Computationally Sound Analysis of a Probabilistic Contract Signing Protocol

Mihhail Aizatulin, Henning Schnoor, and Thomas Wilke

Institut für Informatik, Christian-Albrechts-Universität zu Kiel, 24098 Kiel, Germany
mai@informatik.uni-kiel.de, {schnoor,wilke}@ti.informatik.uni-kiel.de

Abstract. We propose a probabilistic contract signing protocol that achieves balance even in the presence of an adversary that may delay messages sent over secure channels. To show that this property holds in a computational setting, we first propose a probabilistic framework for protocol analysis, then prove that in a symbolic setting the protocol satisfies a probabilistic alternating-time temporal formula expressing balance, and finally establish a general result stating that the validity of formulas such as our balance formula is preserved when passing from the symbolic to a computational setting. The key idea of the protocol is to take a "gradual commitment" approach.

1 Introduction

Contract-signing protocols (CSPs) [BOGMR90, ASW98, GJM99] form a class of cryptographic protocols with complex security goals, which require to explicitly reason about strategies of the involved principals. To analyze CSPs, various techniques have been applied, including a specialized logic [BDD+06], alternating-time temporal logic [KR03, KR02] as well as abstract [KKW05] and computational [CKW07] models. In this paper, we (i) present a new CSP of which we prove that it achieves a central security goal (balance) in the presence of an adversary stronger than the adversaries considered in prior work and (ii) propose a setting where probabilistic strategic security properties of protocols can be transferred from a symbolic to a computational setting.

Recall that a CSP is a security protocol where two partners, Alice and Bob, attempt to sign a contract over a network and that a central security requirement of CSPs is *balance*: No situation should occur in which Bob has a strategy to *resolve the protocol* (obtain a contract) and another strategy to *abort the protocol* (prevent Alice from ever obtaining a contract). Such a situation can be used as an advantage in negotiations with a third party. It is known that to achieve balance a *trusted third party* (TTP) is necessary [PG99], but such a party is also a potential bottleneck. Therefore, a desirable property of CSPs is *optimism*: If Alice and Bob follow the protocol and no network problems occur, the TTP should not be involved in the protocol run [ASW98]. Balanced and optimistic CSPs have been proposed in the above mentioned papers [ASW98] and [GJM99]. These protocols, however, achieve balance only under the assumption that Bob has no way to ensure that his message is the first to reach the TTP.

M. Backes and P. Ning (Eds.): ESORICS 2009, LNCS 5789, pp. 571–586, 2009.

The first contribution of our paper is a protocol that achieves balance under the relaxed assumption that Bob is allowed to arbitrarily delay messages between Alice and the TTP. Clearly, we cannot allow that he prevents their delivery completely. Our protocol achieves the following *probabilistic* notion of balance: The $(n+1)$-round version ensures that in any situation during a protocol run, if Bob has a strategy to resolve the protocol with success probability p_r, then he does not have an abort strategy with success probability greater than $1+\frac{1}{n}-p_r$. Note that for every reasonable protocol there is a state in which the sum of these probabilities is 1, for example in a final state of the protocol run.

The second contribution of this paper is a formal framework in which one can prove security properties of a subclass of probabilistic protocols that use signatures schemes. Our model is a symbolic model in the Dolev-Yao style [DY83], but we also provide it with a computational semantics, based on [BR93]. We explain how probabilistic alternating-time temporal formulas [AHK02, CL07] can be interpreted with regard to both semantics. Our main result is that the validity of a certain class of formulas is preserved when passing from the symbolic to the computational setting, that is, we show that the symbolic setting is computationally sound. Using this and the fact that our protocol is balanced in the symbolic setting and balance can be phrased in alternating-time temporal logic, we obtain that our protocol is balanced in the computational setting (for a single session).

Related Work. In [KKW05], a symbolic definition of balance for CSPs is introduced, and the problem to decide whether a CSP is balanced is proved decidable. In [KKT07], decidability for security properties specified in the μ-calculus (an extension of ATL) is proved. The notion of balance was first defined in [CKS01]; in [CMSS05] an impossibility result concerning balance was established. In [KR03] and [KR02], it was shown how ATL formulas can be used to specify security properties for cryptographic protocols, in particular, CSPs were dealt with. [CKW07] proposes a computational model for analyzing branching-time security properties, which includes a computational definition of balance. Our treatment of strategies in the computational model shares ideas with their use of schedulers. The first result proving that symbolic security transfers to the computational model was obtained in [AR02]. Many generalizations for different kinds of computational models followed (see, e.g., [CKKW06], [CH06], [LM05]).

Our protocol resembles the CSP of Ben-Or et al. [BOGMR90]. In both cases the signers exchange messages that give them increasing power to obtain a replacement contract from the TTP. However, the behavior of the TTP is quite different: In our protocol, if honest Alice gets a rejecting response from the TTP, she can be sure that the TTP will never resolve a request of Bob. In the protocol from [BOGMR90], this is not the case; for instance, when Alice sends the first message to Bob, she has no option to prevent him from trying to resolve the contract at any later time. Thus the resulting state is neither timely nor balanced for Alice. In fact, there are states reachable with non-negligible probability, in which Bob has both a certain strategy to abort and a certain strategy to resolve the contract. In fact, the sum of the two probabilities from above is 2 in [BOGMR90] (as opposed to $1+\frac{1}{n}$ as in our protocol).

Due to the page limit, all proofs and important details of the model are omitted. For details, see the technical report [ASW09].

2 The Gradual Commitment Protocol (GCP)

Recall that, informally, a contract signing protocol is unbalanced if in some situation the dishonest party has both a strategy to abort the protocol run (prevent the honest party from receiving a valid contract) and a strategy to resolve the protocol run (to receive a valid contract). We define the following probabilistic measure of degree of unbalance: The unbalance of a state is at least ϵ if the dishonest party has a strategy that leads from the state to an abort with probability ϵ_a and, in addition, a strategy that leads from the state to a resolve with probability ϵ_r, and $\epsilon_a + \epsilon_r \geq 1 + \epsilon$. Observe that (i) for every state in which a party has obtained a contract, the unbalance is at least 0, (ii) for an unbalanced protocol without randomness, the unbalance is at least (and at most) 1. Our protocol is tailored to guarantee low unbalance: In the version with parameter n the unbalance in any reachable state is not greater than $\frac{1}{n}$.

