(ii) (*completeness*) for every constraint C in Φ and every run r of P, J ensures C in r.

While the completeness condition requires J's verdicts to be sufficiently strict, i.e., at least as strict as the constraints require, fairness guarantees that J's verdicts are correct. Note that the fairness condition does not depend on the accountability property under consideration.

REMARK 1 (AUTOMATIC ANALYSIS). The fairness condition can often be checked automatically by tools for cryptographic protocol analysis since it is a reachability property: For all $B \subseteq \Sigma$, one considers systems in which the agents in B run their honest programs. Then, one checks whether a state can be reached, where J states ψ such that ψ does not evaluate to true if $\mathsf{dis}(b)$ is set to false iff $b \in B$. This can often be done automatically, provided that the cryptographic primitives used and the communication model the protocol builds on can be handled by the analysis tool and provided that the sets $\hat{\Pi}_c$ and Π_c of programs of agents c, as specified in the protocol P, are either finite or as powerful as a Dolev-Yao intruder.

Whether or not the completeness condition can be checked automatically heavily depends on the accountability property under consideration.

Our analysis of the contract-signing protocol considered in Section 7 illustrates how the fairness condition can be checked automatically; in this case, the completeness condition can also be checked automatically, but it is quite trivial.

Computational Accountability 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 $\ell > \ell_0$.

Let P be a protocol with the set Σ of agents. Since we now consider the computational setting, we assume that the programs agents run are ppt ITMs. Let Φ be an accountability property of P. Let π be an instance of P and $J \in \Sigma$ be an agent of P. For a set V of verdicts, we write $\mathsf{Pr}[\pi(1^\ell) \mapsto \{(J:\psi) \mid \psi \in V\}]$ for the probability that π produces a run in which J states ψ for some $\psi \in V$, where the probability is taken over the random coins of the ITMs in π and 1^ℓ is the security parameter given to the ITMs. Similarly, we write $\mathsf{Pr}[\pi(1^\ell) \mapsto \neg(J:\Phi)]$ to denote the probability that π, with security parameter 1^ℓ, produces a run such that J does not ensure C in this run, for some $C \in \Phi$.

An agent J is computationally fair, if he states false verdicts only with negligible probability. Formally, J is *computationally fair* in a protocol P, if $\mathsf{Pr}[\pi(1^\ell) \mapsto \{(J:\psi) \mid \pi \not\models \psi\}]$ is negligible as a function of ℓ, for all instances π of P.

DEFINITION 3 (**Computational accountability**). Let P be a protocol with the set of agents Σ, $J \in \Sigma$, Φ be an accountability property of P, and $\delta \in [0,1]$. We say that *J ensures (Φ,δ)-accountability for protocol P* (or *P is (Φ,δ)-accountable w.r.t. J*) if

(i) (*fairness*) J is computationally fair in P and

(ii) (*completeness*) for every instance π of P, the probability $\mathsf{Pr}\left[\pi(1^\ell) \mapsto \neg(J:\Phi)\right]$ is δ-bounded as a function of ℓ.

In the completeness condition, it is of course desirable that $\delta = 0$, i.e., the probably that J fails to ensure a constraint is negligible. However, as we will illustrate in Section 5, this is often too demanding. Instead of giving up in such cases, by introducing the parameter δ, we can measure the level of completeness a protocol provides.

3. VERIFIABILITY

In this section, we provide a symbolic and a computational definition of verifiability and show that verifiability is a restricted form of accountability. We use the terminology and notation introduced in Section 2.

Symbolic and Computational Verifiability. Let P be a protocol and γ be a property of P, called the *goal of P*. We say that an agent J *accepts a run* r, if in this run J sends the message accept on channel $\mathsf{decision}_J$. Intuitively, J accepts a run if she believes that the goal has been achieved in this run.

The agent J may be a regular protocol participant (voter, bidder, authority, etc.) or an external judge, who is provided with information by (possibly untrusted) protocol participants.

Expressing goals as properties of a protocol is, as in case of accountability, a powerful and flexible tool, which for voting protocols, for example, allows to capture several forms of verifiability considered in the literature: The goal of an agent (a voter, in this case) J could, for example, include all runs in which her vote is counted as cast; this goal aims at what is called *individual verifiability* [42]. Another goal could include all runs in which the ballots shown on a bulletin board are counted correctly; this goal aims at what is called *universal verifiability* [42]. In [44], another type of verifiability is considered, namely *eligibility verifiability*. This is captured by the goal γ that includes those runs where only eligible voters vote at most once. However, the bottom line should be a goal, which we call *global verifiability*, that contains all runs in which the published result exactly corresponds to the votes cast by eligible voters (see Section 5 for a more precise formulation and a more in depth discussion). This goal has not formally been considered in the literature so far, at most implicitly as a conjunction of all the above mentioned goals. Analogously, goals for other kinds of protocols, such as auction protocols, can be formulated (see Section 6).

In our definition of verifiability, we require that an agent J accepts a run, only if the goal of the protocol is satisfied. This requirement, however, would be easily satisfied in *every* protocol by an agent who never accepts a run. Therefore, the definition of verifiability should also contain conditions under which the goal should be achieved and runs should be accepted. Clearly, one may expect that a protocol run should be accepted (and the goal should be achieved), at least when all the protocol participants are honest. Furthermore, in some protocols, such as those for e-voting, one may expect that to achieve the goal it is sufficient that voting authorities follow the protocol, regardless of whether or not the voters behave honestly. Therefore, our definition, besides the goal, has an additional parameter: a positive boolean formula over propositions of the form $\mathsf{hon}(a)$, for an agent a, which describes a group or groups of participants that can guarantee, when running their honest programs, that a goal of a protocol is achieved. We will denote the set of such formulas by $\mathscr{F}_{\mathsf{hon}}$. For example, for an e-voting protocol with a voting machine M and auditors $A_1,\dots,A_r$, one might expect that to achieve the goal of the protocol it is sufficient that M is honest and at least one of the auditors $A_1,\dots,A_r$ is honest. This can be expressed by the formula $\varphi_{ex} = \mathsf{hon}(M) \wedge (\mathsf{hon}(A_1) \vee \cdots \vee \mathsf{hon}(A_r))$.

For an instance π of P and $\psi \in \mathscr{F}_{\mathsf{hon}}$, we write $\pi \models \psi$ if ψ evaluates to true with the proposition $\mathsf{hon}(a)$ set to true, if a is honest in π, and set to false otherwise.

We can now provide symbolic and computational definitions of verifiability.

DEFINITION 4 (**Symbolic verifiability**). Let P be a protocol with the set of agents Σ. Let $J \in \Sigma$, $\psi \in \mathscr{F}_{\mathsf{hon}}$, and γ be a property of P. Then, we say that *the goal γ is guaranteed in P by ψ and verifiable by J* if the following conditions are satisfied:

(i) For every run r of an instance π of P such that $\pi \models \psi$, the agent J accepts r.

(ii) For every run r of an instance of P in which J accepts r, it holds that $r \in \gamma$.

Condition (ii) guarantees that J only accepts a run if the goal is in fact achieved. Condition (i) says that the protocol is sound in the sense that if ψ holds, i.e. certain participants are honest, as described by ψ, then indeed J accepts, which by Condition (ii) implies that the goal is achieved.

This definition can easily be turned into a computational definition of verifiability. For this, by $\mathsf{Pr}[\pi(1^\ell) \mapsto (J : \mathsf{accept})]$ we denote the probability that π, with security parameter 1^ℓ, produces a run which is accepted by J. Analogously, by $\mathsf{Pr}[\pi(1^\ell) \mapsto \neg\gamma, (J : \mathsf{accept})]$ 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 5 (**Computational verifiability**). *Let P be a protocol with the set of agents Σ. Let $\delta \in [0,1]$, $J \in \Sigma$, $\psi \in \mathscr{F}_{\mathsf{hon}}$, and γ be a property of P. Then, we say that* the goal γ is guaranteed in P by ψ and δ-verifiable by J *if for every instance π of P the following conditions are satisfied:*

(i) If $\pi \models \psi$, then $\mathsf{Pr}[\pi(1^\ell) \mapsto (J : \mathsf{accept})]$ is overwhelming as a function of ℓ.

(ii) $\mathsf{Pr}[\pi(1^\ell) \mapsto \neg\gamma, (J : \mathsf{accept})]$ is δ-bounded as a function of ℓ.

Just as in case of accountability, assuming negligibility in Condition (ii), i.e., $\delta = 0$, is too strong for many reasonable protocols.

Relationship to Accountability. The following proposition shows that verifiability can be considered to be a special case of accountability. While, given our definitions, this relationship is easy to prove, in the literature, accountability and verifiability have not been formally connected before.

Let $\varphi \in \mathscr{F}_{\mathsf{hon}}$. We denote by $\overline{\varphi} \in \mathscr{F}_{\mathsf{dis}}$ the negation normal form of φ, where $\neg\mathsf{hon}(b)$ is replaced by $\mathsf{dis}(b)$. For example, for φ_{ex} as above, we have $\overline{\varphi_{ex}} = \mathsf{dis}(M) \vee (\mathsf{dis}(A_1) \wedge \cdots \wedge \mathsf{dis}(A_r))$.

Let P be a protocol and J be an agent such that J states only formulas ψ that imply $\overline{\varphi}$. Furthermore, assume that J accepts a run iff it does not output a formula ψ. Now, the proposition is as follows (see Appendix C for the proof):

PROPOSITION 1. *Let φ, P and J be defined as above. Let γ be a property of P. Then the statement*

$$J \text{ ensures } \{\neg\gamma \Rightarrow \overline{\varphi}\}\text{-accountability for } P \tag{4}$$

implies the statement

$$\gamma \text{ is guaranteed by } \varphi \text{ in } P \text{ and verifiable by } J. \tag{5}$$

If we additionally assume that, in P, J blames only $\overline{\varphi}$ (i.e. if J outputs ψ, then $\psi = \overline{\varphi}$), then we also have that (5) *implies* (4).

This holds for both the symbolic and the computational definitions, where in the latter case the same $\delta \in [0,1]$ can be used for accountability and verifiability.

So, verifiability is implied by a restricted form of accountability. As our case studies show (see Sections 5 and 6), $\overline{\varphi}$ typically does not provide individual accountability, and hence, verifiability is merely a weak form of accountability, and as argued before, often too weak in practice, since in case something goes wrong, it is not possible to held individual parties accountable.

4. RELATED WORK

As already mentioned in the introduction, accountability and verifiability play a crucial role for many cryptographic tasks and protocols. However, in most works, accountability and verifiability or related notions are merely described informally or are tailored to specific protocols and security aspects (see, e.g., [4, 5, 17, 47, 45, 14, 13, 41, 3, 11, 40, 37, 10, 12]).

The only work which tried to deal with the general notion of accountability (and which illustrates that coming up with a convincing definition for accountability is non-trivial) is the one by Jagadessan et al. [23]. Based on an abstract labeled transition system, Jagadessan et al. proposed several candidate definitions for accountability. However, the authors themselves pointed out severe problems with all these candidates. None of these candidates captures the central intuition behind our definition that if a desired goal of the protocol is not met then some misbehaving parties are (rightly) blamed. Moreover, the framework proposed by Jagadessan et al. inherently cannot deal with (even symbolic) cryptography, as, for example, one of their propositions (Proposition 5) capturing properties of the framework would fail in presence of digital signatures.

In [1, 9], tool-supported analysis of specific properties related to accountability have been carried out for a certified email protocol and a non-repudiation protocol, respectively.

In [6], a notion related to accountability is considered in the setting of simulation-based security and tailored specifically to the problem of secure multi-party computation.

In [21], a weaker notion related to accountability, namely, *auditability* is formalized in RCF. The approach is model specific and tailored towards automatic analysis by type checking. It assumes that honest parties trigger audit actions. Also, the properties to be audited are not expressed in relation to the actual traces, but with respect to assume statements that honest and dishonest agents make, where dishonest agents may make false statements.

Auditability based on log files is considered in many papers, with various applications, including network file systems and peer-to-peer email [22], network storage services [46], and business processes [8].

In [44], three types of verifiability, namely *eligibility verifiability*, *universal verifiability*, and *individual verifiability* are formalized within the applied π-calculus (see also Section 3). These definitions are tailored to an automatic analysis and are, as the authors say, merely sufficient conditions for verifiability. Moreover, these definitions are applicable only to e-voting protocols and assume some particular structure of these protocols.

Juels, Catalano and Jakobson [26] present a cryptographic definition of verifiability, which is specifically tailored to their voting protocol [25, 26].

5. ANALYZING BINGO VOTING

In this section, we analyze accountability and verifiability properties of the Bingo voting system [10] in the cryptographic setting. Our analysis reveals some interesting new features of the system. While it turns out that the system does not provide individual accountability, the level of accountability/verifiability it provides does not depend on the random number generator used in the voting booth being honest; the numbers it produces may be predictable. Our analysis also illustrates the necessity of the parameter δ in our computational definitions of accountability and verifiability.

5.1 Informal Description of the Protocol

We denote the Bingo Voting System by $\mathsf{P}_{\mathsf{Bingo}}(n, q_{num}, q_{rec}, s, \vec{p})$, where n is the number of voters, q_{num} and q_{rec} are the probabilities that an honest voter performs the required checks (see below), s is the number of rounds in the zero-knowledge proofs, and $\vec{p} = (p_0, \ldots, p_l)$ is the probability distribution on the possible choices that a voter has, with p_0 being the probability that an honest voter abstains from voting and p_i, $i \in \{1, \ldots, l\}$, being the probability that she votes for candidate i.

In addition to the voters, the participants in this system are: (i) A *voting machine (M)*, which is the main component in the voting process. The machine uses a *bulletin board*, that everybody has read-access to, for broadcasting messages. (ii) A *random number generator (RNG)* which is an independent source of randomness, with its own display, and which is connected to the voting machine. (iii) Some number of *auditors* who will contribute randomness in a distributed way used for randomized partial checking (RPC) in the zero-knowledge proofs provided by the voting machine.

The election consists of three phases described below: initialization, voting, and tallying.

Initialization phase. In this phase, the voting machine, for every candidate j, generates n random numbers $x_1^j, \ldots, x_n^j$, along with an unconditionally hiding commitment $\mathsf{comm}(j, x_i^j)$ for each pair (j, x_i^j); more precisely, Pedersen commitments are used. All commitments are then shuffled and published on the bulletin board. Moreover, zero-knowledge proofs are published to guarantee that the same number n of commitments is created for every candidate (see Appendix A.1).

Voting phase. In this phase, a voter can enter the voting booth to indicate the candidate of her choice, say j, to the voting machine, by pressing a button corresponding to j. Note that a voter can of course also abstain from voting. Then, the RNG creates a fresh random number which is displayed to the voter and transfered to the voting machine. The machine then prints a receipt consisting of the candidate names along with the following numbers next to them: The number next to the chosen candidate is the fresh random number, where the voter is expected to check that this number is the same as the one displayed by the RNG. Next to every other candidate j', the machine prints a so far unused number $x_i^{j'}$, for some i. We assume that an honest voter checks with probability q_{num} whether the receipt shows the number displayed by the RNG at the correct position and complains publicly if this is not the case.

Tallying phase. In this phase, the voting machine first publishes the result of the election as well as all the receipts given to voters (in a lexicographical order). A voter is supposed to check whether her receipt appears on the bulletin board. We assume that a voter checks her receipt on the bulletin board with probability q_{rec}.

The machine also opens the commitments to all pairs (j, x_i^j) where the number x_i^j is unused, i.e., x_i^j has not been printed on any receipt.

Moreover, the machine provides zero-knowledge proofs to show that the commitments that it has not opened yet can be correctly assigned to the receipts, i.e., for every receipt, $l-1$ commitments (belonging to $l-1$ different candidates and different for every receipt) can be assigned to $l-1$ different candidates so that the number next to a candidate coincides with the number in the corresponding commitment. These zero-knowledge proofs are described in Appendix A.1.

Now every observer can determine the result of the election: the number of votes for candidate j is the number of opened commitments of the form $\mathsf{comm}(j, x_i^j)$, for some i, minus the number of abstaining voters.

The probability distributions $\vec{p}$ and $q_{num} \,/\, q_{rec}$ on the choices and the checks, respectively, could be generalized to model that the probabilities q_{num} and q_{rec} are not necessarily independent and, furthermore, the voters do not necessarily act independently of each other; however, we stick to the simpler case above.

5.2 Properties of the Protocol

Goal. Ideally, one might expect the system to provide individual accountability whenever the goal γ_{opt} is violated, where γ_{opt} contains all runs in which the result the machine outputs corresponds exactly to the input of all the voters. However, this goal is too strong for almost all real voting system: It is typically impossible to give any guarantees concerning dishonest voters. In fact, a dishonest voter may, for example, ignore the fact that her receipt is invalid or is not posted on the bulletin board, and she might indicate this to dishonest voting authorities/machines. Hence, the voting machine can potentially alter the dishonest voter's vote without the risk of being detected.

Therefore, the best goal γ we can hope for, in general, is that the result is correct up to the votes of dishonest voters. More formally, γ is satisfied in a run if the published result equals the actual votes of the honest voters in this run and votes of dishonest voters are distributed in some way on the candidates, possibly differently to the actual votes of the dishonest voters. This goal seems realistic and we believe that it is the goal every voting system should aim for. In particular, in the case of verifiability, if this goal is achieved, one can be sure that the votes of the honest voters are counted correctly and that every dishonest voter votes at most once.

