Accountability: Definition and Relationship to Verifiability

Ralf Küsters, Tomasz Truderung, and Andreas Vogt
University of Trier, Germany
{kuesters,truderung,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 weak 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.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol Verication*; C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*

General Terms

Security, Verification

1. INTRODUCTION

Many cryptographic tasks and protocols, such as non-repudiation [25], contract-signing [3], voting [6], auctions [19], identity-based encryption [11], and certain forms of secure multi-party computation [14], 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.

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 [19], 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,

CCS'10, October 4–8, 2010, Chicago, Illinois, USA.

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 present 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) and 6 (auction). Due to lack of space, our case study on contract signing is provided only in our technical report [17], which contains the full details.

2. ACCOUNTABILITY

In this section, we present our definition of accountability. As mentioned in the introduction, we provide 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 constrains. 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. This assumption is convenient for our case studies, but otherwise not essential. Possible symbolic instances of our framework include the applied π-calculus [2] and models based on I/O-automata, see, e.g., [16]. In a computational model, processes would typically be modeled as probabilistic polynomial-time systems of probabilistic polynomial-time interactive Turing machines (ppt ITMs). Our case studies provide concrete examples.

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:

- $\Sigma = \{\mathsf{a}_1, \ldots, \mathsf{a}_n\}$ 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 $\pi = (\pi_{\mathsf{a}_1} \parallel \ldots \parallel \pi_{\mathsf{a}_n})$ 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. If it is interpreted as "not running the honest program", the resulting completeness condition would be much too strong. No protocol would satisfy this condition: It requires J to blame agents for misbehavior that is impossible to be observed by any other party and 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. More importantly, such a completeness condition would miss the main point: Misbehavior that

cannot be observed by any honest party may still be very relevant and harmful. We therefore advocate a completeness condition 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 the misbehavior 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 better to have a language in which the level of accountability 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 $A_1,\ldots,A_r$, if the judge states, say, $\mathsf{dis}(M) \wedge \mathsf{dis}(A_1) \wedge \ldots \wedge \mathsf{dis}(A_r)$, then this expresses the judge's belief that the voting machine and all auditors misbehaved; the judge would state $\mathsf{dis}(M) \vee (\mathsf{dis}(A_1) \wedge \ldots \wedge \mathsf{dis}(A_r))$ 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 $\psi_1,\ldots,\psi_k$, which is equivalent to stating the verdict $\psi_1 \wedge \cdots \wedge \psi_k$.

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 $(\alpha,\psi_1,\ldots,\psi_k)$, written $(\alpha \Rightarrow \psi_1 \mid \cdots \mid \psi_k)$, where α is a property of P and $\psi_1,\ldots,\psi_k \in \mathscr{F}_{\mathsf{dis}}$. We say that such a constraint *covers* a run r, if $r \in \alpha$.

Intuitively, in a constraint $C = (\alpha \Rightarrow \psi_1 \mid \cdots \mid \psi_k)$, the set α contains runs in which some desired goal of the protocol is not met (due to the misbehavior of some protocol participant). The formulas $\psi_1,\ldots,\psi_k$ are the possible (minimal) verdicts that are supposed to be stated by J in such a case; J is free to state stronger verdicts (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 $\psi_1,\ldots,\psi_k$ (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, $A_1,\ldots,A_r$ are auditors, and α contains all runs in which the published result of the election is incorrect:

$$C_1^{ex} = \alpha \Rightarrow \mathsf{dis}(M) \mid \mathsf{dis}(A_1) \mid \cdots \mid \mathsf{dis}(A_r) \quad (1)$$

$$C_2^{ex} = \alpha \Rightarrow \mathsf{dis}(M) \vee (\mathsf{dis}(A_1) \wedge \cdots \wedge \mathsf{dis}(A_r)) \quad (2)$$

$$C_3^{ex} = \alpha \Rightarrow \mathsf{dis}(M) \mid \mathsf{dis}(A_1) \wedge \cdots \wedge \mathsf{dis}(A_r). \quad (3)$$

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, $A_1,\ldots,A_r$ 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 $\mathsf{dis}(A_1) \wedge \cdots \wedge \mathsf{dis}(A_r)$ 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 $(\alpha \Rightarrow \psi_1 \mid \cdots \mid \psi_k)$ provides *individual accountability*, if for every $i \in \{1,\ldots,k\}$, there exists a party a such that ψ_i implies $\mathsf{dis}(a)$. In other words, each $\psi_1,\ldots,\psi_k$ 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 (see our technical report [17]) 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. Clearly, one could also consider δ-boundedness for the fairness condition, however, this is typically not necessary.