The protocol is based on the idea of "gradual commitment". The version with parameter n proceeds in $n+1$ rounds and makes sure that the later the round is the higher is the probability to be able to resolve a protocol run, while, conversely, the lower is the probability to be able to abort a protocol run. Hence the unbalance is low in any state.

In an ordinary run of the protocol the contracting parties exchange $2n+2$ *commitments*, which we refer to by CMT_X^i for $X \in \{O, R\}$ and $i \in \{1, \ldots, n+1\}$. First, the *originator*, O, sends CMT_O^1, then the *responder*, R, sends CMT_R^1, and so on, the last commitment being CMT_R^{n+1}. The pair of the last two commitments, $\langle \mathsf{CMT}_O^{n+1}, \mathsf{CMT}_R^{n+1} \rangle$, is a *valid contract*. The commitments are defined by

$$\mathsf{CMT}_X^i = [\mathsf{text}, O, R, T, i]_X,$$

where text is the document the two parties want to sign, O and R are identifiers for the parties, T is an identifier for the *trusted third party (TTP)*, which can resolve conflicts, and i is the round number. The notation $[\cdot]_X$ stands for a message and its signature.

A commitment CMT_O^i with $i \in \{1, \ldots, n\}$ can be used by R to form a resolve request, RR_R^i, addressed to the TTP. Similarly, a commitment CMT_R^i can be used by O to form a resolve request, RR_O^i. The precise format is

$$\mathsf{RR}_O^i = [\mathsf{CMT}_O^{i+1}, \mathsf{CMT}_R^i]_O, \qquad \mathsf{RR}_R^i = [\mathsf{CMT}_O^i, \mathsf{CMT}_R^i]_R,$$

for $i \in \{1, \ldots, n\}$. In addition, there is one resolve request that O can always form (without having received any commitment by R): $\mathsf{RR}_O^0 = [\mathsf{CMT}_O^1, \mathsf{abort}]_O$, where abort is a fixed token.

Possible replies of T to $m = \mathsf{RR}_X^i$ are the *replacement contract*, denoted $\mathsf{R\text{-}CTR}_X^i$, and defined by $\mathsf{R\text{-}CTR}_X^i = [m]_T$, which is recognized as a *valid contract*, or a *rejection*, $\mathsf{RT}_X^i = [m, \mathsf{rejected}]_T$, where $\mathsf{rejected}$ is a fixed token such as abort.

To formulate the rules by which T handles incoming resolve requests we define a relation $<$ on the resolve requests: $\mathsf{RR}_R^i < \mathsf{RR}_O^j$ if $i < j$ and $\mathsf{RR}_O^i < \mathsf{RR}_R^j$ if $i + 1 < j$. This implies that if $m < m'$ and m is a resolve request by X, then m contains a weaker commitment of X than m'. When T receives a message $m = \mathsf{RR}_X^i$, it reacts according to the following:

1. If m is the first request, resolve m (i.e., send the replacement contract) with probability i/n and reject (i.e., send a rejection) with probability $(n - i)/n$.
2. If any request by X was received before, ignore m.
3. If any request by $\bar{X}$ was received before, say m', then:
   (a) If m' was resolved, then resolve m.
   (b) If m' was rejected and $m' < m$, then resolve m with probability i/n and reject with probability $(n - i)/n$.
   (c) If m' was rejected and $m' \not< m$, then reject m.

Note that if the TTP rejects a resolve request, it may still accept a later one—but only if the party sending the first request "cheats."

3 The Symbolic Protocol Model

To describe our symbolic protocol model we first explain how protocols are defined in our setting and define the set of available actions for the principals and the adversary during each state of a possible protocol execution. The model introduced here is a standard Dolev-Yao model [DY83] extended with the possibility to specify probabilistic actions. This model of protocol execution suffices to test protocols for reachability properties such as nonce secrecy or authenticity, but this is not treated here. We use the model only to define, in the subsequent section, a more complex model in which we can reason in alternating-time temporal logic about strategies available to all principals, which we then use to analyze contract signing protocols.

3.1 Variables, Terms, and Messages

We fix a finite set *Ids* of *identities* and a number k of *roles* that participate in a protocol session. The adversary is denoted by $\mathscr{A}$. We fix a set $\mathscr{V}$ of variables that are typed and have a restriction specifying the maximal depth of a term that principals accept as part of incoming messages. Terms are constructed in the usual way from nonces and constants, where each principal (including the adversary) has a unique set of nonces. The relevant operations are *pairing* and *signing*: the pairing of terms t_1 and t_2 is denoted $\langle t_1, t_2 \rangle$; a term t' signed by the key of A is denoted $\mathsf{sig}(A, N, t')$ where N is a *randomization nonce* used to capture randomness explicitly.

A *message* is a variable-free term. We denote the set of messages by $\mathscr{M}$. A *substitution* is a partial function $\sigma\colon \mathscr{V} \to \mathscr{M}$. By $dom(\sigma)$, we denote the domain of σ. For a term t and a substitution σ, by $t\sigma$ we denote the term

obtained from t by replacing every variable $x \in dom(\sigma)$ appearing in t with $\sigma(x)$. For a term t, substitutions σ and σ', and a message m, we say that m *matches with* t *and* σ *via* σ' if $t\sigma' = m$ and $\sigma'(x) = \sigma(x)$ for every $x \in dom(\sigma)$ (and natural depth- and typing-restrictions are satisfied).

The adversary can derive messages as follows: For a set $\mathscr{I}$ of messages and a set C of (corrupted) principals, the set $d(\mathscr{I}, C)$ of the messages *derivable from* $\mathscr{I}$ *with corrupted* C is the set containing all constants and adversary-generated nonces as well as messages that can be obtained from $\mathscr{I}$ by pairing, unpairing, and signing a message with a key of a corrupted principal.

3.2 Protocols

We distinguish two types of *protocol rules.* A *strategic rule* is of the form $r \longrightarrow_d s$ where r and s are terms and $d \in \mathbb{N} \cup \{\mathscr{A}\}$. The meaning is that s is sent to d as a reaction to r. A *randomized rule* is of the form $\epsilon \longrightarrow_d^p s$ where p is the probability of the rule.