For the analysis of voting systems it is instructive to also consider a family of goals γ_k, where γ_k coincides with γ except that up to k of the votes of *honest* voters (rather than only dishonest voters) may be altered as well; obviously $\gamma = \gamma_0$. Note that since honest voters check their receipts only with a certain probability (q_{num} and q_{rec} in our setting), undetected altering of votes by voting authorities/machines may occur, but hopefully only with a small probability.

We will define this family of goals formally below, after we have described some modeling details. Before that, however, we discuss problems with accountability that the Bingo voting system displays, which can be easily understood without the detailed definition of the goal.

Problems. *Problem 1.* If a voter v accuses the machine of not having printed the number shown by the RNG on the receipt next to the candidate chosen by v, it is unclear who cheated, unless one makes the (unrealistic) assumption that the devices keep a completely trusted log of their actions: the voter (who possibly falsely claimed something went wrong), the RNG (which possibly transmitted the wrong number to the machine), or the machine (which possibly filled out the receipt incorrectly). Hence, a judge can in this case only state $\mathsf{dis}(M) \vee \mathsf{dis}(RNG) \vee \mathsf{dis}(v)$. There are two ways to react to this statement: I) Stop the election process. However, it is difficult to draw any practical consequences from this verdict, such as punishing one of these parties. Also, the problem is that any dishonest voter could easily spoil the whole election process. II) Ignore the statement (formally, the judge should not make such a statement, even if a voter complains) and continue the election process. In this case, one has, however, to weaken the goal γ one aims for: The published result of the election can only be accurate up to honest voters who did *not* complain and, as before, dishonest voters. We discuss variant I) in more detail below; variant II) is discussed in Appendix A.4.

Problem 2. It is problematic if a number occurs twice on two different receipts, which, if parties are honest, should happen with only negligible probability: Consider the case that both the machine and the RNG are dishonest (and cooperate). The machine then can know upfront the values that the RNG will produce. Assume that the RNG will produce the number r for voter v. In this case, the voting machine could create commitments on (c,r) for *all* candidates c. Now, if v votes for some candidate c_0, the machine can print r next to c_0 on the receipt and print a fresh random number next to a different candidate. The machine can then perform correctly the ZK-proof, although it changed the vote of v. As the machine has to open all commitments (possibly after shuffling and re-randomization) it is visible that two times the same number occurs. However, the following cases could hold true: (i) the machine and the RNG are dishonest (as in the case above), (ii) the machine is honest but the RNG produced several times the same number, and (iii) the RNG is honest and the machine produced several times the same number. Hence it is not clear which individual party misbehaved. Since M and the RNG are considered to be part of the authorities, not knowing which specific device to blame is not as problematic as in the previous case.

Judging Procedure. In order to be able to formally state and prove the level of accountability the protocol provides, we first define a judging procedure, which decides whether to accept a run or whether to blame (groups of) parties. Such a procedure should, in fact, be part of the protocol specification.

The judging procedure is based solely on publicly available information, and hence, can be carried out both by an external judge and a regular protocol participant. The procedure consists of the following steps, where we assume that the procedure is run honestly by some party a. In the following, we describe the behavior of the agent a:

J1. If a participant b deviates from the protocol in an obvious way, e.g., the RNG does not display a number or the voting machine does not publish the commitments in the initialization phase, a blames the respective participant by stating the trivial formula $\mathsf{dis}(b)$. The voting machine is also blamed if a zero-knowledge proof is not correct or a voter rightly complains about her receipt, i.e., she has a receipt that is not shown on the bulletin board.

J2. If a voter v complains in the booth, a states the formula $\mathsf{dis}(M) \vee \mathsf{dis}(RNG) \vee \mathsf{dis}(v)$ as explained above (Problem 1). We denote the set of runs in which some voter complains in the booth by α_{compl}.

J3. We denote the event that a number occurs twice on two different receipts with α_{twice}. In this case, the agent a states $\mathsf{dis}(M) \vee \mathsf{dis}(RNG)$, as explained above (Problem 2).

J4. The agent a states $\mathsf{dis}(M)$ if a number occurs twice on *one* receipt or the machine opens a commitment to a number that already appears on a receipt.

J5. If none of the above happens, a accepts the run.

Modeling. The Bingo Voting system can easily be modeled as a protocol in the sense of Definition 1, where in addition to the participants mentioned in Section 5.1, we also consider a scheduler and a voting booth (see Appendix A.2 for details). We denote this protocol by $\mathsf{P}^{a}_{\mathsf{Bingo1}}(n, q_{num}, q_{rec}, s, \vec{p})$, where the agent a carries out the above judging procedure. We list some crucial security assumptions reflected in our modeling:

A1. There is only an unidirectional connection from the RNG to the machine, i.e., the machine cannot send messages to the RNG (see below for the justification).

A2. One of the auditors that contribute to the randomness used for the randomized partial checking of the zero-knowledge proofs is honest. (Clearly, if all auditors were dishonest, the machine could change the result of the election by faking the zero-knowledge proofs without being detected.)

A3. It is not feasible to forge a receipt (see below for the justification). This could be achieved by using special paper for the receipts or by means of digital signatures.

A4. The voters that enter the voting booth are counted correctly (by the voting booth); otherwise, nothing would prevent the voting machine from voting on behalf of the abstaining voters, which would further weaken the goal that can be achieved.

Note that we neither assume that the machine nor the RNG are honest. The RNG can, for example, output some predetermined sequence of numbers instead of random numbers. But then to prove accountability/verifiability for a reasonable goal, assumption A1 is crucial: If it were possible for the machine to send instructions to the RNG, both devices could cooperate to change a voter's vote, see Appendix A.2 for details.

Without assumption A3, the following problem would occur: In case a voter provides a receipt and claims that it does not appear on the bulletin board, it would not be clear whether the machine is dishonest (has not posted the legitimate receipt) or the voter is dishonest (has forged the receipt). Hence, a judge could only blame both parties, resulting in a lower level of accountability. Note that A3 is a standard and reasonable assumption.

In order to formally define the goal γ_k of the protocol, we use the following modeling detail. The only honest program of a voter (that is the only program in $\hat{\Pi}_{v_i}$, where v_i represents a voter) is of the following form. It first determines the voter's choice c. In this case, this choice is picked according to the probability distribution $\vec{p}$. (However, we could also let the adversary determine c, independently of any distribution. Theorem 1 would still be true.) Once c is picked, the voter runs the procedure $\mathsf{Vote}(c)$, which submits the choice c to the voting machine, as specified by the protocol (see Appendix A.2 for the details of Vote).

Formal definition of the goal γ_k for e-voting. Let $v_i \in \Sigma$ be a voter (recall that Σ is the set of agents of the protocol) and r be a run of an instance π (see Section 2.1). Recall that π is of the form $\pi = (\pi_{v_1} \parallel \ldots \parallel \pi_{v_n} \parallel \pi')$, where π_{v_i} is the program run by the voter v_i and the process π' contains the programs of all other participants.[1] Recall also from Section 2.1 that we say that the voter v_i is honest in a run r of the instance π if $\pi_{v_i} \in \hat{\Pi}_{v_i}$, i.e., if this voter runs its honest program; similarly for the other participants. Recall that, in this case, v_i first picks a choice c and then runs $\mathsf{Vote}(c)$. We will say that *c is the choice of the honest voter v_i in the run r*. By this, the choice of an honest voter in a run r is defined precisely. Note that the choice of an honest voter in a run directly expresses

[1]For Bingo voting, we have that $\pi' = \pi_M \parallel \pi_{A_1} \parallel \cdots \parallel \pi_{A_{r'}} \parallel \pi_{RNG} \parallel \pi_{judge} \parallel \pi_{booth} \parallel \pi_{scheduler}$, where π_M, π_{A_i}, π_{RNG}, π_{judge}, π_{booth}, and $\pi_{scheduler}$ are the programs run by the voting machine M, the auditor A_i, the RNG, the judge, the booth, and the scheduler, respectively (see also Appendix A.2). We formulate π in this way, i.e., by using π', to emphasize that the definition of γ_k does not depend on any specific protocol structure. In particular, it does not depend on a specific form of π'.

the *intention* of the voter. Clearly, this does not imply that this choice is actually counted. Whether or not it is counted depends on the correctness of the voting procedure Vote and the behavior of the remaining parties in a run. For example, dishonest authorities might try to drop or alter the vote. Also, in the case of remote electronic voting, $\mathsf{Vote}(c)$ might simply model a human actor, who indicates her choice c to a client program. This client program, which formally would be a different agent, may be malicious and try to cast a vote different to (the voter's intention) c.

Let ρ be a counting function, that is, a function which for a multiset of valid choices returns the (ideal) election result. In our case ρ simply counts and returns the number of votes for each candidate.

Now, we are ready to formally define the goal γ_k. This is a very generic and general goal, which applies to any e-voting protocol.

DEFINITION 6. Let r be a run of (some instance of) the 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 $c_1, \dots, c_{n_h}$ be the choices of the honest voters in this run, as defined above.

We say that γ_k *is satisfied in* r (or r belongs to γ_k, i.e., $r \in \gamma_k$), if there exist valid choices $\tilde{c}_1, \dots \tilde{c}_n$ such that the multiset $\{\tilde{c}_1, \dots \tilde{c}_n\}$ contains at least $n_h - k$ elements of the multiset $\{c_1, \dots c_{n_h}\}$ and the result of the election as published in r (if any) is equal to $\rho(\{\tilde{c}_1, \dots \tilde{c}_n\})$; if no election result is published in r, then γ_k is not satisfied in r.

EXAMPLE 2. Let us consider an example with 5 voters. Let r be a run with three honest and two dishonest voters such that A, A, B are the choices of the honest voters in r, respectively, 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, i.e., $r \in \gamma_1$. This is because the result is equal to $\rho(A,B,B,B,B)$ and the multiset $S = \{A,B,B,B,B\}$ contains $n_h - 1 = 2$ choices of the honest voters, namely, S contains $\{A,B\}$. However, the goal γ_0 is not satisfied in r, because there does not exist a multiset S' such that $\rho(S')$ equals the result published in r and such that S' contains all the choices $\{A,A,B\}$ of honest voters.

REMARK 2. We emphasize that in the above definition, the multiset $\{\tilde{c}_1, \dots \tilde{c}_n\}$ of choices is simply quantified existentially, independently of the specific run r. We only require that this multiset contains $n_h - k$ actual choices of the honest voters in r and that $\rho(\{\tilde{c}_1, \dots \tilde{c}_n\})$ equals the published result in r. The other $k + n_d$ choices in $\{\tilde{c}_1, \dots \tilde{c}_n\}$ can be chosen arbitrarily, and independently of r, as long as $\rho(\{\tilde{c}_1, \dots \tilde{c}_n\})$ equals the published result in r. In particular, we do not require that choices made by dishonest voters in r need to be extracted from r and that these extracted choices need to be reflected in $\{\tilde{c}_1, \dots \tilde{c}_n\}$. This is because, 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. Dishonest voters might even encourage dishonest authorities to do so in order to manipulate the election.[2]

REMARK 3. We also note that our definition of a goal makes only very minimal assumptions about the structure of a voting protocol. Namely, it requires only that, given a run r, it is possible to determine the actual choice (intention) of an honest voter (the parameter of the procedure Vote) and the actual election result as output by the voting authorities in r. Clearly, this should be possible for any reasonable voting protocol. We do not assume here anything more: we do not assume any specific phases of the protocol, nor any specific voting authorities and system components, such as a bulletin board.

Accountability. We now state the level of accountability the Bingo voting system provides. The parameter δ in the computational definition of accountability (Definition 3) will be the following:

$$\delta^k_{\mathsf{Bingo}} = \max\left(\frac{1}{2^s}, \max((1-q_{num}), (1-q_{rec}), \max_{j=1,\dots,l} p_j)^{k+1}\right),$$

where k is the parameter for the tolerated number of incorrectly counted votes of honest voters, as used for the goal γ_k, and s, q_{num}, q_{rec}, and $p_1, \dots, p_l$ are as introduced in Section 5.1.

We show (in Appendix A.3) that the protocol is accountable for Φ_1, where Φ_1 consists of the following constraints:

$$\begin{aligned} \alpha_{compl} &\Rightarrow \mathsf{dis}(M) \vee \mathsf{dis}(RNG) \vee \mathsf{dis}(v_1) \mid \dots \\ &\quad \cdots \mid \mathsf{dis}(M) \vee \mathsf{dis}(RNG) \vee \mathsf{dis}(v_n) \\ \alpha_{twice} &\Rightarrow \mathsf{dis}(M) \vee \mathsf{dis}(RNG), \\ \neg\gamma_k \cap \neg\alpha_{compl} \cap \neg\alpha_{twice} &\Rightarrow \mathsf{dis}(M) \mid \mathsf{dis}(RNG). \end{aligned}$$

THEOREM 1. *Let a be an external judge or a voter. Under the DLOG-assumption*[3]*, the agent a ensures* $(\Phi_1, \delta^k_{\mathsf{Bingo}})$*-accountability for* $\mathsf{P}^a_{\mathsf{Bingo1}}(n, q_{num}, q_{rec}, s, \vec{p})$.

This theorem says that, in $\mathsf{P}^a_{\mathsf{Bingo1}}$, the probability that the goal γ_k is not achieved and a does not blame anybody is at most $\delta^k_{\mathsf{Bingo}}$, up to some negligible value. Moreover, a single agent can be held accountable (and because of fairness rightly so) if, in the case the goal is not achieved, no voter complains in the booth and no number occurs twice on receipts.

We emphasize that the above theorem includes the case where the RNG produces a totally predictable sequence of random numbers. If we had assumed an honest RNG, we could have omitted the term $\max_{j=1,\dots,l} p_j$ in the definition of $\delta^k_{\mathsf{Bingo}}$ in the above theorems. Also, we note that from the proof of Theorem 1 it follows that the parameter $\delta^k_{\mathsf{Bingo}}$ is optimal, i.e., there is a (misbehaving) voting machine which changes $k+1$ votes but is detected only with probability $\delta^k_{\mathsf{Bingo}}$.

Verifiability. Let us observe that, since J ensures $(\Phi_1, \delta^k_{\mathsf{Bingo}})$-accountability, J also ensures $(\neg\gamma \Rightarrow \psi)$-accountability, where $\psi = \bigvee_{a \in \Sigma} \mathsf{dis}(a)$. Also, whenever J states ψ', then ψ' implies ψ. Therefore, due to the fact that the judging procedure is constructed in such a way that J accepts the run if and only if J does not blame anybody, by Proposition 1, we immediately obtain the following result.

COROLLARY 1. *Let a be an external judge or a voter. Under the DLOG-assumption, in* $\mathsf{P}^a_{\mathsf{Bingo1}}(n, q_{num}, q_{rec}, s, \vec{p})$*, the goal* γ_k *is guaranteed by* $\bigwedge_{a \in \Sigma} \mathsf{hon}(a)$ *and* $\delta^k_{\mathsf{Bingo}}$*-verifiable by a.*

[2]For specific protocols, in some cases, one could provide slightly stronger guarantees than what is required by γ_k, though. If, for example, we assume an e-voting system with a bulletin board to which voters submit their ballots along with zero-knowledge proofs of knowledge of the submitted (valid) votes, we could, in addition to what is required by γ_k, also require that the published results equals (possibly again up to a certain number of votes) the result that can be extracted from the (valid) ballots on the bulletin board; the latter is typically referred to as universal verifiability (see also Section 3). Note that a ballot which appears on the bulletin board but which has not been submitted by an honest voter might not actually have been submitted by a dishonest voter either but might have been placed on the bulletin board by a dishonest voting authority, say, possibly replacing a ballot submitted by a dishonest voter.

[3]From this assumption, it follows that it is infeasible to open a Pedersen-commitment to two different values [39].

This corollary says that, in $\mathsf{P}^a_{\mathsf{Bingo1}}$, correctness of the result (up to votes of dishonest voters) is guaranteed only if *all* participants are honest and is $\delta^k_{\mathsf{Bingo}}$-verifiable by a (recall that a uses only public information). This means that a, with overwhelming probability, accepts a run if *everybody* is honest, but he/she accepts a run only with probability at most $\delta^k_{\mathsf{Bingo}}$ if the result is not correct (up to votes of dishonest voters).

This verifiability property reflects the weakness of the system $\mathsf{P}^a_{\mathsf{Bingo1}}(n, q_{num}, q_{rec}, s, \vec{p})$ already revealed by Theorem 1: By wrongly complaining, every single dishonest voter can spoil the election process. This weakness is not present in the version mentioned above, that we study in Appendix A.4, which, however, comes at a price of a weaker goal.

6. THE PRST PROTOCOL

In this section, we study the auction protocol proposed by Parkes, Rabin, Shieber, and Thorpe [38]. More precisely, we study here one of a few variants of the protocol proposed in [38], namely the variant for Vickrey auctions with one item and without so-called delayed decryption key revelation services; our definition also applies to the other variants, though. We carry out our analysis in a symbolic (Dolev-Yao style) model.