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 J (a voter, in this case) could, for example, include all runs in which her vote is counted as cast; this goal aims at what is called *individual verifiability* [21]. 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* [21]. In [22], 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 $A_1,\ldots,A_r$, 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 $A_1,\ldots,A_r$ 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 our full version [17] 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 would not be possible to hold 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., [3, 10, 24, 23, 9, 8, 18, 6, 7]).

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. [13]. 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, 5], 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 [4], 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 [12], 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; if no audit action is triggered by an honest agent, no party is required to be blamed. 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.

In [22], three types of verifiability, namely *eligibility verifiability*, *universal verifiability*, and *individual verifiability* are formal-

ized 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 Jakobsson [15] present a cryptographic definition of verifiability, which is specifically tailored to their voting protocol [15].

5. ANALYZING BINGO VOTING

In this section, we analyze accountability and verifiability properties of the Bingo voting system [6] in the cryptographic setting. This is the first such analysis. It reveals some interesting new problems and features of the system, not observed before. For example, it turns out that the system does not provide individual accountability and that the level of accountability/verifiability the system provides does not depend on whether or not the random number generator used in Bingo Voting is honest. Our analysis also illustrates the usefulness 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*, 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 [17]).

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 an honest 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 (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 [17].

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; we stick, however, 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 is that the result is correct up to the votes of dishonest voters. More precisely, γ is satisfied in a run if the published result reflects the actual choices of the honest voters in this run and some choices of dishonest voters, where the latter choices may differ from the actual choices made by dishonest voters. This goal seems realistic and we believe that it is the goal every voting system should aim for. In particular, in 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.

Problems. By applying our definition of accountability, we identified the following problems of the Bingo voting system.

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 candi-

date 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 then 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 the full version of this paper [17].

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 the same number occurs twice. 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 J. In the following, we describe the behavior of the agent J:

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, J 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, J 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 J states $\mathsf{dis}(M) \vee \mathsf{dis}(RNG)$, as explained above (Problem 2).

J4. The agent J 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, J 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 the full version [17] for details). We denote this protocol by $\mathsf{P}^{J}_{\mathsf{Bingo1}}(n, q_{num}, q_{rec}, s, \vec{p})$, where the agent J 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 [17]).

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.

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:

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

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 $p_1, \ldots, p_l$ are as introduced in Section 5.1.

We show (in the full version [17]) that the protocol is accountable for Φ_1, where Φ_1 consists of the following constraints:

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

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

This theorem says that, in $\mathsf{P}^J_{\mathsf{Bingo1}}$, the probability that the goal γ_k is not achieved and J 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 $\max_{j=1,\ldots,l} p_j$ in the definition of $\delta^k_{\mathsf{Bingo}}$ in the above theorem. 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 $((\neg\gamma \Rightarrow \psi), \delta^k_{\mathsf{Bingo}})$-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 J be an external judge or a voter. Under the DLOG-assumption, in $\mathsf{P}^J_{\mathsf{Bingo1}}(n, q_{num}, q_{rec}, s, \vec{p})$, the goal γ_k is guaranteed by $\bigwedge_{a\in\Sigma} \mathsf{hon}(a)$ and $\delta^k_{\mathsf{Bingo}}$-verifiable by J.*

This corollary says that, in $\mathsf{P}^J_{\mathsf{Bingo1}}$, 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 J (recall that J uses only public information). This means that J, 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 $\mathsf{P}^J_{\mathsf{Bingo1}}(n, q_{num}, q_{rec}, s, \vec{p})$ already revealed by Theorem 1: By wrongly complaining, every single dishonest voter can spoil the election process. This weakness is not present in variant II) mentioned above, which, however, comes at a price of a weaker goal (see the full version [17]).

6. THE PRST PROTOCOL

In this section, we study the auction protocol proposed by Parkes, Rabin, Shieber, and Thorpe [19]. More precisely, we study here one of a few variants of the protocol proposed in [19], 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. Due to lack of space, our improved version of the original protocol and its analysis are presented in our technical report [17] only.

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[2]), 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 $B_1, \ldots, B_n$, the auctioneer A, and the notaries $N_1, \ldots, N_l$. 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 $B_1, \ldots, B_n$ 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 [19] 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: $Com_{\pi(1)}, \ldots, Com_{\pi(n)}$, 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 $C_{\pi(1)}, \ldots, C_{\pi(n)}$. Anybody can verify whether all the commitments posted in S3 have been correctly opened.

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

[1] From this assumption, it follows that it is infeasible to open a Pedersen-commitment to two different values [20].

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

used in C_u, which he can recover using his private key. The auctioneer posts

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

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 $C_1,\ldots,C_n$. 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 $C_1,\ldots,C_n$, and (iii) an honest bidder is declared to be the winner if and only if her bid is the highest in $C_1,\ldots,C_n$.

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 [19].