A *role* $\Pi = (V, E, v_0, \ell)$ is a finite directed edge-labeled tree where (V, E) is a tree with root v_0 and ℓ is a labeling mapping every edge $(v, v') \in E$ to a protocol rule $\ell(v, v')$. The tree (V, E) is the *role tree* and its vertices are the *(local) states* of the role. We require that for a role Π, there is at most one identity A such that Π_i uses nonces belonging to A or signs terms in A's name. We say that A is the *identity of* Π.

For technical reasons, we only allow *randomized* and *strategic* local states: A *randomized local state* is a vertex v where (i) all outgoing edges are labeled with probabilistic rules of the form $\epsilon \longrightarrow_d^p s$, (ii) the probabilities of the outgoing edges sum up to 1, and (iii) all incoming edges are labeled with a strategic rule of the form $r \longrightarrow_d \epsilon$. A *strategic local state* is a state where all outgoing edges are labeled with strategic rules. We partition roles into *network-accepting* and *network-ignoring* roles: The former accept incoming messages from the network (i.e., the adversary), the latter ignore all incoming network messages. In our definition of protocol execution (see below), the adversary may only send messages to network-accepting rules. A *k-roles protocol* is a tuple $Pr = (\Pi_1, \ldots, \Pi_k, \mathscr{I}_0)$ where each Π_i is a protocol role and $\mathscr{I}_0$ is a finite set of messages, the *initial adversary knowledge*. We assume that different protocol rules use disjoint sets of local states and variables.

In order for protocols to be "realistic," roles may only create their own signatures (but may send signatures they have received earlier), and must use different randomization when signing different terms.

3.3 Symbolic Protocol Execution

We define how a protocol $Pr = (\Pi_1, \ldots, \Pi_k, \mathscr{I}_0)$ is executed in our model. A *global state* of Pr is a tuple $q = (a, \sigma, v_1, \ldots, v_k, \mathscr{I}, C, m)$, where $a \in \{1, \ldots, k\} \cup \{\mathscr{A}, \mathscr{S}, \mathscr{K}\}$ is the *active* role, σ is a substitution, v_i is a local state of Π_i, $\mathscr{I}$ is a set of messages, $C \subseteq Ids \cup \{\mathscr{A}\}$, and m is a message. Here, $\mathscr{S}$ represents the *scheduler*, who determines the order of activation in a protocol run, and $\mathscr{K}$

denotes the key generator. For an identity $a \in C$, we say that a is *corrupted* in q. The message m is currently waiting to be processed. With $d(q)$ we denote the set $d(\mathscr{I}, C)$. We define a graph containing all global states of Pr. The *initial state* of Pr is $(\mathscr{K}, \emptyset, v_0^1, \ldots, v_0^k, \mathscr{I}_0, \{\mathscr{A}\}, \epsilon)$, where v_0^i is the root of Π_i. For a state $q = (a, \sigma, v_1, \ldots, v_k, \mathscr{I}, C, m)$, its *successor states* are as follows:

Key generation and initialization. If q is the initial state, then there is a successor state $(\mathscr{A}, \emptyset, v_0^1, \ldots, v_0^k, \mathscr{I}_0, \{\mathscr{A}\}, \epsilon)$,

Corruption of identities. If $a = \mathscr{A}$ and $C = \{\mathscr{A}\}$, then for every set $C' \subseteq Ids$, q has a successor state $(\mathscr{S}, \emptyset, v_0^1, \ldots, v_0^k, \mathscr{I}_0, C', \epsilon)$. This expresses that after key generation, the adversary may corrupt identities.

Adversary send. Assume that $a = \mathscr{A}$ and $m = \epsilon$. Let $m' \in d(q)$ be a message, and let $i \leq k$ such that Π_i is network-accepting. Then there is a successor state $(i, \sigma, v_1, \ldots, v_k, \mathscr{I}, C, m')$. This models that the adversary can deliver the message m' to any network-accepting role, which is activated next.

Adversary receive. Assume that $a = \mathscr{A}$, and $m \neq \epsilon$. Then q has exactly one successor state, namely $(\mathscr{S}, \sigma, v_1, \ldots, v_k, \mathscr{I} \cup \{m\}, C, \epsilon)$. This models that when a principal sends a message over the network, the next step is to add this message to the adversary knowledge. Before the adversary can perform any further action, control is returned to the scheduler.

Principal receive and send. If $a = i \in \{1, \ldots, k\}$, then for each successor v_i' of v_i such that $\ell(v_i, v_i')$ contains the rule $r \to_d s$ or $r \longrightarrow_d^p s$, and there is a substitution σ' such that m matches with r and σ via σ', there is a successor $(d, \sigma', v_1, \ldots, v_{i-1}, v_i', v_{i+1}, \ldots, v_k, \mathscr{I}, C, s\sigma')$ of q, provided that $d \neq \mathscr{A}$ or $s\sigma' \neq \epsilon$. If $d = \mathscr{A}$ and $s\sigma' = \epsilon$ or if there is no v_i' as above, q has a successor state $(\mathscr{S}, \sigma', v_1, \ldots, v_k, \mathscr{I}, C, \epsilon)$. This models that the receiver of a non-empty message sent by a role is activated next to process the message.

Activation scheduling. If $a = \mathscr{S}$, then for all $a' \in \{\mathscr{A}\} \cup \{1, \ldots, k\}$ there is a successor state $(a', \sigma, v_1, \ldots, v_k, \mathscr{I}, C, \epsilon)$. This models that the scheduler can activate any role (or the adversary).

There is one exception to the above rules: In the unique state with $a = \mathscr{A}$ and $C = \{\mathscr{A}\}$ (the state after key generation), only successors obtained by the "corruption rule" are allowed.

Due to the page limit, the above description of the protocol model leaves out some important details, in particular rules avoiding "dead end loops" and infinite protocol runs. See [ASW09] for the complete set of rules.

Our protocol model allows principals to send messages directly to other principals, and ignore messages delivered by the adversary. Hence one could avoid many security problems in protocols by simply letting all communication use these (unrealistic) direct links, disabling the adversary from taking any relevant action in a protocol run. We allow direct links since many security goals cannot be achieved when the adversary may prevent message delivery completely: Optimistic contract signing cannot be realized fairly without a trusted third party [PG99], and obviously the adversary must not be able to circumvent delivery of messages to the trusted third party completely.