While applying our definition of accountability to this protocol, we identified some quite serious problems that allow parties to misbehave and spoil the complete auction process, without facing the risk of being held individually accountable. We propose fixes to the original protocol in order to establish individual accountability and make the protocol useable.

6.1 Informal Description of the Protocol

The protocol assumes a public key infrastructure. In particular, only bidders with registered signature keys can participate in the protocol. The protocol uses digital signatures, a hash function (used to produce commitments[4]), homomorphic randomized encryption (more specifically, Paillier encryption), and non-interactive zero-knowledge proofs for proving correctness of the result (see below).

By $\mathsf{sig}_A[m]$ we abbreviate the message $\langle m, \mathsf{sig}_A(m)\rangle$, where $\mathsf{sig}_A(m)$ is a term representing the signature of A on the message m. By $E_A(m,r)$ we will denote encryption of a message m under the public key of A with random coins r. By $\mathsf{hash}(m)$ we denote the hash of m.

The parties of the protocol are the following: the bidders $B_1, \dots, B_n$, the auctioneer A, and the notaries $N_1, \dots, N_l$. The auctioneer maintains a bulletin board, where he posts all public information about the auction. All posts to the bulletin board carry appropriate digital signatures.

The protocol consists of the following steps. For simplicity of presentation, in the description of the protocol given below, we assume that all the entitled bidders $B_1, \dots, B_n$ participate in the auction and that all their bids are different; this convention is not essential and can easily be dropped. Also, for simplicity, we have left out some additional input provided by the parties for the zero-knowledge proof, since in our symbolic modeling of zero-knowledge proofs this input is not needed (see [38] for details).

S1. A posts (on the bulletin board) basic information about the auction: the terms of the auction, an identifier Id, the deadlines T_1, T_2, T_3 for different stages of the auction, and his public encryption key.

S2. To participate in the auction, a bidder B_i chooses her bid b_i and encrypts it as $C_i = E_A(b_i, r_i)$ using a random coin r_i. B_i then commits to C_i, computing $Com_i = \langle \mathsf{hash}(C_i), Id\rangle$, signs this commitment, and sends $\mathsf{sig}_{B_i}[Com_i]$ to A and her notaries, if used, before time T_1. The notaries forward the signed commitments to A. A replies by sending a signed receipt $R_i = \mathsf{sig}_A[Com_i, Id, T_1]$ to B_i. If B_i does not obtain her receipt, she complains.

S3. At time T_1, the auctioneer A posts all the received commitments in a random order: $Com_{\pi(1)}, \dots, Com_{\pi(n)}$, where π is a randomly chosen permutation of the indices of submitted commitments.

S4. Between time T_1 and T_2 any bidder B_i who has a receipt R_i for a commitment which is not posted can appeal her non-inclusion (by providing her receipt).

S5. After time T_2, every B_i sends to A her encrypted bid C_i. After time T_3, A posts $C_{\pi(1)}, \dots, C_{\pi(n)}$. Anybody can verify whether all the commitments posted in S3 have been correctly opened.

S6. A recovers the bids $b_1, \dots, b_n$, by decrypting the encrypted bids with his private decryption key, and determines the winner B_w of the auction and the price b_u the winner has to pay, which is supposed to be the second highest bid. He also constructs a (universally verifiable) zero-knowledge proof P that the result is correct, i.e. C_w contains the biggest bid and C_u contains the second biggest bid b_u: This is done by proving appropriate inequalities between the bids in the ciphertexts $C_1, \dots, C_n$, without revealing these bids, and by revealing the random coin used in C_u, which he can recover using his private key. The auctioneer posts

$$B_w,\ b_u,\ \mathsf{sig}_{B_w}[Com_w],\ P. \tag{6}$$

[4]A hash function is used to commit on values with high entropy.

6.2 Properties of the Protocol

In this section, we state accountability and verifiability properties of the protocol.

Goal. The protocol should satisfy the goal γ which, informally, is achieved in a run if the protocol successfully produces a result which is correct with respect to the committed bids. Note that in a run the committed bids are (computationally) determined by the commitments to the encrypted bids $C_1, \dots, C_n$. Now, more precisely, γ requires that (i) all the submitted commitments are different, (ii) the result is published and the published price b_u is the second highest bid amongst the bids encrypted in $C_1, \dots, C_n$, and (iii) an honest bidder is declared to be the winner if and only if her bid is the highest in $C_1, \dots, C_n$.

Conditions (ii) and (iii) capture that the announced result corresponds to the bids committed by the bidders. In addition, condition (i) prevents that a dishonest bidder B_j who somehow got to know the commitment of another bidder B_i (e.g., a dishonest auctioneer revealed the commitment to B_j) can place the same bid as B_i, without even knowing it. This problem was not considered in [38].

Ideally, we would hope that the protocol satisfies individual accountability, i.e., if the goal is not achieved, then individual parties can be (rightly) blamed for this. Unfortunately, as our analysis reveals, the protocol does not guarantee this strong level of accountability, due to the following problems, which will be reflected in the accountability property we prove for this protocol.

Problems. In the following, for a set of agents A, let ψ^*_X be the verdict stating that all but possibly one agent in X misbehaved. For instance, $\psi^*_{\{a,b,c\}} = (\mathsf{dis}(a) \wedge \mathsf{dis}(b)) \vee (\mathsf{dis}(a) \wedge \mathsf{dis}(c)) \vee (\mathsf{dis}(b) \wedge \mathsf{dis}(c))$.

Problem 1. This problem boils down to the fact that the protocol does not offer any effective mechanism for non-repudiable communication, even though the notaries were introduced for this purpose: If (a) a bidder B_i claims that she did not obtain her receipt after she had sent her signed commitment to the auctioneer in Step S2 and (b) the auctioneer claims that he did not obtain the signed commitment from the bidder, then it is impossible to resolve the dispute. Therefore, in such a case, the judge can only state $\mathsf{dis}(A) \vee \mathsf{dis}(B_i)$.

A similar problem occurs if, after Step S5, a bidder B_i claims that her encrypted bid C_i has not been posted on the bulletin board and A claims that he has not received this bid. Again, it is impossible to resolve the dispute. This problem is more serious than the previous one, as at this point the auctioneer knows all the values of the bids and the corresponding bidders, and he may have an interest in manipulating the auction. It is also a good opportunity for a dishonest bidder to disturb the auction process.

Problem 2. If two (or more) commitments posted in Step S3 have the same value, then it is not clear who is to be blamed, even if the auctioneer provided the signatures of the bidders on these commitments. In fact, it is possible that one of the these bidders B_i honestly followed the protocol, but the auctioneer forwarded her commitment to the other bidders who submitted this commitment with their own signatures. It may, however, as well be the case that A is honest, but all the mentioned bidders are dishonest and submitted the same commitment.

Problem 3. A quite serious problem occurs at the end of the auction. Suppose that the auctioneer posts a result as in (6), for some w, u, with a correct zero-knowledge proof P. Suppose also that some bidder $B_j \neq B_w$ claims that C_w is *her* encrypted bid. Then, even if we assume that the judge requests both B_w and B_j to send him their receipts and to prove their knowledge of the random coin r_w used in C_w, the judge is not able to blame a specific party. In fact, all the following scenarios are possible: (1) A is honest and B_w, B_j are dishonest: B_w submits the commitment for C_w and then forwards to B_j her receipt and the random coin r_w. (2) B_w is honest and A, B_j are dishonest: A provides B_j with the receipt R_w of bidder B_w and her random coin r_w; note that A can extract the random coin from C_w. (3) B_j is honest and A, B_w are dishonest: B_j submits her commitment, obtains her receipt, but A declares that B_w is the winner, providing B_w, as above, with the receipt of B_j and her random coin.

This is a serious problem, since a judge cannot blame a specific party among the parties A, B_w, and B_j; he can only state the verdict $\psi^*_{\{A,B_w,B_j\}}$ and cannot determine who actually won the auction.

Judging Procedure. In order to be able to formally state and prove the level of accountability the protocol provides, we first define a judging procedure, which decides whether to accept a run or whether to blame (groups of) parties. Such a procedure should, in fact, be part of the protocol specification.

The judging procedure is based solely on publicly available information, and hence, can be carried out both by an external judge and a regular protocol participant. The procedure consists of the following steps, where we assume that the procedure is run by some party V.

V1. If a bidder B_i complains in Step S2, then V states $\mathsf{dis}(A) \vee \mathsf{dis}(B_i)$ (Problem 1).

V2. If A does not publish the list of commitments when expected (Step S3), then V blames A (states $\mathsf{dis}(A)$). If A posts this list, but, for $l > 1$, l commitments have the same value (Problem 2), then A is requested to provide signatures of l bidders $B_{i_1}, \dots, B_{i_l}$ on these commitments. If A refuses to do so, V blames A; otherwise, V states $\psi^*_{\{A,B_{i_1},\dots,B_{i_l}\}}$.

V3. If, in Step S4, B_i posts a receipt without a corresponding commitment posted by A in the previous step, V blames A.

V4. In Step S5, if some previously posted commitment Com_i is not opened, A should provide the signature of B_i on Com_i. If A does not provide the requested signature, V blames him. Otherwise, V states $\mathsf{dis}(A) \vee \mathsf{dis}(B_i)$ (Problem 1).

V5. If, in Step S6 A, does not post a result with a valid zero-knowledge proof and a valid signature $\mathsf{sig}_w[Com_w]$, then V blames A.

V6. If, after Step S6, some bidder B_j with $j \neq w$ complains and provides a receipt of A on Com_w as well as the random coins for Com_w, then V states the verdict $\psi^*_{\{A,B_w,B_j\}}$ (Problem 3).

V7. If none of the above happens, then V accepts the run.

Modeling. We consider a few variants of the protocol: By P^J_{PRST} we denote the version of the protocol with an additional party, the *judge*. This party is assumed to be honest and run the judging procedure described above. By P^X_{PRST}, for $X \in \{B_1, \dots, B_n\}$, we denote the version of the protocol, where X is assumed to be honest and his/her honest program is *extended* by the judging procedure (i.e. X, in addition to his/her protocol steps, also carries out the judging procedure). In each of these systems, besides X also the bulletin board is assumed to be honest. All the remaining parties are not assumed to be honest. For a detailed modeling of these systems (in a symbolic setting) see Appendix B.2.

Accountability Property. Now, we define the accountability property of the protocol. Let α^i_{rec} be the set of runs where B_i claims that she has sent her signed commitment in Step S2, but has not obtained her receipt (Problem 1). Let α^i_{open} be the set of runs where some commitment Com_i is not opened in Step S5 and A provides the signature of B_i on this commitment (Problem 2). Let α^X_{reuse}, where X is a set of at least two bidders, be the set of runs where A, as described in Step V2, reveals signatures of all the bidders in X on the same commitment. Finally, let $\alpha^{w,j}_{win}$ be the set of runs where the auctioneer posts a result of the form (6), for some w, u, with a correct zero-knowledge proof P and some bidder $B_j \neq B_w$ claims that C_w is her bid and provides the receipt of A on Com_w as well as the random coins of C_w (Problem 3). Let $\neg\alpha$ denotes the set of runs which are not in α^i_{rec}, α^i_{open}, α^X_{reuse}, and $\alpha^{w,j}_{win}$, for any i, j, w, X.

We will show that the protocol is accountable for Φ, where Φ consists of the following constraints:

$$\alpha^i_{rec} \Rightarrow \mathsf{dis}(B_i) \vee \mathsf{dis}(A) \quad \text{for all } i \in \{1, \dots, n\}, \tag{7}$$

$$\alpha^i_{open} \Rightarrow \mathsf{dis}(B_i) \vee \mathsf{dis}(A) \quad \text{for all } i \in \{1, \dots, n\}, \tag{8}$$

$$\alpha^X_{reuse} \Rightarrow \psi^*_{X \cup \{A\}} \quad \text{for all } X \subseteq \{B_1, \dots, B_n\}, |X| > 1 \tag{9}$$

$$\alpha^{w,j}_{win} \Rightarrow \psi^*_{\{A,B_w,B_j\}} \quad \text{for all } w, j \in \{1, \dots, n\}, \tag{10}$$

$$\neg\alpha \cap \neg\gamma \Rightarrow \mathsf{dis}(A). \tag{11}$$

Note that, amongst the above accountability constraints, only (11) provides individual accountability.

THEOREM 2. *Let $V \in \{J, B_1, \dots, B_n\}$. V ensures Φ-accountability for P^V_{PRST}.*

The proof of this theorem is given in Appendix B.5. This theorem guarantees that whenever the goal γ is not satisfied, agent V states some verdict, where the agent A is held accountable individually if none of the cases α^i_{rec}, α^i_{open}, α^X_{reuse}, and $\alpha^{w,j}_{win}$ occurs. As explained, occurrence of $\alpha^{w,j}_{win}$ is very problematic.

Verifiability. As in Section 5.2, by Proposition 1, we immediately obtain the following result.

COROLLARY 2. *The goal γ is guaranteed in P^V_{PRST} by $\mathsf{hon}(A) \wedge \mathsf{hon}(B_1) \wedge \cdots \wedge \mathsf{hon}(B_n)$ and verifiable by V, for any $V \in \{J, B_1, \ldots, B_n\}$.*

6.3 Our Improved Version

We now propose fixes to the original auction protocol in order to establish individual accountability and make the protocol useable. In this section, we only briefly sketch these fixes, with the detailed description of our version of the protocol presented in the Appendix B.1.

For our protocol, we assume an *independent and honest bulletin board* (replacing the bulletin board controlled by the auctioneer), where the auctioneer and the bidders can post messages. Now, every bidder, instead of sending her signed commitment $\mathsf{sig}_{B_i}[Com_i]$ to the auctioneer in Step S2, posts the message $E_A(\mathsf{sig}_{B_i}[Com_i], r'_i)$ (for some random coin r'_i) on the bulletin board. Similarly, instead of sending the encrypted bid to A in Step S5, a bidder posts her encrypted bid on the bulletin board. One can show that this enables the judge to resolve the disputes described in Problems 1 and 2.

To prevent Problem 3, we modify the commitment Com_i of B_i: In our version, $Com_i = \langle \mathsf{hash}(C_i), \mathsf{hash}(q_i), Id \rangle$, where q_i is a random nonce generated by B_i. The bidder is supposed to keep the nonce q_i secret, except for using it to resolve the dispute described in Problem 3: If B_j notices that the commitment signed by B_w in (6) is her own commitment, B_j posts q_j on the bulletin board; resulting in B_w being blamed.

We prove that our version $P^V_{PRST'}$ of the protocol provides a high level of accountability: individual parties are held accountable whenever in a protocol run the goal γ is not achieved, where γ is defined as in Section 6.2. Let Φ' consist of only one individual accountability constraint: $\neg\gamma \Rightarrow \mathsf{dis}(A) \mid \mathsf{dis}(B_1) \mid \cdots \mid \mathsf{dis}(B_n)$. We have the following result (see Appendix B.4 for the proof).

THEOREM 3. *Let $V \in \{J, A, B_1, \ldots, B_n\}$. V ensures Φ'-accountability for protocol $P^V_{PRST'}$.*

As in the case of the original version of the protocol, the accountability property stated in Theorem 3 allows us to immediately obtain the corresponding verifiability property of our version of the protocol. It is interesting to observe that, even though the two versions of the protocol enjoy very different levels of accountability, the verifiability properties for both of them are exactly the same. In fact, in both protocols, any dishonest bidder can spoil the auction procedure and, therefore, the goal needs to be guaranteed by all the participants. This, again, illustrates that verifiability is too coarse a notion and is not able to distinguish between protocols that provide strong incentives for the participants to behave honestly from those that do not provide such incentives.

7. ASW PROTOCOL

In this section, we study accountability properties of the ASW optimistic contract-signing protocol [4] in the symbolic setting. We only sketch the protocol and our results (see Appendix D for details).

Description of the Protocol. The objective of the ASW protocol is to enable two parties, A (the originator) and B (the responder), to obtain each other's signature on a previously agreed contractual text with the help of a trusted third party T, who, however, is only invoked in case of a problem. In the intended run of the protocol, A first indicates her willingness to sign the contract to B, by sending a message m_1. Then B sends his willingness to sign the contract to B (m_2). Next, A sends a message m_3 that together with m_1 forms a valid signature. Finally, B sends a message m_4 that, again, together with m_2 forms a valid signature to A. If after A has sent m_1, B does not respond, A may contact T to obtain an *abort token* a_T. If after A has sent m_3, she does not obtain the signature from B, A may ask T for a *replacement contract* r_T; analogously for B. Once T issued a_T (r_T), T should never issue r_T (a_T) afterwards.

Properties of the Protocol. We are interested in the accountability of T. Ideally, we would like to hold T accountable whenever it produces both a_T and r_T. However, this is unrealistic: T could produce a_T and r_T and never send these messages to anybody. We therefore consider only the case where there is a dispute in which the judge is faced with both a_T and r_T. More precisely, by P_{ASW} we denote the protocol (in the sense of Definition 1) modeling the ASW protocol, where, in addition to A, B and T, we consider an additional party, the *judge* J. The honest programs of A, B, and T run each one instance of their role, as specified by the protocol, where T can deal with up to three requests. The judge, who is assumed to be honest, blames T if and only if he obtains a message of the form $\langle a_T, r_T \rangle$ for some contract. For P_{ASW}, we define α to be the set of runs where J obtains a message of the form $\langle a_T, r_T \rangle$ for some contract, modeling that J is faced with both a_T and r_T. Let Φ consist of the accountability constraint $\alpha \Rightarrow \mathsf{dis}(T)$. We obtain the following theorem:

THEOREM 4. *J ensures Φ-accountability for P_{ASW}.*

Following Remark 1, we verified Theorem 4 automatically using the protocol analysis tool by Millen and Shmatikov [36] (see [33] for our formal modeling). As mentioned, the completeness condition is rather trivial in this case.