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 X, 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 $B_{i_1},\ldots,B_{i_l}$ on these commitments. If A refuses to do so, V blames A; otherwise, V states $\psi^*_{\{A,B_{i_1},\ldots,B_{i_l}\}}$.

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 $X\in\{B_1,\ldots,B_n\}$, 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 the full version of the paper [17].

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:

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

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

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

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

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

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

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

The proof of this theorem is given in [17]. 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, the 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 $V \in \{J,B_1,\ldots,B_n\}$.*

Acknowledgements. This work was partially supported by the DFG under Grant KU 1434/5-1.

7. 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 *POPL 2001*, p. 104–115, ACM.

[3] N. Asokan, V. Shoup, and M. Waidner. Asynchronous Protocols for Optimistic Fair Exchange. In *S&P 1998*, p. 86–99, IEEE Computer Society.

[4] Y. Aumann and Y. Lindell. Security against Covert Adversaries: Efficient Protocols for Realistic Adversaries. In *TCC 2007*, vol. 4392 of *LNCS*, p. 137–156. Springer.

[5] G. Bella and L. Paulson. Accountability Protocols: Formalized and Verified. *ACM Trans. Inf. Syst. Secur.*, 9(2):138–161, 2006.

[6] J.-M. Bohli, J. Müller-Quade, and S. Röhrich. Bingo Voting: Secure and Coercion-Free Voting Using a Trusted Random Number Generator. In *VOTE-ID 2007*, vol. 4896 of *LNCS*, p. 111–124. Springer.

[7] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Ryan, E. Shen, and A. Sherman. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. In *EVT 2008*, USENIX Association.

[8] D. Chaum, P. Ryan, and S. Schneider. A Practical, Voter-verifiable Election Scheme. In *ESORICS 2005*, vol. 3679 of *LNCS*, p. 118–139, Springer.

[9] B. Chevallier-Mames, P.-A. Fouque, D. Pointcheval, J. Stern, and J. Traoré. On Some Incompatible Properties of Voting Schemes. In *WOTE 2006*.

[10] J.A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free Optimistic Contract Signing. In *CRYPTO 1999*, vol. 1666 of *LNCS*, p. 449–466, Springer.

[11] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box Accountable Authority Identity-based Encryption. In *CCS 2008*, p. 427–436, ACM.

[12] N. Guts, C. Fournet, and F. Nardelli. Reliable Evidence: Auditability by Typing. In *ESORICS 2009*, p. 168–183, Springer.

[13] R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely. Towards a Theory of Accountability and Audit. In *ESORICS 2009*, p. 152–167, Springer.

[14] W. Jiang, C. Clifton, and M. Kantarcioglu. Transforming Semi-honest Protocols to Ensure Accountability. *Data Knowl. Eng.*, 65(1):57–74, 2008.

[15] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant Electronic Elections. In *WPES 2005*, p. 61–70, ACM Press.

[16] D. Kähler, R. Küsters, and Th. Wilke. A Dolev-Yao-based Definition of Abuse-free Protocols. In *ICALP 2006*, vol. 4052 of *LNCS*, p. 95–106, Springer.

[17] R. Küsters, T. Truderung, and A. Vogt. Accountabiliy: Definition and Relationship to Verifiability. ePrint, Report 2010/236.

[18] T. Moran and M. Naor. Split-ballot Voting: Everlasting Privacy with Distributed Trust. In *CCS 2007*, p. 246–255. ACM.

[19] D. Parkes, M. Rabin, S. Shieber, and C. Thorpe. Practical Secrecy-preserving, Verifiably Correct and Trustworthy Auctions. In *ICEC 2006*, p. 70–81.

[20] T. Pedersen. Non-interactive and Information-theoretic Secure Verifiable Secret Sharing. In *CRYPTO 1991*, vol. 576 of *LNCS*, p. 129–140, Springer.

[21] K. Sako and J. Kilian. Receipt-Free Mix-Type Voting Scheme — A Practical Solution to the Implementation of a Voting Booth. In *EUROCRYPT 1999*, vol. 921 of *LNCS*, p. 393–403, Springer.

[22] B. Smyth, M. Ryan, S. Kremer, and M. Kourjieh. Election Verifiability in Electronic Voting Protocols. In *WISSec 2009*.

[23] M. Talbi, B. Morin, V. Tong, A. Bouhoula, and M. Mejri. Specification of Electronic Voting Protocol Properties using ADM Logic: Foo Case Study. In *ICICS 2008*, vol. 5308 of *LNCS*, p. 403–418, Springer.

[24] Li Yunfeng, He Dake, and Lu Xianhui. Accountability of Perfect Concurrent Signature. *ICCEE 2008*, p. 773–777.

[25] J. Zhou and D. Gollmann. A Fair Non-repudiation Protocol. In *S&P 1996*, p. 55–61, IEEE Computer Society Press.