To realistically express this situation in our model, one can introduce a special party which serves as a "buffer" between other principals, whose only function is to relay received messages between other principals (but also forwards all received messages to the adversary). This approach has the advantage that one can explicitely speak about "strategies" of this "buffer principal," see [ASW09] for details.

Observe that the above list only fixes the *set* of possible successor states, and does not state which of the available successor states is entered in an actual protocol run. The semantics of probabilistic protocol execution are defined by the game structure induced by a protocol, see Section 4.2.

4 Probabilistic ATL and GS

We now define the logical framework in which we analyze security properties of cryptographic protocols. We use alternating-time temporal logic (ATL*), as introduced in [AHK02], extended with probabilistic operators as considered in [CL07].

4.1 Game Structures and Strategies

Definition 4.1 (probabilistic game structure). *A* probabilistic game structure *(PGS) is a* 6*-tuple* $\mathscr{G} = (PR, Q, \Delta, \delta, \Pi, PV)$ *where*

- PR *is a finite set of* principals,
- Q *is a (possibly infinite) set of* states,
- PV *is a finite set of* propositional variables,
- $\Pi\colon PV \to 2^Q$ *is a* propositional truth assignment,
- Δ *is a* move function *assigning to each state* q *and principal* $a \in PR$ *a set* $\Delta(q, a)$ *of moves,*
- δ *is a probabilistic* transition function[1].

It is required that for each $q \in Q$ *there is at most one principal* $a \in PR$ *with* $\Delta(q, a) \neq \emptyset$*. This unique principal* a *is denoted by* $Pr(q)$*. The transition function must be such that* $\delta(q, m) \in Q$ *for all* $q \in Q$ *and* $m \in \Delta(q, Pr(q))$*. For each* q *and* m *the* support *of* $\delta(q, m)$*, i.e., the set* $\{q' \in Q \mid prob(\delta(q, m) = q') > 0\}$*, must be finite.*

For a set $A \subseteq PR$, let $\overline{A} = PR \setminus A$. We say that a state q is *final* if $Pr(q)$ is undefined, i.e., $\Delta(q, a) = \emptyset$ for all $a \in PR$.

Definition 4.2 (strategies). *Let* $\mathscr{G} = (PR, Q, \Delta, \delta, \Pi, PV)$ *be a GS.*

1. *A* strategy *for a principal* $a \in PR$ *is a function* s *such that for all* $q \in Q$*, if* $Pr(q) = a$*, then* $s(q) \in \Delta(q, a)$*.*
2. *A* strategy *for* $A \subseteq PR$ *is a set* $S_A = \{s_a \mid a \in A\}$ *such that for each* $a \in A$*,* s_a *is a strategy for* a*.*

[1] For a state and a move, δ specifies a probability distribution on the possible successor states.

Note that strategies depend on the *state* only, and not on the history of the computation. History-aware strategies can be defined analogously.

Let $S_{PR} = \{S_a \mid a \in PR\}$ be a strategy for PR, and let $P = p_0 p_1 p_2 \dots$ be a path, i.e., a (possibly infinite) sequence of states in $\mathscr{G}$. By $|P|$, we denote the number of states in P (which might be ∞); $P[i]$ denotes the ith state on P, and $P[i, \infty]$ is the sub-path of P starting at $P[i]$. We now define

$$\mathrm{prob}_{S_{PR}}(P) = \prod_{i<|P|} \left(prob(\delta(p_i, s_{Pr(p_i)}) = p_{i+1})\right),$$

i.e., the probability that when the principals follow their strategies from S_{PR}, the resulting play follows the path P. For a set S of paths, $\mathrm{prob}_{S_{PR}}(S)$ is the probability of the resulting path being an element of S.

We now define syntax and semantics of pATL* (see also [CL07]).

Definition 4.3 (probabilistic alternating-time pemporal formulas).
- *Each propositional variable $p \in PV$ is a state formula.*
- *If φ, ψ are state formulas, then so are $\varphi \wedge \psi$, $\varphi \vee \psi$ and $\neg\varphi$.*
- *If $A \subseteq PR$, $\alpha \in [0,1]$, and φ is a path formula, then $\langle\langle A\rangle\rangle^{\geq\alpha}\varphi$, $\langle\langle A\rangle\rangle^{\leq\alpha}\varphi$, $[[A]]^{\geq\alpha}\varphi$, and $[[A]]^{\leq\alpha}\varphi$ are state formulas (analogously for $<$ and $>$).*
- *Every state formula is a path formula.*
- *If φ, ψ are path formulas, then so are $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\neg\varphi$.*
- *If φ and ψ are path formulas, then $\varphi \mathsf{U} \psi$ and $\varphi \mathsf{R} \psi$ are path formulas.*

A pATL**-formula is a state formula, unless explicitly specified otherwise.*

In the following we usually omit the cases $\leq\alpha$, $<\alpha$, and $>\alpha$, these are always treated in the obvious way.

Definition 4.4 (semantics of logic). *Let $\mathscr{G} = (PR, Q, \Delta, \delta, \Pi, PV)$ be a GS.*
- *For a variable $p \in PV$, $\mathscr{G}, q \models p$ if and only if $q \in \Pi(p)$.*
- *Boolean connectives are treated as usual.*
- *Let $A \subseteq PR$ and $\alpha \in [0,1]$. Then*
  - *$\mathscr{G}, q \models \langle\langle A\rangle\rangle^{\geq\alpha}\varphi$ iff there is a strategy S_A for A such that for all strategies $S_{\overline{A}}$ for $\overline{A}$, $prob_{S_A \cup S_{\overline{A}}}(\{P \mid \mathscr{G}, P \models \varphi, P[0] = q\}) \geq \alpha$,*
  - *$\mathscr{G}, q \models [[A]]^{\geq\alpha}\varphi$ iff for all strategies S_A for A, there is a strategy $S_{\overline{A}}$ for $\overline{A}$ such that $prob_{S_A \cup S_{\overline{A}}}(\{P \mid \mathscr{G}, P \models \varphi, P[0] = q\}) \geq \alpha$.*
- *For a path P and a state formula φ, $\mathscr{G}, P \models \varphi$ iff $\mathscr{G}, P[0] \models \varphi$.*
- *For a path P and path formulas φ, ψ, we have $\mathscr{G}, P \models \varphi \mathsf{U} \psi$ iff there is some $i \geq 0$ such that (i) $\mathscr{G}, P[i, \infty] \models \psi$, and (ii) for all $j < i$, we have that $\mathscr{G}, P[j, \infty] \models \varphi$.*
- *For a path P and path formulas φ, ψ, we have $\mathscr{G}, P \models \varphi \mathsf{R} \psi$ iff for all $i \geq 0$, (i) $\mathscr{G}, P[i, \infty] \models \psi$, or (ii) there is some $j \leq i$ such that $\mathscr{G}, P[j, \infty] \models \varphi$.*