8. REFERENCES

[1] M. Abadi and B. Blanchet. Computer-assisted verification of a protocol for certified email. *Sci. Comput. Program.*, 58(1-2):3–27, 2005.

[2] 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.

[3] B. Adida and R. L. Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In *Workshop on Privacy in the Electronic Society (WPES 2006)*, pages 29–40, 2006.

[4] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99. IEEE Computer Society, 1998.

[5] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.

[6] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In Salil P. Vadhan, editor, *Proceedings of the 4th Theory of Cryptography Conference,(TCC 2007)*, volume 4392 of *Lecture Notes in Computer Science*, pages 137–156. Springer, 2007.

[7] M. Backes, M. Maffei, and D. Unruh. Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol. In *2008 IEEE Symposium on Security and Privacy (S&P 2008)*, pages 202–215. IEEE Computer Society, 2008.

[8] Adam Barth, John C. Mitchell, Anupam Datta, and Sharada Sundaram. Privacy and utility in business processes. In *20th*

IEEE Computer Security Foundations Symposium (CSF 2007), pages 279–294. IEEE Computer Society, 2007.

[9] Giampaolo Bella and Lawrence C. Paulson. Accountability protocols: Formalized and verified. *ACM Trans. Inf. Syst. Secur.*, 9(2):138–161, 2006.

[10] J.-M. Bohli, J. Müller-Quade, and S. Röhrich. Bingo Voting: Secure and Coercion-Free Voting Using a Trusted Random Number Generator. In A. Alkassar and M. Volkamer, editors, *E-Voting and Identity (VOTE-ID 2007)*, volume 4896 of *Lecture Notes in Computer Science*, pages 111–124. Springer, 2007.

[11] D. Chaum. `http://punchscan.org/`.

[12] 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)*, 2008.

[13] D. Chaum, P.Y.A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS 2005)*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.

[14] B. Chevallier-Mames, P.-A. Fouque, D. Pointcheval, J. Stern, and J. Traoré. On Some Incompatible Properties of Voting Schemes. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, 2006.

[15] J. Clark, A. Essex, and C. Adams. Secure and Observable Auditing of Electronic Voting Systems using Stock Indices. In *Proceedings of the Twentieth IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2007)*, 2007.

[16] 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.

[17] J.A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Advances in Cryptology – CRYPTO'99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. Springer-Verlag, 1999.

[18] O. Goldreich. *Foundations of Cryptography*, volume 1. Cambridge Press, 2001.

[19] Vipul Goyal. Reducing trust in the pkg in identity based cryptosystems. In *Proceedings of the 27th Annual International Cryptology Conference (CRYPTO 2007)*, volume 4622 of *Lecture Notes in Computer Science*, pages 430–447. Springer, 2007.

[20] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box accountable authority identity-based encryption. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *Proceedings of the 2008 ACM Conference on Computer and Communications Security (CCS 2008)*, pages 427–436. ACM, 2008.

[21] Nataliya Guts, Cédric Fournet, and Francesco Zappa Nardelli. Reliable evidence: Auditability by typing. In Michael Backes and Peng Ning, editors, *Proceedings of the 14th European Symposium on Research in Computer Security (ESORICS 2009)*, volume 5789 of *Lecture Notes in Computer Science*, pages 168–183. Springer, 2009.

[22] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. Peerreview: practical accountability for distributed systems. In Thomas C. Bressoud and M. Frans Kaashoek, editors, *Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007, SOSP 2007*, pages 175–188. ACM, 2007.

[23] Radha Jagadeesan, Alan Jeffrey, Corin Pitcher, and James Riely. Towards a theory of accountability and audit. In Michael Backes and Peng Ning, editors, *Proceedings of the 14th European Symposium on Research in Computer Security (ESORICS 2009)*, volume 5789 of *Lecture Notes in Computer Science*, pages 152–167. Springer, 2009.

[24] Wei Jiang, Chris Clifton, and Murat Kantarcioglu. Transforming semi-honest protocols to ensure accountability. *Data Knowl. Eng.*, 65(1):57–74, 2008.

[25] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Proceedings of Workshop on Privacy in the Eletronic Society (WPES 2005)*. ACM Press, 2005.

[26] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. Cryptology ePrint Archive, Report 2002/165, 2002. `http://eprint.iacr.org/`.

[27] D. Kähler, R. Küsters, and T. Truderung. Infinite State AMC-Model Checking for Cryptographic Protocols. In *Proceedings of the Twenty-Second Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 181–190. IEEE, Computer Society Press, 2007.

[28] D. Kähler, R. Küsters, and Th. Wilke. A Dolev-Yao-based Definition of Abuse-free Protocols. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Proceedings of the 33rd International Colloqium on Automata, Languages, and Programming (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 95–106. Springer, 2006.

[29] S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In *Computer Security Foundations Workshop 2002 (CSFW 2002)*, pages 206–220. IEEE Computer Society, 2002.

[30] R. Küsters. Simulation-Based Security with Inexhaustible Interactive Turing Machines. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW-19 2006)*, pages 309–320. IEEE Computer Society, 2006.

[31] R. Küsters and T. Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *2009 IEEE Symposium on Security and Privacy (S&P 2009)*, pages 251–266. IEEE Computer Society, 2009.

[32] R. Küsters and T. Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. Technical Report arXiv:0903.0802, arXiv, 2009. Available at `http://arxiv.org/abs/0903.0802`.

[33] R. Küsters, T. Truderung, and A. Vogt. Automated verification of asw. Available at `http://infsec.uni-trier.de/publications/software/KuestersTruderungVogt-ASW-2010.zip`.

[34] 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.

[35] J. Millen and V. Shmatikov. Constraint Solver, a protocol security analyzer. Available at `http://www.homepages.dsu.edu/malladis/research/ConSolv/Webpage/`.

[36] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In

Proceedings of the 8th ACM conference on Computer and Communications Security, pages 166–175. ACM Press, 2001.

[37] T. Moran and M. Naor. Split-ballot voting: everlasting privacy with distributed trust. In P. Ning, S. De Capitani di Vimercati, and P. F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007*, pages 246–255. ACM, 2007.

[38] D. Parkes, M. Rabin, S. Shieber, and C. Thorpe. Practical secrecy-preserving, verifiably correct and trustworthy auctions. In *Proceedings of the Eighth International Conference on Electronic Commerce (ICEC'06)*, pages 70–81, 2006.

[39] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference (CRYPTO 1991)*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

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

[41] P. Y. A. Ryan and S. A. Schneider. Prêt à Voter with Re-encryption Mixes. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *Proceedings of the European Symposium on Research in Computer Security (ESORICS 2006)*, volume 4189 of *Lecture Notes in Computer Science*, pages 313–326. Springer, 2006.

[42] K. Sako and J. Kilian. Receipt-Free Mix-Type Voting Scheme — A practical solution to the implementation of a voting booth. In *Advances in Cryptology — EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer-Verlag, 1995.

[43] V. Shmatikov and J.C. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science (TCS), special issue on Theoretical Foundations of Security Analysis and Design*, 283(2):419–450, 2002.

[44] B. Smyth, M. Ryan, S. Kremer, and M. Kourjieh. Election verifiability in electronic voting protocols. In *Proceedings of the 4th Benelux Workshop on Information and System Security, WISSec 2009*, 2009.

[45] Mehdi Talbi, Benjamin Morin, Valérie Viet Triem Tong, Adel Bouhoula, and Mohamed Mejri. Specification of electronic voting protocol properties using adm logic: Foo case study. In Liqun Chen, Mark Dermot Ryan, and Guilin Wang, editors, *Proceedings of the 10th International Conference Information and Communications Security (ICICS 2008)*, volume 5308 of *Lecture Notes in Computer Science*, pages 403–418. Springer, 2008.

[46] Aydan R. Yumerefendi and Jeffrey S. Chase. Strong accountability for network storage. *ACM Transactions on Storage (TOS)*, 3(3), 2007.

[47] Li Yunfeng, He Dake, and Lu Xianhui. Accountability of perfect concurrent signature. *International Conference on Computer and Electrical Engineering*, 0:773–777, 2008.

[48] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 55–61. IEEE Computer Society Press, 1996.

APPENDIX

A. THE BINGO VOTING SYSTEM

A.1 Zero-knowledge Proofs for Bingo Voting

Here we describe the zero-knowledge proofs used in the tallying phase and the initialization phase of the Bingo Voting system in more detail.

ZK-proofs in the tallying phase. The following steps are performed for every receipt: First, the voting machine generates a new commitment on the pair (j,r), where j is the chosen candidate and r is the number generated by the RNG and printed next to j. Then, all the commitments for the receipt are published in a random order: one of them is the commitment just described, the other $(l-1)$ commitments are unopened commitments published on the bulletin board in the initialization phase, where for different receipts, different commitments are taken from the bulletin board. An observer can verify that this is the case. Next, these commitments are re-randomized and shuffled twice; both the intermediate and the final set of commitments are published. The final commitments are opened. Now an observer can check that there is exactly one commitment for each candidate and that all numbers shown on the receipt were in fact contained in the final commitments. Finally, the auditors choose a random bit in some distributed way, see, e.g., [15]. Depending on the value of this bit, the voting machine publishes the random factors for the first or for the second re-randomization step.

If the voting machine would try to cheat, this would be detected with a probability of at least 50%; this probability can be increased to $1-(\frac{1}{2})^s$ by repeating the procedure s times.

ZK-proofs in the initialization phase. This proof was not precisely defined in [10], but it can be implemented by randomized partial checking similarly to the zero-knowledge proof in the tallying phase. To this end, we assume that a commitment $\mathsf{comm}(j,x_i^j)$ on a pair (j,x_i^j) is implemented as $(C_i^j,D_i^j)=(\mathsf{comm}(j),\mathsf{comm}(x_i^j))$, where the commitments on the single components are Pedersen commitments. Now, to show that among the published commitments there are exactly n of the form $\mathsf{comm}(j,x_i^j)$ for every candidate j, the zero-knowledge proof proceeds similarly as in the tallying phase, except that it only uses the first component C_i^j of a commitment.

A.2 Modelling of the Bingo Voting Protocol

The modelling of the Bingo Voting system, is based on a quite standard computational model, similar to models for simulation-based security (see, e.g., [30]), in which *inexhaustible interactive Turing machines (IITMs)* communicate via tapes. In this model, only one IITM is active at a time. Such a machine may perform computations polynomially bounded by the security parameter and the input of its enriching tapes. It may send a message to another IITM which is then triggered.

There is one special IITM, called the master, which is triggered first. This machine is also triggered if, at some point, no message is sent.

Here, every IITM only has so-called *consuming* tapes. By this, it is guaranteed that polynomially bounded IITMs plugged together form a polynomially bounded system. See [30] for a detailed discussion of this issue.

We define $\mathsf{P}^a_{\mathsf{Bingo1}}$ in the following way as the protocol $(\Sigma,\mathsf{Ch},\mathrm{In},\mathrm{Out},\{\Pi_a\}_{a\in\Sigma},\{\hat{\Pi}_a\}_{a\in\Sigma})$:

Participants. The set Σ of the protocol participants consists of the voters $v_1, \dots, v_n$, the voting machine M, the auditors $A_1, \dots, A_{r'}$, the random number generator RNG, the judge a, and, additionally, *the scheduler* $\mathscr{S}$ and the *voting booth* $\mathscr{B}$. The role of the scheduler (who formally will be the master) is to make sure that every party gets a chance to perform some actions in every protocol phase. It is important, because otherwise we could not blame parties who did not perform required actions (for instance, did not open their commitments, as required, in the appropriate protocol stage). $\mathscr{B}$ models the voting booth, including the Bulletin Board.

Channels. The set of channels (correspond to tapes in the IITM model) we consider here includes the channels ch_b^a, for every $a, b \in \Sigma \setminus \{RNG\}$. The channel ch_b^a is an output channel of a and an input channel of b. Therefore a and b can communicate using ch_b^a and ch_a^b. Further, the RNG is connected to the voting booth $\mathscr{B}$ and there is a channel ch_M^{RNG}, modelling that the RNG can only be triggered when a voter indeed is physically in the voting booth. In particular, as we will assume that $\mathscr{B}$ is honest, the voting machine M cannot send messages to the RNG. This reflects security assumption A1.

If it were possible for the machine to send instructions to the RNG, both devices could cooperate to change a voter's vote, from, say, candidate i to candidate j in the following way: The machine chooses an unopened commitment on (i, x), for some x, and instruct the RNG to display x as the freshly generated number. Then the machine chooses a fresh random number y and writes y next to j and x next to i. By this, the machine produces a valid ballot for candidate j without the risk of being detected.

Sets of programs Π_a. First, we define the sets of honest programs of the protocol participants.

The RNG. The RNG is triggered by $\mathscr{B}$ when a voter is in the voting booth (see below). The honest program of the RNG then creates a fresh random number and sends this number to $\mathscr{B}$. We assume that the probability that two randomly picked numbers coincide is negligible.

The Voter. The IITM running the honest program of a voter waits until it is triggered by the scheduler. It then determines the voter's choice c according to the probability distribution $\vec{p}$ and runs the following procedure $\mathsf{Vote}(c)$. It does nothing, if the choice c is to abstain (which happens with probability p_0). Otherwise, the voter enters the voting booth by sending an `enter`-message to $\mathscr{B}$. After being triggered by $\mathscr{B}$, it sends the chosen candidate c to $\mathscr{B}$. $\mathscr{B}$ forwards the number from the RNG and the receipt to the voter. The voter checks with probability q_{num} whether the number on the receipt next to the chosen candidate corresponds to the number from the RNG. If this is not the case, the voter sends a `complain`-message to the judge. Further, in the tallying phase, with probability q_{rec}, the voter checks the receipt, i.e. demands from $\mathscr{B}$ the information on the Bulletin Board and compares that with the receipt that she received from $\mathscr{B}$ in the voting phase. If the receipts does not match, the voter sends the receipt to the judge.

The Machine. The honest program of the voting machine is described in the protocol description, where the machine publishes everything that is to be published (e.g. the commitments and the receipts) by sending these values to $\mathscr{B}$ (see below).

The Auditors. The honest program of an auditor picks, always when it is triggered by the scheduler, a random bit and sends this bit to the scheduler.

The Judge. The honest program of the judge is specified by the judging procedure.

The Scheduler. As we have mentioned in every instance of the protocol, the processes of the scheduler is the master. The role of the scheduler, as we have also mentioned, is to trigger every party, so that it is given a chance to perform the actions required by the protocol. We assume that the run is stopped if the judge outputs something on his $\mathsf{decision}$-tape.

The honest program of $\mathscr{S}$ is as follows:

- It starts the system by first triggering the voting booth $\mathscr{B}$ that is in charge of choosing the setup parameters (see below).
- It then triggers the voting machine M which is supposed to send the commitments (according to the protocol specifications) to the voting booth $\mathscr{B}$ (recall that $\mathscr{B}$ plays the role of the Bulletin Board), who confirms to $\mathscr{S}$ that this has been done correctly. If $\mathscr{S}$ does not receive a confirmation, it sends a message to the judge a who then outputs $\mathsf{dis}(M)$ on the tape $\mathsf{decision}_a$.
- $\mathscr{S}$ then triggers in turn the auditors and computes the random challenges from the responses by taking the XOR over the responses. We assume that at least one auditor is honest, see below for a discussion of this issue.
- Then the machine M is triggered with these random challenges. The machine is supposed to open the commitments corresponding to the challenges, i.e. sending the respective information to $\mathscr{B}$. $\mathscr{S}$ triggers the judge who checks if this ZK-proof is done correctly, and, in case it is not, outputs $\mathsf{dis}(M)$.
- Otherwise, $\mathscr{S}$ starts the voting phase by triggering the first voter v_1, chosen randomly among all voters. Note that $\mathscr{S}$ could also trigger the voters in a fixed order, or even trigger the voters in an order determined by some other party, without changing the result (if every voter gets the possibility to vote).
- After it received a $\texttt{done}(v_1)$-message from the booth $\mathscr{B}$ (see below), it triggers the next voter and so on.
- If in this phase, at some point no message is sent the scheduler (as it is the master) is triggered. In this case the scheduler asks the booth $\mathscr{B}$ which participant misbehaved (see below) and forwards this to the judge who outputs the corresponding blaming on his $\mathsf{decision}$-tape.
- After all voters have been triggered, $\mathscr{S}$ starts the tallying phase by triggering the voting machine M, who is supposed to open the unused commitments by sending the respective information to $\mathscr{B}$. Then $\mathscr{S}$ triggers the judge who checks whether the commitments are opened correctly, and, in case this is not the case, outputs $\mathsf{dis}(M)$. Otherwise, analogously to the first ZK-proof, the auditors are triggered, $\mathscr{S}$ computes the random challenges from the responses and the machine is asked to open the respective commitments.
- After that, $\mathscr{S}$ triggers in turn every voter, who ask $\mathscr{B}$ with probability q_{rec} for the information on the Bulletin Board and check whether her receipt has been published correctly. If the judge receives a receipt from a voter, he/she checks that the receipt is correctly signed and blames in case.
- After the tallying phase, $\mathscr{S}$ triggers the judge a who checks the ZK-proofs, whether a number occurs twice or whether the published result is compatible to the number of voters that voted (the judge gets this information from $\mathscr{B}$) and behaves accordingly to the judging procedure.

The voting booth. The set of honest programs of the voting booth $\mathscr{B}$ consists of the following program:

- When it is first triggered by $\mathscr{S}$, it sets up the parameters for the cryptographic primitives, i.e. for the commitment scheme and the digital signatures. We model our security assumption A3, i.e. that it is not possible to forge a receipt, by means of digital signatures. We assume that $\mathscr{B}$ chooses the parameters of a digital signature scheme in a way that the probability that a polynomially bounded algorithm can forge a signature is negligible. As $\mathscr{B}$ serves as the Bulletin Board, every participant may request these parameters. In order that $\mathscr{B}$ does not get exhausted, every participant can only once demand these values.
- $\mathscr{B}$ also serves as Bulletin board for the messages that the machine has to publish, e.g. the commitments in the initialization phase. Every participant may once demand these values. $\mathscr{B}$ also reports to $\mathscr{S}$ if messages to be published are not correct, i.e. if the number of commitments does not match or the commitments do not belong to the space of commitments specified by the setup parameters.
- It accepts exactly one `enter`-message from every eligible voter v_i, modelling that every eligible voter may only enter once the voting booth. $\mathscr{B}$ counts correctly the voters, reflecting the security assumption A4. Every participant may once demand the total number of voters after the tallying phase.
- After $\mathscr{B}$ received a `enter`-message, she triggers the RNG who is supposed to answer with a fresh random number.
- After that $\mathscr{B}$ triggers the voter who is supposed to answer with a choice for a candidate.
- Then the voting machine M is triggered by $\mathscr{B}$ by sending the choice and the random number to it. The machine is supposed to answer with the receipt (specified by the protocol).
- If $\mathscr{B}$ does not receive messages of the expected type she reports this to the judge, who blames the respective participant.
- If $\mathscr{B}$ does not receive a message at all (from the voter, the RNG or the machine) the scheduler is activated who then asks $\mathscr{B}$ for the misbehaving party. $\mathscr{B}$ answers correctly to that request.
- If everybody behaves correctly, $\mathscr{B}$ sends the entire receipt to the voter together with a digital signature. The voter then may send a `complain`-message to the booth.
- $\mathscr{B}$ forwards this complain to the judge a (if we consider $\mathsf{P}^a_{\mathsf{Bingo1}}$) who states $\mathsf{dis}(M) \vee \mathsf{dis}(RNG) \vee \mathsf{dis}(v_i)$ or ignores it (if we consider Variant II, see below).
- Finally, $\mathscr{B}$ sends a $\mathtt{done}(v_i)$-message to the scheduler.

We define the set Π_b of the programs of b to be the set of honest programs of b, i.e. $\Pi_b = \{\hat{\Pi}_b\}$, as defined above, if b is honest, i.e. for $\mathscr{S}$, $\mathscr{B}$, and the judge a. For simplicity of presentation we assume that auditor A_1 is honest. Note that our security assumption A2 only states that *one* auditor is honest. This could be directly encoded in the left hand sides of the accountability constraints (by only considering the runs in which at least one auditor is honest). However, as from the responses of the auditors, the random challenges are computed in a symmetric way (by XOR) it does not matter which auditor is honest.

The set of all possible programs is only limited by the network configuration, if b is not assumed to be honest.

By these specifications we guarantee that every honest participant has enough ressources in order to follow the protocol. However, the entire system remains polynomially bounded.

A.3 Proof of Theorem 1

Fairness. By the definition of the judging procedure, and the honesty of $\mathscr{B}$ and S, it follows that a is fair: This is trivial if a blames some participant b (i.e. outputs the trivial formula $\mathsf{dis}(b)$) because of an obvious deviation. Further, a states $\mathsf{dis}(M) \vee \mathsf{dis}(RNG) \vee \mathsf{dis}(v)$ iff v complains in the voting booth. In this case, either the machine or the RNG indeed cheated or the voter is dishonest, because she complained for no reason. Further, if a states $\mathsf{dis}(M) \vee \mathsf{dis}(RNG)$, then a number occurs on two different receipts, which only happens with negligible probability if both are honest. For the same reason, the probability that a states $\mathsf{dis}(M)$ because of a number occuring twice is negligible if M is honest.

Completeness. Let π be an instance of $\mathsf{P}^a_{\mathsf{Bingo1}}$. For a set of runs α, with $\Pr[\pi(1^\ell) \mapsto \alpha]$ we denote the probability that $\pi(1^\ell)$ produces a run in α. Then we have

$$\Pr[\pi(1^\ell) \mapsto \alpha_{compl},\ \neg(a : \mathsf{dis}(M) \vee \mathsf{dis}(RNG) \vee \mathsf{dis}(v_1)), \ldots$$
$$\ldots, \neg(a : \mathsf{dis}(M) \vee \mathsf{dis}(RNG) \vee \mathsf{dis}(v_n))] = 0$$

and

$$\Pr[\pi(1^\ell) \mapsto \alpha_{twice},\ \neg(a : \mathsf{dis}(M) \vee \mathsf{dis}(RNG))] = 0,$$

as by the definition of the judging procedure, a states the respective formulas in case of the respective events. Hence it remains to show, that $\Pr[X]$ is $\delta^k_{\mathsf{Bingo}}$-bounded, where X is the event that a run of $\pi(1^\ell)$ does not belong to γ_k, α_{compl}, α_{twice} and that a does not state $\mathsf{dis}(M)$ nor $\mathsf{dis}(RNG)$.

As neither the machine nor the RNG is blamed in a run belonging to X, we have in particular that these two parties do not deviate from the protocol in an obvious way, e.g., by not producing numbers at all. Recall that in this case the booth $\mathscr{B}$ would inform $\mathscr{S}$ about the misbehavior, and $\mathscr{S}$ would inform then the (honest) judge a about this, who would output the respective verdict on his tape $\mathsf{decision}_a$.

We distinguish whether or not the machine tries to fake a zero-knowledge proof, i.e. if the machine does not produce the same number of commitments for every candidate or writes different commitments next to the receipts in the second ZK-proof. Let F denote the event that the machine tries to fake some zero-knowledge proof. Then we have

$$\begin{aligned}\Pr[X] &= \Pr[X \mid F] \cdot \Pr[F] + \Pr[X \mid \overline{F}] \cdot \Pr[\overline{F}] \\ &\leq \max(\Pr[X \mid F], \Pr[X \mid \overline{F}]).\end{aligned}$$

To complete the proof it is enough to show that

$$\Pr[X \mid F] \leq \frac{1}{2^s} + f(\ell) \qquad (12)$$

and

$$\Pr[X \mid \overline{F}] \leq \max((1 - q_{num}), (1 - q_{rec}), \max_{j=1,\ldots,l} p_j)^{k+1} + f(\ell) \qquad (13)$$

for some negligible function f.

First, let us prove inequality (12). So, assume that F holds, i.e. the machine fakes some zero-knowledge proof. X means, among others, that a does not state $\mathsf{dis}(M)$. As we assume that $\mathscr{B}$ picks the parameters for the commitment scheme honestly and as the runtime of system is polynomially bounded, under the DLOG-assumption, the probability that the machine opens a commitment on one value to another value is negligible. Further, if the machine tries to cheat in the shuffling and re-randomization phase of a zero-knowledge proof by pretending that it can open a commitment to another value, this is detected with probability $\frac{1}{2^s}$ as the challenges are really random. Hence, as can be shown by a reduction proof, a does not state $\mathsf{dis}(M)$ only with probability $\frac{1}{2^s} + f(\ell)$ for some negligible function f.

Now, we will prove (13). So, consider the probability X given $\overline{F}$, i.e. the machine does not fake a zero-knowledge proof. $\overline{F}$ implies that the machine produces the same number of commitments for every candidate. Further, we get the following claims:

CLAIM 1. *Assume that $\overline{F}$ and X hold. Then with overwhelming probability, every receipt published on the bulletin board is* well formed *in the following sense: Let $x_1, \dots, x_l$ be the numbers printed on a receipt next to candidates $1, \dots, l$, respectively. For $(l-1)$ elements of $\{x_1, \dots, x_l\}$, a commitment on (i, x_i) has been published on the bulletin board in the initialization phase and no commitment on the only remaining number x has been posted on the bulletin board.*

Otherwise, as the machine has to assign $l-1$ commitments from the bulletin board to that receipt and open them (after shuffling and masking, what is done correctly due to $\overline{F}$), a would state $\mathsf{dis}(M)$, what conflicts with X. Further suppose that a commitment on the only remaining number has been posted on the bulletin board. Then, this commitment cannot be assigned to the considered receipt (as this would mean that the machine assigned l previously published commitments to one receipts which would imply that a states $\mathsf{dis}(M)$ what contradicts X). If it were assigned to some other receipt, then, by $\overline{F}$, x would be also printed on this receipt, which would contradict α_{twice} (and X). Finally, if it were not assigned to any receipt, then it would be opened (with overwhelming probability) and a would state $\mathsf{dis}(M)$ (Step J4), which, again, contradicts X.

CLAIM 2. *Assume that $\overline{F}$ and X hold. Then the probability that M posts commitments on (a, R) and (b, R) (that share the same number R) for candidates $a \neq b$ on the bulletin board in the initialization phase is negligible.*

Otherwise, as the machine eventually opens correctly every commitment (possibly after masking and shuffling, what is done correctly due to $\overline{F}$), R would occur twice with overwhelming probability, which conflicts with X (either we have α_{twice} or a states $\mathsf{dis}(M)$).

Now, Claim 1 implies the following. If, for some voter who choses candidate i, (i) the number x_i printed next to the chosen candidate is the number provided by the RNG and (ii) no commitment to this number was posted on the bulletin board in the initialization phase, then the machine produces a receipt which corresponds to a vote for candidate i (i.e. the machine assigns exactly one commitment (that has not been assigned to a receipt so far) for each candidate but i to this receipt).

Hence, if the machine changes a vote of an honest voter, then one of the following cases must hold: (a) the receipt handed out to the voter does not match the receipt that is published on the Bulletin Board, (b) this receipt matches but condition (i) is violated, or (c) this receipt matches and condition (ii) is violated. Case (a) can happen undetectedly, only if the voter does not check whether her receipt appears on the bulletin board, which has probability $(1 - q_{rec})$. Case (b) can only happen undetectedly, if the voter does not check her receipt in the voting booth, which has probability $(1 - q_{num})$. Finally, case (c) holds, by the well-formedness of ballots, when it happens that the candidate j in the commitment $\mathsf{comm}(j, x_i)$ on the number x_i produced by the RNG coincides with the candidate chosen by the voter, that is if $j = i$. As the machine does not know in the initialization phase which candidate the voter will choose, and the RNG cannot learn the voters choice (assumption A1), this happens only with probability $\leq \max_{j=1,\dots,l} p_j$. Note that, by Claim 2, the candidate j is uniquely determined by x_i.

Summing up, the probability that the machine changes undetectedly the vote of a (fixed) honest voter, given $\overline{F}$, is bounded by $\max((1 - q_{num}), (1 - q_{rec}), \max_{j=1,\dots,l} p_j)$ and some negligible function. Hence, the probability that, given $\overline{F}$, the machine changes undetectedly the votes of $k+1$ honest voters is smaller than $\max((1 - q_{num}), (1 - q_{rec}), \max_{j=1,\dots,l} p_j)^{k+1} + f(\ell)$. As by our assumption A4, the machine cannot vote on behalf of abstaining voters, the goal γ_k is not achieved only if there are $k+1$ changed votes, we get (13), which completes the proof.

A.4 Variant II

In this section, we discuss Variant II mentioned in Section 5, i.e. the variant where the judge ignores that a voter complained in the voting booth and the voting process simply continues.

For this purpose we will consider in this section a version of the system where the judge does not blame anybody if a voter complains in the voting booth.

Clearly, in this case, the machine can change the votes of the voters by just ignoring the number transmitted from the RNG: an honest voter will complain in that case, but the election will not be aborted. For this variant of the system, we will therefore further weaken the goal γ_k obtaining a goal γ'_k which is already achieved in a run if the result produced by the machine is correct up to votes of dishonest voters, up to *votes of voters complaining in the voting booth*, and up to k votes of honest voters who do not complain in the booth.

We denote by $\mathsf{P}^a_{\mathsf{Bingo2}}$ this variant of the protocol, defined as $\mathsf{P}^a_{\mathsf{Bingo1}}$ with the difference that J2 is omitted (for reasons already discussed).

Accountability of $\mathsf{P}^a_{\mathsf{Bingo2}}$ As already stated in Section 5, a severe problem with Theorem 1 is that in case a voter complains, it is not clear who to blame specifically, the authority (which M and the RNG are part of) or a voter.

With the new judging procedure, this problem disappears, but at the price of a weaker goal. More specifically, we have the following accountability property:

$$\Phi_2 = \{\alpha_{twice} \Rightarrow \mathsf{dis}(M) \vee \mathsf{dis}(RNG),$$
$$\neg\gamma'_k \cap \neg\alpha_{twice} \Rightarrow \mathsf{dis}(M) \mid \mathsf{dis}(RNG)\}.$$

For this, we obtain the following theorem:

THEOREM 5. *Let a be an external judge or a voter. The agent a ensures $(\Phi_2, \delta^k_{\mathsf{Bingo}})$-accountability for $\mathsf{P}^a_{\mathsf{Bingo2}}(n, q_{num}, q_{rec}, s, \vec{p})$.*

PROOF. The proof of Theorem 5 is very similar to that of Theorem 1. In order to conflict γ'_k, $k+1$ votes of honest, *non-complaining* voters have to be changed. This can be done in the same ways as in Theorem 1. But now, if the machine does not use the number transmitted from the RNG at the correct place on the receipt, that does not change the vote of an honest non-complaining voter if the voter complains. Hence, also in this case, in order to change an honest, *non-complaining* voters vote (by wrongly using the number transmitted from the RNG), the machine has to hope that the voter does not check the number. Hence we are exactly in the situation of Theorem 1. □

This theorem says that, in $\mathsf{P}^a_{\mathsf{Bingo2}}$, the probability that the goal γ'_k is not achieved and still a does not blame anybody is at most

$\delta^k_{\mathsf{Bingo}}$. Since now a always (rightly) accuses *authorities*, it is easier to hold them accountable, even though not always individually. Moreover, unlike in $\mathsf{P}^a_{\mathsf{Bingo1}}$, no voter can spoil the voting process. On the downside, the goal now is weaker, and hence, the published result may deviate more from the actual votes than previously, without a blaming anybody.

Verifiability of $\mathsf{P}^a_{\mathsf{Bingo2}}$ As already discussed in Section 5, the verifiability property stated in Corollary 1 reflects the weakness of the system $\mathsf{P}^a_{\mathsf{Bingo1}}(n, q_{num}, q_{rec}, s, \vec{p})$ already revealed by Theorem 1: By wrongly complaining, every single dishonest voter can spoil the election process. This weakness is not present in the system $\mathsf{P}^a_{\mathsf{Bingo2}}$ as stated in the corollary below, which, however, comes at a price of a weaker goal:

COROLLARY 3. *Let a be an external judge or a voter. The goal γ'_k is guaranteed in $\mathsf{P}^a_{\mathsf{Bingo2}}(n, q_{num}, q_{rec}, s, \vec{p})$ by $\mathsf{hon}(M) \wedge \mathsf{hon}(RNG)$ and $\delta^k_{\mathsf{Bingo}}$-verifiable by a.*

B. THE PRST PROTOCOL

B.1 Our Improved Version

In this section, we describe in details our version of the PRST protocol.

For our protocol, we assume an *independent and honest bulletin board*, replacing the bulletin board controlled by the auctioneer: the auctioneer A and every bidder B_i can post messages on the bulletin board. The messages posted on the bulletin board appear with a time-stamp, provided by the bulletin board. We implicitly assume that the messages posted by A are signed by him; messages posted by bidders are not. For privacy of the bidders, one can assume that bidders can post messages on the bulletin board anonymously. Notaries are superfluous in our protocol.

Now, our version of the protocol consists of the following steps, where Steps S1' and S6' are exactly like the corresponding steps in the original version of the protocol:

S1'. A posts (on the bulletin board) basic information about the auction: the terms of the auction, an identifier Id, the deadlines T_1, T_2, T_3 for different stages of the auction, and his public encryption key.