The abbreviations $\Diamond\varphi$ for $true\mathsf{U}\varphi$ ("φ is true eventually") and $\Box\varphi$ for $\neg\Diamond\neg\varphi$ ("φ is always true") are often used. Note that by construction, $\neg\langle\langle A\rangle\rangle^{\geq\alpha}\neg\varphi$ is equivalent to $[[A]]^{>1-\alpha}\varphi$, and $\langle\langle A\rangle\rangle^{\leq\alpha}\varphi$ is equivalent to $\langle\langle A\rangle\rangle^{\geq 1-\alpha}\neg\varphi$ (similar equivalences hold for $>$ and $<$). Analogously, U is the dual operator of R.

Using the above dualities and classic results from game theory [Kuh53, BL69], one can show[2] that there is no need to consider mixed strategies for the principals and that we only need to study formulas in [[.]]*-free positive normal form*: In these formulas no [[.]], $\langle\langle . \rangle\rangle^{\leq \alpha}$, or $\langle\langle . \rangle\rangle^{< \alpha}$ appears, and negation is allowed only immediately in front of propositional variables. In the remainder of the paper we only talk about such formulas.

4.2 Game Structures for Protocol Analysis

We now define the PGS induced by a protocol and our symbolic protocol model. This PGS canonically represents the states and actions described in Section 3.3. The set of propositional variables used in the PGS allows formulas to reason about all relevant properties of a protocol run.

Definition 4.5 (protocol game structure). *Let* $Pr = (\Pi_1, \ldots, \Pi_k, \mathscr{I}_0)$ *be a* k*-roles protocol. protocol. The* probabilistic game structure (PGS) for Pr *is* $\mathscr{G}_{Pr} = (PR, Q, \Delta, \delta, \Pi, PV)$ *where*

- $PR = \{1, \ldots, k, \mathscr{A}, \mathscr{S}\}$,
- Q *is the set of global states of* Pr,
- *the set* Δ *of moves is as below,*
- PV *contains a variable* x_v *for each local state* v *of a rule in* $\Pi_1, \ldots, \Pi_k$, *a variable* c_a *for each principal* $a \in Ids$, *and a variable* a_d *for each* $d \in \{1, \ldots, k, \mathscr{S}, \mathscr{A}\}$,
- *for a variable* x_v, $\Pi(x_v)$ *is the set of all global states in which the (uniquely determined) role containing the state* v *is in the state* v*; for a variable* c_a, $\Pi(c_a)$ *contains all states in which* a *is corrupted; and for a variable* a_d, $\Pi(a_d)$ *contains all states* q *with* $Pr(q) = d$.

For a state $q = (a, \sigma, v_1, \ldots, v_k, \mathscr{I}, C)$, $Pr(q) = a$, *this principal's moves and consequences of the moves are defined exactly as in the execution of a symbolic protocol (see Section 3.3), except for the following cases:*

1. *If* $a \in \{1, \ldots, k\}$, *and its current local state is randomized, then*
   - *the principal* a *has a single move available in* q,
   - *the outcome of* q *specified by* δ *results from choosing each possible successor with the corresponding probability from the protocol.*
2. *If* $a = \mathscr{K}$, *then let* $Pr(q) = \mathscr{S}$, *and there is exactly one available move, which results in the state* $(\mathscr{A}, \emptyset, r_1, \ldots, r_k, \mathscr{I}_0, \{\mathscr{A}\}, \epsilon)$.

With q^0_{Pr}, we denote the initial state of Pr in $\mathscr{G}_{Pr}$. A pATL**-formula for* Pr is a pATL*-formula using only principals and propositional variables from $\mathscr{G}_{Pr}$.

5 The Computational Model

Our computational model is fairly standard (see, for instance, [BR93]), we only mention the key points. We use probabilistic polynomial-time (per activation)

[2] Note that our model ensures that games are sequential, and all players have complete information.

interactive Turing machines where each pair of machines shares a communication tape. The active machines are: (i) for each protocol role a *principal machine*, simulating the protocol role in the obvious way, (ii) an *adversary machine*, which is a probabilistic polynomial-time algorithm that plays the same role as the adversary in the symbolic model, and (iii) a *scheduler* controlling activation of principals as well as the adversary in the same way as in the symbolic model. We augment this by so-called *strategy machines*: When a principal or the scheduler makes a strategic decision in the protocol run (when it has more than one choice about an action to perform), it accesses its strategy machine to determine the action it follows. Strategy machines have access to the entire configuration of the protocol, and the adversary is informed of any strategic or randomized decision the principal makes.

5.1 Computational Protocol Execution

A computational protocol run is described by the following experiment, where, as usual, η denotes the security parameter and 1^η is the input to the experiment.

1. *Key Generation and machine initialization.* For each identity $a \in Ids$, generate private and public keys and distribute accordingly. Initialize machines for every role $1, \ldots, k$, $\mathscr{A}$, and $\mathscr{S}$, and strategy machines for each role $1, \ldots, k$ and $\mathscr{S}$.
2. *Corruption.* The adversary prints a set C of identities and receives the private key of every $a \in C$.
3. *Protocol Run.* The scheduler $\mathscr{S}$ is activated and may activate principals or the adversary, according to the protocol description. After the adversary terminates, the scheduler may continue to activate principals.

A *computational state* q of Pr consists of the configurations of all involved Turing machines. The set of corrupted players in a computational state is defined canonically. With $\mathscr{C}^\eta_{Pr}$, we denote the computational system running with security parameter η. With q^η_{init}, we describe the initial state of $\mathscr{C}^\eta_{Pr}$. We do not make the machine $\mathscr{A}$ explicit in the notation, as it will always be clear from the context: The adversary machine is never changed during a protocol run.

To model that principals may change their strategy during a protocol run, we allow the set of running strategy machines to change during the execution of the protocol.