S2'. Bidder B_i computes her encrypted bid $C_i = E_A(b_i, r_i)$, generates a nonce q_i, and computes her commitment as $Com_i = \langle \mathsf{hash}(C_i), \mathsf{hash}(q_i), Id \rangle$. The bidder is supposed to keep the nonce q_i secret, except for the situation described below. B_i posts (on the bulletin board)

$$Id,\ E_A(\mathsf{sig}_{B_i}[Com_i], r'_i) \tag{14}$$

before time T_1. The (hash of the) nonce q_i will be used to prevent Problem 4 (see Step S7'). Posting (14), instead of sending the signed commitment directly to A will prevent Problem 1. The signature in (14) is encrypted to hide B_i's identify from other bidders and observers. Note that B_i does not use notaries and does not send her signed commitment directly to A. Also, A is not required to send receipts.

S3'. At time T_1, the auctioneer decrypts and collects all the commitments posted in the previous step and posts these commitments in a random order:

$$Com_{\pi(1)}, \ldots, Com_{\pi(n)} \tag{15}$$

where π is a randomly chosen permutation of the indices of previously posted commitments.

If two or more commitments in this list have the same value, then the auctioneer additionally posts the list of bidder's signatures on all these commitments. The bidder whose signature is at the first position on this list is *challenged*: she is supposed to open her commitment before time T_2 (see the next step). This procedure enables the judge to resolve the conflict described in Problem 2.

S4'. Between time T_1 and T_2 any bidder B_i whose bid is not included in the list of commitments posted by A in the previous step, appeals by posting

$$Id,\ \mathsf{sig}_{B_i}[Com_i],\ r'_i. \tag{16}$$

(If the identity of B_i is to be kept secret, this message may be sent to the judge only).

Also, before time T_2, a bidder B_i who has been challenged in the previous step, opens her commitment (if she does not do it, she will be blamed; otherwise all the other bidders whose signatures are on this list will be blamed; see V2').[5]

S5'. After time T_2, every B_i opens her commitment by posting $\langle Com_i, C_i \rangle$ on the bulletin board (posting $\langle Com_i, C_i \rangle$ instead of sending C_i to A prevents Problem 2). After time T_3, A posts

$$C_{\pi(1)}, \ldots, C_{\pi(n)} \tag{17}$$

(while this step is redundant, we keep it for compliance with the original version of the protocol) and posts bidder's signature on every unopened commitment.

S6'. A recovers the bids $b_1, \ldots, b_n$, by decrypting the encrypted bids with his private decryption key, and determines the winner B_w of the auction and the price b_u the winner has to pay, which is supposed to be the second highest bid. He also constructs a (universally verifiable) zero-knowledge proof P that the result is correct, i.e. C_w contains the biggest bid and C_u contains the second biggest bid b_u: This is done by proving appropriate inequalities between the bids in the ciphertexts $C_1, \ldots, C_n$, without revealing these bids, and by revealing the random coin used in C_u, which he can recover using his private key. The auctioneer posts

$$B_w,\ b_u,\ \mathsf{sig}_{B_w}[Com_w],\ P. \tag{18}$$

(If more than one committed bid contains the highest value, then the winner/winners are determined according to some pre-agreed policy; due to space limitation, we do not consider this case further.)

S7'. A bidder $B_j \neq B_w$ checks whether the signed commitment Com_w posted by the auctioneer in (18) is her own commitment. If it is the case, she claims the fact, by posting q_j on the bulletin board, before some determined time. Note that by this B_j does not reveal her identity.

The new *judging procedure* performed by V is as follows:

V1.' If A does not publish the list of commitments when expected in Step S3', then V blames A (states $\mathsf{dis}(A)$).

V2.' If two or more commitments in (15) have the same value c and A does not post signatures on these commitments as required in S3', then A is blamed. If A posts such a list $\mathsf{sig}_{B_{i_1}}(c), \ldots, \mathsf{sig}_{B_{i_l}}(c)$ then the following verdicts are stated: If the commitment c is opened before time T_2 (supposedly by the challenged bidder B_{i_1}), then V states $\mathsf{dis}(B_{i_2}) \wedge \cdots \wedge \mathsf{dis}(B_{i_l})$; otherwise, V states $\mathsf{dis}(B_{i_1})$.

[5]One could extend this procedure such that, if the challenged bidder does not open the commitment, then the next bidder is challenged, and so on. By this we could guarantee higher precision of blaming.

V3.' If in Step S4', message (16) is posted such that there is a corresponding encryption in a message of the form (14) posted before time T_1 and Com_i is not included in the list of commitments posted by A in Step S3', then V blames A.

V4.' After Step S5', if the bulletin board does not contain an entry that opens some commitment Com_i, then: If A has not provided the required signature in Step S5', then V blames him. Otherwise, the party identified by this signature is blamed. Furthermore, if A does not post (17), then A is blamed.

V5.' If, in Step S6', A does not post a result with a valid zero-knowledge proof and a valid signature $\mathsf{sig}_w[Com_w]$, then V blames A.

V6.' If, in Step S7', a nonce q_w is posted such that Com_w contains $\mathsf{hash}(q_w)$, then V blames B_w.

We, again, consider a few variants of the protocol: By $P^J_{PRST'}$ we denote our version of the protocol with an additional, honest party, the *judge*, who runs the judging procedure described above. By $P^X_{PRST'}$, for $X \in \{A, B_1, \dots, B_n\}$, we denote our version of the protocol, where X is assumed to be honest and his/her honest program is *extended* by the judging procedure.

B.2 Symbolic Model of Processes

In this section we instantiate the abstract notion of a protocol by a symbolic model, where atomic processes, following [31], are represented as functions that for a sequence of input messages (the messages received so far) produce output messages.

This model, unlike many other symbolic models (like π-calculus), by means of a so called master process (a scheduler), enables us to precisely model phases of the protocol so that every party is given a chance to perform the required actions in every stage of the protocol (and therefore can be fairly held accountable if he/she does not do it).

While we focus here on specific cryptographic primitives used in the PRST protocol, the model presented here is generic and can be used with a different set of cryptographic primitives.

Messages. Let Σ be some signature for cryptographic primitives (including a possibly infinite set of constants for representing participant names, etc.), $X = \{x_1, x_2, \dots\}$ be a set of variables, and Nonce be an infinite set of *nonces*, where the sets Σ, X, and Nonce are pairwise disjoint. For $N \subseteq \mathsf{Nonce}$, the set T_N of *terms* over $\Sigma \cup N$ and X is defined as usual. Ground terms, i.e., terms without variables, represent messages. The set of all messages will be denoted by Msg.

We assume some fixed equational theory associated with Σ and denote by $\equiv$ the congruence relation on terms induced by this theory. The particular signature Σ we take to model the PRST protocol, along with an appropriate equivalence theory, is given below.

Cryptographic Primitives for PRST. We use a term of the form $\langle m, m' \rangle$ to represent a pair of messages m and m'; with $\mathsf{first}(p)$ and $\mathsf{sec}(p)$ yielding, respectively, the first and the second component of a pair p. A term $\mathsf{sig}_k(m)$ represents the signature on a message m under a (private) key k. Such a signature can be verified using $\mathsf{pub}(k)$, the public key corresponding to k. A term $\mathsf{hash}(m)$ represents the result of applying the hash function to m.

We use the following terms to represent randomized encryption with homomorphic property: $\{m\}^r_k$ represents a term m encrypted under a (public) key k using a randomness r; $\mathsf{dec}(c, k)$ represents a decryption of a ciphertext c with a key k (k is intended to be a private key corresponding to the public key under which c is encrypted). The ability to extracting the random coin from a given ciphertext is expressed using the symbol $\mathsf{extractCoin}$.

$$\mathsf{checkSig}(\mathsf{sig}_m(k), \mathsf{pub}(k)) = \mathsf{T}$$

$$\mathsf{dec}(\{x\}^r_{\mathsf{pub}(k)}, k) = x$$

$$\mathsf{extractCoin}(\{x\}^r_{\mathsf{pub}(k)}, k) = r$$

$$\{m_1\}^{r_1}_k \times \{m_2\}^{r_2}_k = \{m_1 + m_2\}^{r_1+r_2}_k$$

$$\mathsf{first}(\langle x, y \rangle) = x \qquad \mathsf{sec}(\langle x, y \rangle) = y$$

$$\begin{array}{rclcrcl} x \doteq x & = & \mathsf{T} & \quad & \mathsf{T} \vee x & = & \mathsf{T} \\ \mathsf{T} \wedge \mathsf{T} & = & \mathsf{T} & & x \vee \mathsf{T} & = & \mathsf{T} \end{array}$$

Figure 1: Theory E — equational theory for modeling PRST.

To model the set of possible bids (which is finite) we introduce symbols $0, \dots, (M-1)$ along with operators $+$ and $\times$ (needed for expressing the homomorphic property of the encryption) and symbols $<$ and $\leq$. We assume full axiomatization of this finite set of numbers w.r.t. these operators.

Zero-knowledge proofs. We will model the zero-knowledge proofs used in the protocol following the approach of [7] and [32]. A zero-knowledge proof will be represented by a term $P = \mathsf{ZK}^{n,k}_{\varphi}(t_1, \dots, t_n; s_1, \dots, s_n)$ where $t_1, \dots, t_n$ are terms called *the private component* (the proof will keep these terms secret), terms $s_1, \dots, s_n$ are called *the public component* (the proof reveals these terms), and φ is a term built upon variables $x_1, \dots, x_n, y_1, \dots, y_n$ (no other variables and no nonces can occur in this term; x_i is intended to refer to t_i, while y_i is intended to refer to s_i), called *the formula of P*.

We have the following equalities associated to zero-knowledge proofs. The first group of equations reveals the public components (also the formula) of a proof. The second one allows one to check validity of a proof.

$$\mathsf{public}(\mathsf{ZK}^{n,k}_{\varphi}(t_1, \dots, t_n, s_1, \dots, s_n)) = \langle \varphi, s_1, \dots, s_n \rangle$$

$$\mathsf{check}(\mathsf{ZK}^{n,k}_{\varphi}(t_1, \dots, t_n, s_1, \dots, s_n)) = \mathsf{T}$$

if φ is a formula build upon $x_1, \dots, x_n, y_1, \dots, y_n$, and $\varphi[t_i/x_i, s_i/y_i] \equiv_E \mathsf{T}$.

To model the zero-knowledge proofs used in the protocol, we will use the expression $P_<(b_1, b_2, k; c_1, c_2)$ representing the proof that the c_1 is of the form $\{b_1\}^{r_1}_k$, for some r_1, and c_2 is of the form $\{b_2\}^{r_2}_k$, for some r_2, with $b_1 < b_2$. Formally, $P_<(b_1, b_2, k; c_1, c_2)$ stands for $\mathsf{ZK}^{1,1}_{\varphi}(b_1, b_2, k; c_1, c_2)$, where

$$\varphi = \left[\mathsf{dec}(y_1, x_3) = x_1 \wedge \mathsf{dec}(y_2, x_3) = x_2 \wedge x_1 < x_2\right].$$

Similarly, $P_\leq(b_1, b_2, k; c_1, c_2)$ representing the proof that the c_1 is of the form $\{b_1\}^{r_1}_k$ and c_2 is of the form $\{b_2\}^{r_2}_k$, with $b_1 \leq b_2$. Formally, $P_\leq(b_1, b_2, k; c_1, c_2)$ stands for $\mathsf{ZK}^{1,1}_{\varphi}(b_1, b_2, k; c_1, c_2)$, where

$$\varphi = \left[\mathsf{dec}(y_1, x_3) = x_1 \wedge \mathsf{dec}(y_2, x_3) = x_2 \wedge x_1 \leq x_2\right].$$

Runs and views. Let Ch be a set of *channels* (*channel names*). An *event* is of the form $(c : m)$, for $c \in \mathsf{Ch}$ and $m \in \mathsf{Msg}$. Such an event is meant to express that the message m is delivered on channel c. The set of all events will be denoted by Ev. A finite or infinite sequence of events is called a *run*.

For a run $\rho = (c_1 : m_1)\,(c_2 : m_2), \dots$, we denote by $\mathrm{chan}(\rho)$ the sequence $c_1, c_2, \dots$ of channels in this sequence. For $C \subseteq \mathsf{Ch}$, we denote by $\rho_{|C}$ the subsequence of ρ containing only the events $(c : m)$ with $c \in C$. Let $\tau \in T_N$ be a term. Then, with ρ as above, we denote by $\tau[\rho]$ the message $\tau[m_1/x_1, m_2/x_2, \dots]$, where x_i is replaced by m_i. (Recall that the set of variables is $X = \{x_1, x_2, \dots\}$.)

EXAMPLE 3. Assume that $\tau_{ex} = \mathsf{dec}(x_1, \mathsf{first}(x_2))$ and $\rho_{ex} = (c_1 : \{a\}^r_{\mathsf{pub}(k)}), (c_2 : \langle k, b\rangle)$. Then

$$\tau_{ex}[\rho_{ex}] = \mathsf{dec}(\{a\}^r_{\mathsf{pub}(k)}, \mathsf{first}(\langle k, b\rangle)) \equiv_{ex} a.$$

Borrowing the notion of static equivalence from [2], we call two runs ρ and ρ' *statically equivalent w.r.t. a set* $C \subseteq \mathsf{Ch}$ *of channels and a set* $N \subseteq \mathsf{Nonce}$ *of nonces*, written $\rho \equiv^C_N \rho'$, if (i) $\mathrm{chan}(\rho_{|C}) = \mathrm{chan}(\rho'_{|C})$ and (ii) for every $\tau_1, \tau_2 \in T_N$ we have that $\tau_1[\rho_{|C}] \equiv \tau_2[\rho_{|C}]$ iff $\tau_1[\rho'_{|C}] \equiv \tau_2[\rho'_{|C}]$. Intuitively, $\rho \equiv^C_N \rho'$ means that a party listening on channels C and a priori knowing the nonces in N cannot distinguish between the inputs received according to ρ and those received according to ρ'. We call the equivalence class of ρ w.r.t. $\equiv^C_N$, the (C,N)*-view* on ρ.

EXAMPLE 4. For example, if k, k', a, and b are different constants, r and r' are nonces, $C = \{c_1, c_2\}$, and $N = \emptyset$, then it is easy to see that $\rho^1_{ex} = (c_1 : \{a\}^r_{\mathsf{pub}(k)}), (c_2 : \langle k', b\rangle), (c_3 : k)$ and $\rho^2_{ex} = (c_1 : \{b\}^{r'}_{\mathsf{pub}(k)}), (c_2 : \langle k', b\rangle)$ yield the same (C,N)-view w.r.t. $\equiv_{ex}$.

Processes. Processes are built from atomic processes. An atomic process is basically a function that given a finite history (representing the messages delivered so far) returns ε (if the process does not send any message) or an element of the form $(c : \tau)$ (if the process sends some message). We require that an atomic process behaves the same on inputs on which it has the same view. Formally, atomic processes are defined as follows.

DEFINITION 7. An *atomic process* is a tuple $p = (I, O, N, f)$ where

(i) $I, O \subseteq \mathsf{Ch}$ are finite sets of *input* and *output* channels, respectively,
(ii) $N \subseteq \mathsf{Nonce}$ is a set of *nonces used by p*,
(iii) f is a mapping which assigns, to each (I,N)-view U, a responce $f(U)$ of the form ε or $(c : \tau)$ with $c \in O$ and $\tau \in T_N$.

We refer to I, O and N by I_p, O_p, and N_p, respectively. We note that the sets I_p and O_p do not have to be disjoint (which means that p can send messages to itself).

We note that (iii) guarantees that p performs the same computation on event sequences that are equivalent according to $\equiv^I_N$, and hence, on which p has the same view. This is why f is defined on (I,N)-views rather than on sequences of input events.

For a history ρ such that U is the equivalence class of ρ' w.r.t. $\equiv^I_N$, we write $p(\rho)$ for the output produced by p on input ρ. This output is ε, if $f(U) = \varepsilon$, or $(c : \tau[\rho|_I])$ if $f(U) = (c : \tau)$.

EXAMPLE 5. Let $I = \{c_1, c_2\}$, $N = \emptyset$, and U be the equivalence class of ρ^1_{ex}. Assume also that $f(U) = (c_4 : \langle x_1, \mathsf{first}(x_2)\rangle)$. Then, $p(\rho^1_{ex}) = (c_4 : \langle \{a\}^r_{\mathsf{pub}(k)}, \mathsf{first}(\langle k', b\rangle)\rangle)$, which modulo $\equiv_{ex}$ can be equivalently written as $(c_4 : \langle \{a\}^r_{\mathsf{pub}(k)}, k'\rangle)$ and $p(\rho^2_{ex}) = (c_4 : \langle \{b\}^{r'}_{\mathsf{pub}(k)}, \mathsf{first}(\langle k', b\rangle)\rangle)$, which modulo $\equiv_{ex}$ can be equivalently written as $(c_4 : \langle \{b\}^{r'}_{\mathsf{pub}(k)}, k'\rangle)$. Note that since ρ^1_{ex} and ρ^2_{ex} yield the same (I,N)-view w.r.t. $\equiv_{ex}$, p performs the same transformation on ρ^1_{ex} and ρ^2_{ex}.

For the definition of a process, given below, we assume that there is a distinct channel $\mathsf{ch}_{\mathsf{init}} \in \mathsf{Ch}$ and a distinct constant init in the signature Σ.

DEFINITION 8. A *process* is a finite set π of atomic processes with disjoint sets of input channels and sets of nonces (i.e., $I_p \cap I_{p'} = \emptyset$ and $N_p \cap N_{p'} = \emptyset$, for distinct $p, p' \in \pi$) such that there is an atomic proces $p_0 \in \pi$ with $\mathsf{ch}_{\mathsf{init}} \in I_p$ and $\mathsf{ch}_{\mathsf{init}} \notin O_p$ for all p in π. The atomic process p_0 is called the *master atomic process* of π.

Runs of processes. For a process π, we define the run of π in the following way. In each step, we have a configuration that consists of a finite run ρ (the events delivered so far) and the current event e (the event to be delivered in the next step). We start with the initial configuration with the empty run ρ and $e = \varepsilon$. In each step we extend the run and compute the next current event, obtaining the new configuration (ρ', e'), as follows. By definition of processes, there exists at most one atomic process, say p, in π with an input channel corresponding to e (if $e \neq \varepsilon$). If such a process p exists (which means that the current event can be delivered to p), then we obtain the next configuration, by taking $\rho' = \rho e$ and $e' = p(\rho')$. If such a process does not exists—which can happen if there is no message to be delivered ($e = \varepsilon$) or there is no atomic process with the appropriate input channel—then we trigger the master atomic process p_0 by sending to it the init event: we take $\rho' = \rho(\mathsf{ch}_{\mathsf{init}} : \mathsf{init})$ and $e' = p_0(\rho')$. Note that, in the first step of a run of π, the master atomic process is always triggered. Now, the *run of* π is an infinite run induced by the sequence of finite runs in the consecutive configurations, as defined above.

We will consider only *fair* runs, where the master atomic process is triggered infinitely often (which means that no regular processes can "hijack" the run by performing some infinite computations).

B.3 Modeling the PRST System

In this section we provide the formal description of the PRST protocol, based on the model described above. We give, however, only the modeling of our variant of this system; the original variant can be modeled in an analogously way. Also, since in our variant of the protocol, the security properties we prove do not depend on the use of notaries, we skip these parties in the modelling. Moreover, for the simplicity of presentation, we consider only the case with an external judge (the result easily carries out to the case when one of the bidders plays the role of the verifier).

We define $P^J_{PRST'}$ as the system $(\Sigma, \mathsf{Ch}, \mathrm{In}, \mathrm{Out}, \{\Pi_a\}_{a\in\Sigma}, \{\hat{\Pi}_a\}_{a\in\Sigma})$ with the components defined below. We assign to every party $a \in \Sigma$ an infinite set N_a of nonces a can use.

Participants. The set Σ of the protocol participants consists of $B_1, \dots, B_n$ (the bidders), A (the auctioneer), BB (the bulletin board), the judge J, and, additionally, *the scheduler S* and the *key distribution center* (KDC). The role of the scheduler (who formally will be the master atomic process; see Section B.2) is to make sure that every party gets a chance to perform some actions in every protocol phase. It is important, because otherwise we could not blame parties who did not perform required actions (for instance, did not open their commitments, as required, in the appropriate protocol stage). The role of KDC is to generate and distribute private and public keys of the participants.

Recall that the judge and the bulletin board are assumed to be honest. We also assume that the scheduler, and KDC are honest. The remaining parties are not assumed to be honest.

Channels. The set of channels we consider here consists of channels ch^a_b, for every $a, b \in \Sigma$, the channel $\mathsf{ch}_{\mathsf{init}}$, and decision chan-

time	expected action
t_1	A performs S1'
t_2	every B_i performs S2'
$t_3\ (= T_1)$	A performs S3'
t_4	every B_i performs S4'
$t_5\ (= T_2)$	every B_i performs (the first part of) $S5'$
$t_6\ (= T_3)$	A performs (the second part of) S5'
t_7	A performs S6'
t_8	every B_i performs S7'
t_9	the judge performs V2'–V5'

Figure 2: The expected response of the protocol participants to the consecutive time messages

nels $\mathsf{decision}_a$, for every $a \in \Sigma$. The channel ch^a_b is an output channel of a and an input channel of b. So, a and b can communicate using ch^a_b and ch^b_a. For $a \in \Sigma$, we define $\mathrm{In}(a) = \{\mathsf{ch}^{a'}_a : a' \in \Sigma\}$ and $\mathrm{Out}(a) = \{\mathsf{ch}^a_{a'} : a' \in \Sigma\} \cup \{\mathsf{decision}_a\}$. For the scheduler, we additionally assume that $\mathsf{ch}_{\mathsf{init}}$ is in $\mathrm{In}(S)$ (therefore, the scheduler is running master atomic processes).

Sets of programs Π_a. Depending on whether a party $a \in \Sigma$ under consideration is assumed to be honest (the judge, the bulletin board, the scheduler, and the key distribution center) or, possibly, dishonest (the auctioneer and the bidders) we define the set Π_a of the programs of a to be: (1) the set of honest programs of a, i.e. $\Pi_a = \hat{\Pi}_a$, as defined below, if a is honest, or (2) the set of all possible programs only limited by the network configuration, i.e. $\Pi_a = \mathsf{Proc}(\mathrm{In}(a), \mathrm{Out}(a), N_a)$, if a is not assumed to be honest.

Now, we define the sets of honest programs of the participants. We provide here only high-level description of these programs. It is, however, straightforward (although tedious) to encode these programs formally as atomic processes, as defined in the previous section.

The scheduler: As we have mentioned, $\mathsf{ch}_{\mathsf{init}}$ is an input channel of S and, hence, in every instance of the protocol, the processes of the scheduler is the master (atomic) process. The role of the scheduler, as we have also mentioned, is to to trigger every party, so that it is given a chance to perform the actions required by the protocol.

The set of honest programs $\hat{\Pi}_S$ of the scheduler contains all the processes π defined as follows. A process $\pi \in \hat{\Pi}_S$ sends two kind of messages to protocol participants (using the channels ch^S_a, for $a \in \Sigma$): the message $\mathsf{trigger}$ and messages $t_0, t_1, t_2, \ldots$ (time messages). Exactly one message is sent every time π is invoked by delivering $(\mathsf{ch}_{\mathsf{init}} : \mathsf{init})$, which, by fairness, happens an infinite number of times. The order of messages sent to different participants is not restricted (different processes in $\hat{\Pi}_S$ have different order corresponding to different scheduling), except for the following requirement: (1) for every i, the message t_i is sent exactly once to every protocol participant and (2) if $i < j$, then t_i is sent before t_j.

By the above, it is guaranteed that every party is triggered in every stage i, using the message t_i and, apart from this, he/she can be triggered an arbitrary number of times by the message $\mathsf{trigger}$.

The expected response of the protocol participants to the consecutive time messages is, informally, summarized in Figure 2.

The bulletin board: The set of programs of the bulleting board consists of only one program which collects all the received messages and, on request, provides the list of these messages to every party.

The key distribution center: The set of programs of the bulleting board consists of only one program which, in response to $\mathsf{request}$ sent by a participant a, sends back to a the message

$$keys_a = \langle k_a, \mathrm{pub}(k_a), \mathrm{pub}(k_A), \mathrm{pub}(k_{B_1}), \ldots, \mathrm{pub}(k_{B_n})\rangle,$$

where k_b, for $b \in \Sigma$ the private key of b generated by KDC (formally, it is a distinct nonce of KDC). Note that $\mathrm{pub}(k_b)$ represents the corresponding public key (see Figure 1). Therefore, the response of KDC contains the private key of a and the public keys of the auctioneer and all bidders.

The bidders: The set of honest programs Π_{B_i} of a bidder B_i consists of two kinds of programs: (1) the empty program that ignores all the incoming messages (representing the case when the bidder abstains from bidding); (2) for each bid value b, the program π^b_i representing the case when the bidder, following the protocol, bids b. This program in response to time messages t_2, t_4, t_5 and t_8 performs steps S2', S4', S5' (the first part), and S7', respectively (other messages are left without any reply).

We describe in more details only the response of π^b_i to the message t_2 and t_4 sent by the scheduler; the remaining steps can be implemented similarly.

When we say that t_2 is delivered to π^b_i, then, formally, the following happens: the atomic process π^b_i (or, more precisely, the function of this process) is given a run ρ representing the history of the system so far, with the last event $(\mathsf{ch}^S_{B_i} : t_2)$. Now, we define the response to ρ as follows: The process sends $\mathsf{request}$ to the KDC, i.e. it returns $(\mathsf{ch}^{B_i}_{KDC} : \mathsf{request})$. The key distribution center, by the definition of its program, sends back to B_i the message $keys_{B_i}$. Therefore, π^b_i is invoked again, this time with ρ extended by $(\mathsf{ch}^{KDC}_{B_i} : keys_{B_i})$. Now, the process sends the request $(\mathsf{ch}^{B_i}_{BB} : \mathsf{request})$ to the bulletin board who, by the definition, responds immediately, providing the complete list of messages posted so far. This lists should contain the identifier Id of the auction signed by the auctioneer (if it is not the case, π^b_i halts). Now, the process posts (14) on the bulleting board, i.e. sends this message on $\mathsf{ch}^{B_i}_{BB}$. To construct (14) the process uses its own nonces, the retrieved keys, and Id obtained in the previous step.

Similarly, in responce to t_4, the process π^b_i sends the request $(\mathsf{ch}^{B_i}_{BB} : \mathsf{request})$ to the bulletin board who, by the definition, responds immediately, providing the complete list of messages posted so far. This list should contain, in particular, the list of commitments signed and posted by the auctioneer (if it is not the case, the process halts). Now, if the commitment of B_i (sent previously) is not in this list, π^b_i appeals, by sending (16) to the bulleting board. Further, if B_i is challenged (i.e. the bulletin board contains a list posted by A of signatures on the same commitment, with the signature of B_i on the first position on this list), then the process posts his encrypted bid C_i. Otherwise, the process does not produce any output.

The auctioneer: The set of honest programs of the auctioneer consists of only one program which in response to messages t_1, t_3, t_6, and t_7 performs Steps S1', S3', S5' (the second part), and S6', respectively. Additionally, if requested by the judge in step V3' (see below), it sends back to the judge the required signature.

The judge: The set of honest program of the judge consists of only one program which: (1) obtains the public keys of all parties (when triggered for the first time); (2) in every protocol step t_1–t_9, retrieves the content of the bulletin board; (3) in response to message t_9 (retrieves the complete content of the bulleting board and), performs steps V1'–V6'. Note that, at this point, the judge has a complete view on the history of the system recorded on the bulletin board.

B.4 Proof of Theorem 3

Before we prove Theorem 3, we provide a formal specification of the goal γ, sketched in Section 6.2. The property γ is satisfied if and only if the following conditions hold:

(a) A posts a list (15) of commitments which includes the commitments of the honest bidders and where all commitments are different.

(b) All the published commitments are correctly opened and the sequence $C_{\pi(1)},\dots,C_{\pi(n)}$ of the corresponding encrypted bids is posted by A in (17).

(c) A publishes a result, as in (18).

(d) The published price b_u is the second highest bid amongst the bids encrypted in $C_1,\dots,C_n$.

(e) An honest bider is declared to be the winner if and only if her bid is the highest in $C_1,\dots,C_n$.

Now we are ready to prove the theorem.

Fairness: To show that V is fair in $P = P^V_{PRST'}$, let π be an instance of P and r be a run of π such that V states a verdict ψ in r. We need to show that $\pi \models \psi$. Since V states a verdict, by definition of the honest program of V, one of the cases given by V1'–V6' must hold.

We will present here only proofs for two most interesting cases: V2' and V6'. The proofs for the remaining cases are quite straightforward.

Case V2': Suppose that the case described in V2' holds. Without loss of generality we can assume that the case described in V1' does not hold (we consider this case separately). It means that A posts the list (15) of commitments and two or more commitments on this list have the same value c. We have to consider two sub-cases.

First, suppose that A does not post signatures on these commitments as required in S3'. In this case V states $\psi = \mathsf{dis}(A)$. By the definition of the honest program of the auctioneer, A cannot be honest in π (the honest program of A always posts the required signatures, as described in S3'). Therefore the verdict ψ is true in π.

Now, suppose that A posts a list $\mathsf{sig}_{B_{i_1}}(c),\dots,\mathsf{sig}_{B_{i_l}}(c)$ of required signatures. We consider two cases depending on whether or not the commitment c is opened before time T_2.

If the commitment c is not opened before T_2, then the verdict is $\psi = \mathsf{dis}(B_{i_1})$. We need to show that B_{i_1} is not honest in π. Suppose that it is not the case, but, on the contrary, B_{i_1} is honest in π. As, by the definitions of honest programs of a bidder and the honest program of the key issuer, the private key of B_{i_1} is never revealed to other parties. Therefore, by the equational theory given in Figure 1, it must have been B_{i_1} who produced $\mathsf{sig}_{B_{i_1}}(c)$ and, thus, by the definition of her honest programs, c is the commitment of B_{i_1} and she is able to open it, which she does before time T_2, since she is challenged. This contradict the assumption that c is not opened.

If the commitment c is opened before time T_2 then V states $\psi = \mathsf{dis}(B_{i_2}) \wedge \dots \wedge \mathsf{dis}(B_{i_l})$. We need to show that, for all $k \in \{2,\dots,l\}$, the bidder B_{i_k} is not honest in π. For the sake of contradiction, suppose that B_{i_k} is honest in π. Then, as previously, one can show that the signature $\mathsf{sig}_{B_{i_1}}(c)$ must have been produced by B_{i_k} and that c is the commitment of B_{i_k}. But then, as B_{i_k} is not challenged, she does not open this commitment before time T_2. Also, nobody else is able to open this commitment before this time, by the equational theory under consideration (see Figure 1).

Case V6': Suppose that none of V1'–V5' holds and that V6' holds, which means that A posts a result (18) with a valid zero-knowledge proof and a valid signature $\mathsf{sig}_w[Com_w]$ (this is because the case described in V5' does not hold) and, moreover, a value q_w is posted on the bulletin board such that Com_w is of the form $\langle \mathsf{hash}(C_w), \mathsf{hash}(q_w), Id\rangle$. Therefore, V states the verdict $\psi = \mathsf{dis}(B_w)$. To prove that ψ is true in π, we need to show that B_w is not honest in π (i.e. the process of B_w used in π is not in $\hat{\Pi}(B_i)$).

To do so, let us suppose, that this is not the case, but, on the contrary, B_w runs some of his honest programs. As, by the definitions of honest program of a bidder and the honest program of the key issuer, the private key of B_w is never revealed to other parties. Therefore, it must have been B_i who has signed the commitment Com_w and, thus, B_i must have produced this commitment. However, in such a case, again by the definition of honest programs of bidders, B_w never reveals the nonce q_w used in Com_w and, in particular, does not post this value. Since the only term containing q_w, known to other parties, is $\mathsf{hash}(q_w)$ and the symbol $\mathsf{hash}(\cdot)$ is free in the equational theory under consideration (see Figure 1), no other party is able to derive q_w and post it, which contradicts the assumption that q_w is posted.

Completeness: To show that the only constraint of Φ' is ensured by V in every run of $P_{PRST'}$, let us suppose that γ does not hold in some run r of $P_{PRST'}$. It means that one of the conditions (a)–(e) is violated in r. In each of these cases we need to prove that an individual party is blamed by V in r, i.e. V states in r a verdict which implies (at least) one of $\mathsf{dis}(A), \mathsf{dis}(B_1), \cdots, \mathsf{dis}(B_n)$.

Condition (a) is violated, i.e. A does not post a list (15) of commitments which includes the commitments of the honest bidders. If A does not posts this list at all, then A is blamed (i.e. V states the verdict $\mathsf{dis}(A)$), by V1'. If A posts such a list, but the commitment of some *honest* bidder B_i is not included, then, by the definition of honest programs of bidders, B_i appeals by posting (16) and, in consequence, V blames A, by V3'. If, as previously, A posts such a list, but two or more commitments in this list have the same value c, then, by V2', V either states $\mathsf{dis}(A)$ (if A does not provides the required list of signatures) or $\mathsf{dis}(B_{i_2}) \wedge \dots \wedge \mathsf{dis}(B_{i_l})$ (if A posts such a list $\mathsf{sig}_{B_{i_1}}(c),\dots,\mathsf{sig}_{B_{i_l}}(c)$). Note that in both cases individual parties are blamed. In particular, in the latter case, the verdict implies, for instance, $\mathsf{dis}(B_{i_2})$.

Condition (b) is violated: If some commitment is not opened, then, by V4', V blames either A some bidder B_i. If the list (17) is not posted by A, then A is blamed, also by V4'.

Condition (c) is violated, i.e. A does not publish any result (18). In this case V blames A, by V5'.

Condition (d) is violated, i.e. the result (18) is published, but the published price b_u is not the second highest bid amongst the bids encrypted in $C_1,\dots,C_n$. In this case, by the equational theory under consideration, A is not able to construct a valid zero-knowledge proof, as required in S6' and, therefore, V blames A, by V5'.

Condition (e) is violated: We can assume that condition (b) is satisfied (we have considered a violation of this condition above).

If an honest bidder is declared as the winner, then her signature on the commitment Com_w corresponding to the winning encrypted bid C_w is posted in (18). By the definition of her honest programs, the definition of the honest program of the key distribution center, and by the equational theory we consider, this is possible only if this honest bidder in fact has produced C_w. Now, if the bidder declared as the winner did not bid the highest value, i.e. C_w does not contain the highest value, then, by the equational theory under consideration, A would not be able construct the required valid zero-knowledge proof P. Therefore V would blame A by V5'.