A *computational path* of Pr is a sequence P_c of computational states. For a computational state q_c, a set S_A of strategy machines for all principals in $\{1, \ldots, k, \mathscr{S}\}$ and a pATL*-formula φ, by $\mathrm{prob}_{q_c, S_A, \varphi}(P_c)$, we denote the probability that the computation follows the path P_c, when the strategy machines S_A are used and the formula φ is given to the adversary as input (see below). Our complete protocol model [ASW09] ensures that in both the symbolic and the computational model, there is only a bounded number of actions in each protocol run. The bound depends only on the protocol.

5.2 ATL Semantics in the Computational Model

We now define what it means for a protocol to "computationally satisfy" a pATL*-formula. A *strategy set* fixes the strategies used by the involved principals to achieve certain security goals.

Definition 5.1 (strategy set). *Let φ be a* pATL**-formula in* [[.]]*-free positive normal form. A* strategy set *for φ is a pair $(\mathscr{A}, S)$ of an adversary $\mathscr{A}$ and a function S such that, for each subformula $\psi = \langle\langle A\rangle\rangle^{\geq\alpha}\chi$ and each $a \in \{1, \ldots, k, \mathscr{S}\}$, $S(a, \psi)$ is a strategy machine for a.*

We often write $S(\psi)$ for the set $\{S(a, \psi) \mid a \in \{1, \ldots, k, \mathscr{S}\}\}$. We now define what it means for a pATL*-formula to be computationally satisfied by a protocol and a pre-selected strategy set. The question which strategies are executed in a protocol run will be addressed later.

The following definition is straightforward, except for the case $\psi = \langle\langle A\rangle\rangle^{\geq\alpha}\chi$. In this case, the principals in A "switch" to their strategy machines specified by the strategy set for achieving the formula ψ. Additionally, the adversary gets "informed" of the current security goal that is to be reached (i.e., the adversary is handed ψ as input). In the case that $\mathscr{A} \in A$, this is necessary, since we want to evaluate the adversary's strategy to achieve the formula ψ, hence we need to make sure that the adversary indeed follows that strategy. In the case that $\mathscr{A} \notin A$, we want to evaluate the adversary's strategy *against* the formula ψ (which the coalition A tries to make true), and to ensure that the adversary actively tries to make ψ false, we require that it is informed of this "goal" attempted by the coalition A.

Definition 5.2 (computational pATL* **semantics).** *Let Pr be a k-roles protocol, let φ be a* pATL**-formula for Pr in* [[.]]*-free positive normal form, let $St = (\mathscr{A}, S)$ be a strategy set for φ, let q_c be a computational state of Pr, let P_c be a computational path of Pr, and let ψ be a subformula of φ.*

- *If $\psi = a_i$, then $\mathscr{C}^{\eta}_{Pr}, St, q_c \models \psi$ iff in q_c, i is activated next.*
- *If $\psi = c_a$, then $\mathscr{C}^{\eta}_{Pr}, St, q_c \models \psi$ iff a is corrupted in q.*
- *If $\psi = x_v$, for a variable x_v, then $\mathscr{C}^{\eta}_{Pr}, St, q_c \models \psi$ iff in q_c, the protocol rule containing the state v is in the local state v.*
- *Boolean connectives are dealt with as usual.*
- *If $\psi = \langle\langle A\rangle\rangle^{\geq\alpha}\chi$, then $\mathscr{C}^{\eta}_{Pr}, St, q_c \models \psi$ iff* $\sum_{P_c\colon \mathscr{C}^{\eta}_{Pr}, St, P_c \models \chi} prob_{q_c, S(\psi), \psi}(P_c) \geq \alpha$.
- *If ψ is a state formula, then $\mathscr{C}^{\eta}_{Pr}, St, P_c \models \psi$ iff $\mathscr{C}^{\eta}_{Pr}, St, P_c[0] \models \psi$.*
- *If $\psi = \chi \mathsf{U} \phi$, then $\mathscr{C}^{\eta}_{Pr}, St, P_c \models \psi$ iff there is an $i \geq 0$ with $\mathscr{C}^{\eta}_{Pr}, St, P_c[i, \infty] \models \phi$ and $\mathscr{C}^{\eta}_{Pr}, St, P_c[j, \infty] \models \chi$ for all $j < i$.*
- *If $\psi = \chi \mathsf{R} \phi$, then $\mathscr{C}^{\eta}_{Pr}, St, P_c \models \varphi, v$ iff for all $i \geq 0$ $\mathscr{C}^{\eta}_{Pr}, St, P_c[i, \infty] \models \phi$, or $\mathscr{C}^{\eta}_{Pr}, St, P_c[j, \infty] \models \chi$ for some $j \leq i$.*

We now define which strategy machines will be running in the execution of a protocol. Let φ be a pATL*-formula for a protocol Pr and $\psi = \langle\langle A\rangle\rangle^{\geq\alpha}\chi$ a subformula of φ. A principal $i \in \{1, \ldots, k, \mathscr{S}\}$ is *universally quantified (existentially*

quantified) in ψ if $i \notin A$ ($i \in A$). A *strategy enumeration* fixes the values for the quantified strategies in a formula.

Definition 5.3 (strategy enumeration). *Let φ be a* pATL**-formula. A* universal (existential) strategy enumeration for φ *is a function f such that for each pair (ψ, i) where ψ is a subformula of φ and i is universally (existentially) quantified in ψ, $f(\psi, i)$ is a strategy machine for i.*

For a pair of universal and existential strategy enumerations and an adversary $\mathscr{A}$, the strategy set running in the system is the pair $(\mathscr{A}, U \cup E)$ (note that U and E have disjoint domains).

6 Computational Soundness

Our intention is to prove that the computational model "satisfies the same formulas" as the symbolic one. However, in the computational model we cannot completely rule out that the adversary might "break" the protocol, since with some (low) probability, signatures may be forged, random numbers selected by different parties may coincide, etc. Therefore we consider "relaxed" versions of the involved pATL*-formulas in the computational setting.