Now, suppose that and honest bidder B_j who has bid the highest value is not declared as the winner (i.e. $j \neq w$). Then, since B_j has bid the highes value and her encrypted bid C_j is in (17) (recall our assumption that (b) is not violated), by the correctness of the zero-knowledge proof P, $C_w = C_j$ and, hence, she knows and posts the nonce $q_j = q_w$. In consequence, V blames B_w. $\square$

B.5 Proof of Theorem 2

We use the detailed specification of the goal γ given in Appendix B.4.

Fairness: To show that V is fair in $P = P^V_{PRST}$, let π be an instance of P and r be a run of π such that V states a verdict ψ in r. We need to show that $\pi \models \psi$. Since V states a verdict, by definition of the honest program of V, one of the cases given by V1–V6 must hold. We will present here only a proof for the case, where V states a verdict of the form $\psi^*_{\{A,B_{i_1},\dots,B_{i_l}\}}$, as described in Step V2. This is is the most interesting case; the proofs for the remaining cases are quite straightforward.

So, suppose that A posts the list of commitments such that l commitments on this list, for $l > 1$, have the same value c and A provides the signatures $\mathsf{sig}_{B_{i_1}}(c), \dots, \mathsf{sig}_{B_{i_l}}(c)$ as required in V2. Hence, by the definition of the judging procedure, V states $\psi = \psi^*_{\{A,B_{i_1},\dots,B_{i_l}\}}$. To prove that ψ is true in π, it is enough to show that if one of $A, B_{i_1}, \dots, B_{i_l}$ is honest, then the remaining ones are not honest in π.

First, suppose that A is honest in π. Then, A does not reveal the value of c before time T_1. Therefore, every bidder in $\{B_{i_1}, \dots, B_{i_l}\}$ must have either revealed her own commitment or submitted somebody else's commitment. In both cases the bidder is not honest.

Now, suppose that one of the bidders, say B_{i_1} is honest. Because she has signed c, it must be her own commitment. Since she did not reveal her commitment c, except for sending it directly to A, the auctioneer must have revealed c to the other bidders, letting them sign c, before publishing the list of commitments. Therefore, A is not honest in π. Also $B_{i_2}, \dots, B_{i_k}$ are not honest in π, as they have signed not their own commitments.

Completeness: We need to show that V ensures all the constraints in Φ in every run r of P_{PRST}:

Constraint (7). If α^i_{rec} holds (i.e. $r \in \alpha^i_{rec}$), then V states $\mathsf{dis}(A) \vee \mathsf{dis}(B_i)$, by V1.

Constraint (8). If α^i_{open} holds, then V states $\mathsf{dis}(A) \vee \mathsf{dis}(B_i)$ by V4.

Constraint (9). If α^X_{reuse} holds, then V states $\psi^*_{X \cup \{A\}}$, by V2.

Constraint (10). If $\alpha^{w,j}_{win}$ holds, then V states $\psi^*_{\{A,B_w,B_j\}}$, by V6.

Constraint (11). Suppose that neither of α^i_{rec}, α^i_{open}, α^X_{reuse}, $\alpha^{w,j}_{win}$ holds, and that $\neg\gamma$ holds. It means that one of the conditions (a)–(e) is violated in r. In each of these cases we to prove that A is individually blamed:

Condition (a) is violated: If A does not post a list (15) of commitments at all, then A is blamed, by V2. If A posts this list, but the commitment of some honest bidder B_i is not included, then this bidder provides her receipt (we know that she obtained her receipt, because otherwise, she would claim, which would mean that α^i_{rec} holds) and A is blamed, by V3. We also know that commitments on this list are pairwise different, because, as we assumed, α^X_{reuse} does not hold, for any X.

Condition (b) is violated: If some commitment is not opened (in particular, if the list is not posted at all), then it must hold that A does not provide any signature on any commitment (otherwise α_{open} would hold). In this case, A is blamed by V4.

Condition (c) is violated, i.e. A does not publish any result (18). In this case V blames A, by V5.

Condition (d) is violated, i.e. the result (18) is published, but the published price b_u is not the second highest bid amongst the bids encrypted in $C_1, \dots, C_n$. In this case, by the equational theory under consideration, A is not able to construct a valid zero-knowledge proof, as required in S6' and, therefore, V blames A, by V5.

Condition (e) is violated: We can assume that condition (b) is satisfied (we have considered a violation of this condition above).

If an honest bidder is declared as the winner, then her signature on the commitment Com_w corresponding to the winning encrypted bid C_w is posted in (18). This is possible only if this honest bidder in fact has produced C_w. Now, if the bidder declared as the winner did not bid the highest value, i.e. C_w does not contain the highest value, then, by the equational theory under consideration, A would not be able construct the required valid zero-knowledge proof P. Therefore V would blame A by V5.

Now, suppose that and honest bidder B_j, who has bid the highest value, is not declared as the winner (i.e. $j \neq w$). One can show that this case cannot hold, as it contradicts our assumption that $\alpha^{w,j}_{win}$ does not hold. $\square$

C. PROOF OF PROPOSITION 1

In this section we prove Proposition 1.

C.1 Symbolic Setting

Assume that (4) **holds.** That means that

(a) J is fair, i.e. if J states ψ in a run r of an instance π of P then $\pi \models \psi$ and

(b) J ensures $(\neg\gamma \Rightarrow \overline{\varphi})$, i.e. for every run r we have $r \in \gamma$ or J states a formula that implies $\overline{\varphi}$.

We now show that the two conditions of Definition 4 hold.

1. Let r be a run of an instance π of P such that $\pi \models \varphi$. We have to show that J accepts r.

   If J does not accept r, then, by definition, J outputs a formula ψ that, by assumption, implies $\overline{\varphi}$. From (a) we get that $\pi \models \psi$. As ψ implies $\overline{\varphi}$, this implies $\pi \models \overline{\varphi}$. That contradicts $\pi \models \varphi$. Hence we get the first condition of verifiability of (5).

2. Let r be an arbitrary run of an instance π of P in which J accepts r. We have to show that $r \in \gamma$.

   As J accepts r, J does not state a formula ψ (that implies $\overline{\varphi}$). By (b) we have that $r \in \gamma$, what is the second condition of verifiability.

Now we prove that (5) implies (4) under the condition that J, if it states a formula, he states $\overline{\varphi}$.

Assume that (5) **holds.** That means that

(c) For every run r of an instance π of P such that $\pi \models \varphi$, J accepts r.

(d) For every run r in which J accepts r we have $r \in \gamma$.

We have to show that the fairness and completeness conditions of Definition 2 are satisfied.

1. Let r be a run of an instance π of P such that J states a verdict ψ. We have to show that $\pi \models \psi$.

   By assumption we have $\psi = \overline{\varphi}$. As by definition, J does not accept r, by (c) we have that $\pi \models \varphi$ does not hold, what implies $\pi \models \overline{\varphi}$, hence J is fair.

2. Further, let r be an arbitrary run of an instance of P. We have to show that J ensures $(\neg\gamma \Rightarrow \overline{\varphi})$ in P. This is trivial when $r \in \gamma$, hence let $r \in \neg\gamma$. If J would not state a verdict

that implies $\overline{\varphi}$, then, by definition, J would accept this run. By (d) we would get $r \in \gamma$. This contradiction implies the completeness.

C.2 Computational Setting

Assume that (4) **holds for some** $\delta \in [0,1]$**.** That means that

(a) J is computationally fair, i.e. $\mathsf{Pr}[\pi(1^\ell) \mapsto \{(J:\psi) \mid \pi \not\models \psi\}]$ is negligible as a function of ℓ, for all instances π of P and

(b) for every instance π of P, the probability that J does not ensure $(\neg\gamma \Rightarrow \overline{\varphi})$ is δ-bounded as a function of ℓ.

We now show that the two conditions of Definition 5 hold.

1. We have to show that $\mathsf{Pr}[\pi(1^\ell) \mapsto (J:\mathsf{accept})]$ is overwhelming as a function of ℓ for all instances π of P with $\pi \models \varphi$.

   So let π with $\pi \models \varphi$, i.e. $\pi \not\models \overline{\varphi}$, be arbitrary. If $\mathsf{Pr}[\pi(1^\ell) \mapsto (J:\mathsf{accept})]$ is not overwhelming, then $\mathsf{Pr}[\pi(1^\ell) \mapsto \{(J:\psi) \mid \psi \text{ implies } \overline{\varphi}\}]$ is not negligible as J accepts iff J does not output any ψ (that implies $\overline{\varphi}$). For all ψ that imply $\overline{\varphi}$, we have $\pi \not\models \psi$. Hence $\mathsf{Pr}[\pi(1^\ell) \mapsto \{(J:\psi) \mid \pi \not\models \psi\}] \geq \mathsf{Pr}[\pi(1^\ell) \mapsto \{(J:\psi) \mid \psi \text{ implies } \overline{\varphi}\}]$ is also not negligible, what contradicts (a).

2. Further we have to show that $\mathsf{Pr}[\pi(1^\ell) \mapsto \neg\gamma,\ (J:\mathsf{accept})]$ is δ-bounded as a function of ℓ.

   This follows directly from (b), as J does not ensure $(\neg\gamma \Rightarrow \overline{\varphi})$ in a run r means that $r \in \neg\gamma$ and J does not state a formula that implies $\overline{\varphi}$. The latter implies that J does not state any formula and hence, J accepts the run. Hence the second condition of computational verifiability is satisfied.

Now we prove that (5) implies (4) under the condition that J, if it states a formula, states $\overline{\varphi}$.

Assume that (5) **holds for some** $\delta \in [0,1]$**.** That means that for every instance π of P

(c) If $\pi \models \varphi$, then $\mathsf{Pr}[\pi(1^\ell) \mapsto (J:\mathsf{accept})]$ is overwhelming as a function of ℓ.

(d) $\mathsf{Pr}[\pi(1^\ell) \mapsto \neg\gamma,\ (J:\mathsf{accept})]$ is δ-bounded as a function of ℓ.

We have to show that the fairness and completeness conditions of Definition 3 are satisfied.

1. Let π be an arbitrary instance of P. We have to show that $\mathsf{Pr}[\pi(1^\ell) \mapsto \{(J:\psi) \mid \pi \not\models \psi\}]$ is negligible as a function of ℓ.

   As if J states a formula, then J states $\overline{\varphi}$, it is enough to show that the probability that J states $\overline{\varphi}$ is negligible if $\pi \not\models \overline{\varphi}$. The latter implies $\pi \models \varphi$ and hence, by (c), $\mathsf{Pr}[\pi(1^\ell) \mapsto (J:\mathsf{accept})]$ is overwhelming. This implies that the probability that J states $\overline{\varphi}$ is negligible, as by definition, J states a formula iff J does not accept. Hence J is computationally fair.

2. Further we have to show that the probability that J ensures $(\neg\gamma \Rightarrow \overline{\varphi})$ is δ-bounded.

   This follows directly from (d), as J does not ensure $(\neg\gamma \Rightarrow \overline{\varphi})$ in a run r means that $r \in \neg\gamma$ and J does not state a formula that implies $\overline{\varphi}$. The latter means that J does not state any formula and hence, J accepts the run. This implies the completeness.

D. ASW PROTOCOL

In this section, we provide some details on modeling and analysis the ASW contract-signing protocol. [4].

D.1 Description of the Protocol

The objective of the ASW protocol is to enable two parties, A (the originator) and B (the responder), to obtain each other's signature on a previously agreed contractual text $\mathsf{contract}$ with the help of a trusted third party T, where, however, T is only invoked in case of a problem. In other words, the ASW protocol is an optimistic two-party contract-signing protocol.

In the following, similarly to Section 6, we write $\mathsf{sig}_k[m]$ as an abbreviation for $\langle m, \mathsf{sig}_k(m)\rangle$, where $\mathsf{sig}_k(m)$ is a term representing the signature on the message m with the key k. We will also write $\langle m_1,\dots,m_n\rangle$ to represent the concatenation of the messages $m_1,\dots,m_n$. We denote the public (or verification) key of a principal A by k_A.

In the ASW protocol, there are two kinds of messages that are considered to be valid contracts: the *standard contract* $\langle \mathsf{sig}_{k_A}[m_A], N_A, \mathsf{sig}_{k_B}[m_B], N_B\rangle$ and the *replacement contract*

$$r_T = \mathsf{sig}_T[\langle \mathsf{sig}_A[m_A], \mathsf{sig}_B[m_B]\rangle]$$

where N_A and N_B are nonces generated by A and B, respectively, $m_A = \langle k_A, k_B, k_T, \mathsf{contract}, \mathsf{hash}(N_A)\rangle$, and $m_B = \langle \mathsf{sig}_{k_A}[m_A], \mathsf{hash}(N_B)\rangle$

The ASW protocol consists of three subprotocols: the *exchange*, *abort*, and *resolve* protocols. These subprotocols are explained next.

Exchange protocol. The basic idea of the exchange protocol is that A first indicates her interest to sign the contract. To this end, she sends to B the message $\mathsf{sig}_{k_A}[m_A]$ as defined above, where N_A is a nonce generated by A. By sending this message, A "commits" to signing the contract. Then, similarly, B indicates his interest to sign the contract by generating a nonce N_B and sending the message $\mathsf{sig}_{k_B}[m_B]$ to A. Finally, first A and then B reveal N_A and N_B, respectively. At this point both participants are able to build a standard contract.

Abort protocol. If, after A has sent her first message, B does not respond, A may contact T to abort, i.e., A runs the abort protocol with T. In the abort protocol, A first sends the message $\mathsf{sig}_{k_A}[\langle \mathsf{aborted}, \mathsf{sig}_{k_A}[m_A]\rangle]$. If T has not received a resolve request before (see below), then T sends back to A the *abort token* $a_T = \mathsf{sig}_{k_T}[\langle \mathsf{aborted}, \mathsf{sig}_{k_A}[\langle \mathsf{aborted}, \mathsf{sig}_{k_A}[m_A]\rangle]\rangle]$. Otherwise (if T received a resolve request, which in particular involves the messages $\mathsf{sig}_{k_A}[m_A]$ and $\mathsf{sig}_{k_B}[m_B]$ from above), it sends the replacement contract r_T to A.

Resolve protocol. If, after A has sent the nonce N_A, B does not respond, A may contact T to resolve, i.e., A runs the resolve protocol with T. In the resolve protocol, A sends the message $\langle \mathsf{sig}_{k_A}[m_A], \mathsf{sig}_{k_B}[m_B]\rangle$ to T. If T has not sent out the abort token a_T before, then T returns the replacement contract r_T, and otherwise T returns the abort token a_T. Analogously, if, after B has sent his commitment to sign the contract, A does not respond, B may contact T to resolve, i.e., B runs the resolve protocol with T similarly to the case for A.

We assume that both in the abort and the resolve protocol, the communication with T is carried out over a reliable channel. Communication between A and B is carried out over an unreliable network channels.

D.2 Properties of the Protocol

Several properties of this protocol were studied in the literature, including fairness, balance, and abuse-freeness, under the assumption that the trusted third party behaves honestly (see, e.g., [29, 43, 28, 27]). More specifically, it was assumed that the trusted third party never produces both the abort token a_T and a replacement contract r_T. Here, we do not make this assumption but ask whether the trusted third party can be held accountable in case it misbehaves. This is a crucial question, as a positive answer justifies the assumption that the trusted third party behaves honestly.

Ideally, we would like to hold T accountable whenever it produces both a_T and r_T. However, it is easy to see that it is unrealistic: the mere fact that both messages were produced does not necessarily mean that they were sent to any honest party, let alone observed by the judge. We therefore consider only the case where there is a dispute in which the judge is faced with both a_T and r_T.

D.3 Modeling

By P_{ASW} we denote the protocol (in the sense of Definition 1) modeling the ASW protocol, where, in addition to A, B and T, we consider an additional party, the *judge* J. The honest programs of A, B and T are defined as specified by the protocol. The judge blames T if and only if he obtains a message of the form $\langle a_T, r_T \rangle$, where a_T and r_T are defined as above, for some A, B, $\mathsf{contract}$, N_A, and N_B.

We assume that the set of programs that the judge can run consists only of his honest program, which means that we assume that the judge is honest. However, the sets of programs of A, B, and T consist of all possible processes that these parties can run, where the processes are merely limited by the network configuration. Hence, for these three parties any dishonest behavior is considered.

D.4 Automated Proof of Theorem 4

Following Remark 1, we have verified the property stated by this theorem automatically, using the constraint solving for protocol analysis tool [35], documented in [36]. Our modelling is available at [33].

Fairness. For the fairness condition, we have encoded the system with only honest T and J and with the intruder subsuming A and B (which means that he knows their private keys and has access to the same network interface these parties have, including the interface to T), where T is capable to deal with up to three requests. The tool has verified that a state in which J blames T is unreachable in this system (note that J blames only T).

Completeness. To show that the only constraint of Φ is ensured by J, we have encoded the system with only honest J and all the remaining parties subsumed by the intruder. In this case, the verification is trivial, and the tool, as expected, has confirmed that it is impossible to reach a state, where the judge obtains both a_T and r_T and T is not blamed.