Definition 6.1 (ϵ-tolerant formulas). *Let φ be a* pATL**-formula in* $[[.]]$*-free positive normal form and let $\epsilon > 0$. Then φ^ϵ, the* ϵ-tolerant version of φ, *is obtained from φ by replacing, in each outermost $\langle\langle . \rangle\rangle$-operator, every occurrence of a probability bound α with $\alpha - \epsilon$.*

Another difference between the symbolic and computational model is that the symbolic model allows quantification over strategies "during a protocol execution," which leads to problems in the computational model: A machine chosen in an protocol run with security parameter η could have a hard-coded table of prime factorizations of all integers with bit length up to η, compromising the security of signature schemes relying on the hardness of the factorization problem. Hence we fix the set of available strategies before the protocol is actually run. This is also a very natural requirement, as intuitively, a "strategy" should be a plan that works for every security parameter. The existential and universal quantification over strategies now become quantifications over strategy enumerations, and the quantification happens before a protocol run. A special role is played by the adversary: Recall that we only consider a single adversary machine, which does not change during a protocol run. We therefore disallow formulas to quantify the adversary both existentially and universally—this leads to a natural subclass of formulas, as a security property is usually phrased in describing what the adversary *can* or *cannot* do. Formally, a pATL*-formula for Pr in $[[.]]$-free positive normal is $\mathscr{A}$-positive ($\mathscr{A}$-negative), if for all subformulas $\langle\langle A \rangle\rangle^{\geq \alpha} \chi$, we have $\mathscr{A} \in A$ ($\mathscr{A} \notin A$). A formula is $\mathscr{A}$-monotone if it is $\mathscr{A}$-positive or $\mathscr{A}$-negative. Note that in [KKT07], similarly defined monotone formulas are studied to obtain a decidability result. Except for the aforementioned differences, both models satisfy the same formulas:

Theorem 6.2 (computational soundness). *Assume that the signature scheme is resistant against existential forgery. Let Pr be a protocol and let φ be an $\mathscr{A}$-positive ($\mathscr{A}$-negative)* pATL**-formula such that $\mathscr{G}_{Pr}, q^0_{Pr} \models \varphi$. Then there exists an existential strategy enumeration E and an adversary machine $\mathscr{A}$ (for all adversary machines $\mathscr{A}$) such that for every universal strategy enumeration U, if $St = (\mathscr{A}, E \cup U)$, then there is a negligible function $\epsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that for all security parameters η, $\mathscr{C}^\eta_{Pr}, St, q^\eta_{init} \models \varphi^{\epsilon(\eta)}$.*

The theorem states that for any security goal satisfied in the symbolic model, there are strategy machines achieving the goal in the computational model: One can implement algorithms for the protocol roles such that when given a "command" to achieve a specific protocol situation, they can compute the corresponding actions (in this case the "command" is the subformula stating the goal to be reached).

7 Application to Contract Signing and the Gradual Commitment Protocol

In the following, let Pr be a contract-signing protocol with same setup as GCP, i.e., the roles in the protocol are an originator O, a responder R, a trusted third party T, and a buffer principal B (securely relaying messages between T, O, and R). For analyzing the protocol, we treat one of the signers O and R as dishonest. Formally, this means we choose $X \in \{O, R\}$ as honest, and denote the dishonest signer as $\overline{X}$. Since we assume that $\overline{X}$ works together with the adversary, for the analysis we treat Pr as a 3-roles protocol: The honest signer X, the trusted third party T, and a buffer principal B relaying messages from X to T and vice versa. The role B is assumed to be network-ignoring, i.e., only X and T have write access to the buffer. In the following, we use X, T, and B as principals in the protocol instead of numbers. Note that in the game structure for Pr, in addition to the above-mentioned roles, there are principals $\mathscr{A}$ and $\mathscr{S}$. In order to be able to generate messages signed by $\overline{X}$ in the protocol run, the first move of the adversary is to corrupt $\overline{X}$.

To formally define unbalance, let φ_{nc} express that $\mathscr{A}$ did not corrupt X or T, let $\varphi_{\mathscr{A}\mathsf{c}}$ ($\varphi_{X\mathsf{c}}$) be true if $\mathscr{A}$ (X) has a valid contract, and let φ_{dl} indicate that B has delivered all messages. These can easily be expressed given the available propositional variables. We now formally state the goals the adversary is trying to reach. Consider φ_{abr} defined by $\varphi_{\mathsf{abr}} = \Box\,(\varphi_{\mathsf{nc}} \wedge \Diamond\varphi_{\mathsf{dl}} \wedge \neg\varphi_{X\mathsf{c}})$. This formula describes all protocol runs in which X and T never get corrupted, every request written into a buffer principal is eventually delivered, and X never obtains a contract. The formula for resolving the protocol is $\varphi_{\mathsf{res}} = \varphi_{\mathsf{nc}} \wedge \varphi_{\mathsf{dl}} \wedge \varphi_{\mathscr{A}\mathsf{c}}$, i.e., a state is resolved if the adversary has a contract, the buffer has delivered all messages, and neither X nor T have been corrupted.

Definition 7.1 (balance). *A contract signing protocol Pr is symbolically (p_a, p_r)-unbalanced against X,*

$$\mathscr{G}_{Pr}, q^0_{Pr} \models \langle\langle \mathscr{A}, T, B, X, \mathscr{S} \rangle\rangle^{>0} \Diamond \big(\langle\langle \mathscr{A}, \mathscr{S}, B \rangle\rangle^{\geq p_a} \Diamond \varphi_{\mathsf{abr}} \wedge \langle\langle \mathscr{A}, \mathscr{S}, B \rangle\rangle^{\geq p_r} \Diamond \varphi_{\mathsf{res}} \big)\,.$$

This definition naturally captures the previously mentioned definition of unbalance, that a state is reachable (expressed by the first $\langle\langle . \rangle\rangle$-operator) where the adversary has strategies with the relevant success probabilities (expressed by the remaining $\langle\langle . \rangle\rangle$-operators). With "unbalanced" we mean "unbalanced for R or O," and use "balanced" for "not unbalanced." Our main result on GCP is:

Theorem 7.2 (balance of GCP**).** *For all $n \geq 2$,* GCP_n *is (p_a, p_r)-balanced for all $p_a + p_r \geq 1 + (1/n)$.*

We illustrate our model and soundness result by comparing it with [CKW07]. If the computational definition given in [CKW07] is adapted to the setting with explicit probabilities, it reads as follows.[3] A protocol is computationally (p_a, p_r)-unbalanced against X, if there is an adversary A, a strategy machine S_B for B, a strategy machine $S_{\mathscr{S}}$ for $\mathscr{S}$, a set of strategy machines S_1 for $\{T, X\}$ such that for all sets of strategy machines S_2 for $\{T, X\}$ the following experiment, on input 1^η, returns 1 with non-negligible probability:

1. *(Key Generation)* Generate keys for all involved identities.
2. *(Corruption)* The adversary prints a list of identities and receives their private keys.
3. *(Reach unbalanced state)* Simulate the protocol execution with $\mathscr{A}$ and strategy machines S_B, S_1, and $S_{\mathscr{S}}$ until the adversary prints `unbalanced` on a special tape.
4. *(Verify unbalancedness)* Start two copies of the experiment with $\mathscr{A}$, strategy machines S_B, S_2, and $S_{\mathscr{S}}$ starting in the current state:
   (a) All strategy machines and the adversary get `abort` as input. The sub-experiment is successful, if from here, the probability that the contract signing is aborted is at least p_a.
   (b) All strategy machines and the adversary get `resolve` as input. The sub-experiment is successful, if from here, the probability that the contract signing is resolved is at least p_r.

   The entire experiment is successful if and only if both sub-experiments are successful.

Here the signing is aborted (resolved) if X has not received a contract ($\mathscr{A}$ did receive a contract), the protocol is in a final state, and neither X not T have been corrupted. One can easily show that their definition exactly corresponds to the guarantees implied by our symbolic definition of balance above—with Theorem 6.2, we obtain the following corollary:

Corollary 7.3. *A contract signing protocol is computationally (p_a, p_r)-unbalanced if and only if it is symbolically (p_a, p_r)-unbalanced.*

For GCP, we conclude

Corollary 7.4. *If $p_a + p_r \geq 1 + (1/n)$,* GCP_n *is computationally (p_a, p_r)-balanced.*

[3] Note that we slightly simplified their definition in omitting their polynomial-time "challenge"-function and fair scheduling—it is clear that this function can be computed in polynomial time in our setting. Also, fairness of scheduling is implicit in our model—hence we can regard the scheduler as working together with the adversary.

8 Conclusion

We have suggested an optimistic contract-signing protocol that remains balanced even when the adversary has control over the order in which messages are received by the TTP. We have introduced a formal model for analysis of probabilistic protocols and proved its soundness with respect to computational security, implying that our protocol is balanced in the sense of [CKW07].

An obvious question suggested by the current work is the extension of our results to additional cryptographic primitives, most importantly encryption. Another interesting issue is to consider a setting in which the adversary and the principals only have access to the information available to it from the observed network traffic. We believe that applying a variant of ATL that deals with incomplete information will help in this situation.

References

[AHK02] Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM 49(5), 672–713 (2002)

[AR02] Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). Journal of Cryptology 15(2), 103–127 (2002)

[ASW98] Asokan, N., Shoup, V., Waidner, M.: Asynchronous protocols for optimistic fair exchange. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, pp. 86–99. IEEE Computer Society Press, Los Alamitos (1998)

[ASW09] Aizatulin, M., Schnoor, H., Wilke, T.: Computationally sound analysis of a probabilistic contract signing protocol. Technical Report 0911, Institut für Informatik, Christian-Albrechts-Universität zu Kiel (2009)

[BDD+06] Backes, M., Datta, A., Derek, A., Mitchell, J.C., Turuani, M.: Compositional analysis of contract-signing protocols. Theoretical Computer Science 367(1-2), 33–56 (2006)

[BL69] Buchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Transactions of the American Mathematical Society 138, 295–311 (1969)

[BOGMR90] Ben-Or, M., Goldreich, O., Micali, S., Rivest, R.L.: Fair protocol for signing contracts. IEEE Transactions on Information Theory 36(1), 40–46 (1990)

[BR93] Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)

[CH06] Canetti, R., Herzog, J.: Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 380–403. Springer, Heidelberg (2006)

[CKKW06] Cortier, V., Kremer, S., Küsters, R., Warinschi, B.: Computationally sound symbolic secrecy in the presence of hash functions. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 176–187. Springer, Heidelberg (2006)

[CKS01] Chadha, R., Kanovich, M.I., Scedrov, A.: Inductive methods and contract-signing protocols. In: ACM Conference on Computer and Communications Security, pp. 176–185 (2001)

[CKW07] Cortier, V., Küsters, R., Warinschi, B.: A cryptographic model for branching time security properties - the case of contract signing protocols. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 422–437. Springer, Heidelberg (2007)

[CL07] Chen, T., Lu, J.: Probabilistic alternating-time temporal logic and model checking algorithm. In: Lei, J. (ed.) FSKD (2), pp. 35–39. IEEE Computer Society Press, Los Alamitos (2007)

[CMSS05] Chadha, R., Mitchell, J.C., Scedrov, A., Shmatikov, V.: Contract signing, optimism, and advantage. Journal of Logic and Algebraic Programming 64(2), 189–218 (2005)

[DY83] Dolev, D., Yao, A.C.-C.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–207 (1983)

[GJM99] Garay, J.A., Jakobsson, M., MacKenzie, P.D.: Abuse-free optimistic contract signing. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 449–466. Springer, Heidelberg (1999)

[KKT07] Kähler, D., Küsters, R., Truderung, T.: Infinite state AMC-model checking for cryptographic protocols. In: LICS, pp. 181–192. IEEE Computer Society Press, Los Alamitos (2007)

[KKW05] Kähler, D., Küsters, R., Wilke, T.: Deciding properties of contract-signing protocols. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 158–169. Springer, Heidelberg (2005)

[KR02] Kremer, S., Raskin, J.-F.: Game analysis of abuse-free contract signing. In: CSFW. IEEE Computer Society Press, Los Alamitos (2002)

[KR03] Kremer, S., Raskin, J.-F.: A game-based verification of non-repudiation and fair exchange protocols. Journal of Computer Security 11(3), 399–430 (2003)

[Kuh53] Kuhn, H.W.: Extensive games and the problem of information. Annals of Mathematics Studies 28, 193–216 (1953)

[LM05] Lakhnech, Y., Mazaré, L.: Computationally sound verification of security protocols using Diffie-Hellman exponentiation. Technical report, Verimag (2005)

[PG99] Pagnia, H., Gartner, F.C.: On the impossibility of fair exchange without a trusted third party. Technical report, Darmstadt University of Technology (1999)