INSTITUT FÜR INFORMATIK

Computationally Sound Analysis of a Probabilistic Contract Signing Protocol

Mihhail Aizatulin
Henning Schnoor
Thomas Wilke

Bericht Nr. 0911
April 2009

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Computationally Sound Analysis of a Probabilistic Contract Signing Protocol

Mihhail Aizatulin
Henning Schnoor
Thomas Wilke

Technical Report Nr. 0911
April 2009

mai@informatik.uni-kiel.de,
{schnoor—wilke}@ti.informatik.uni-kiel.de

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), see, e. g., [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.

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. We propose a parametrized protocol that achieves the following *probabilistic* notion of balance under the above assumptions: 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. 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, the symbolic and the computational one. 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) as well.

Related Work. In [KKW05], a symbolic definition of balance for CPSs 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 fair contract-signing protocol 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, so that the sum of the two probabilities from above, $p_r + p_a$, is 2 in [BOGMR90] (as opposed to $1 + \frac{1}{n}$ as in our protocol).

2 The Gradual Commitment Protocol (GCP)

In this section, we present our contract signing protocol and state its key properties in an informal fashion. The protocol is based on [Aiz08].

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). For a probabilistic setting, we define the following 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 reasonable protocol there is a state in which a party has obtained a contract, which means the unbalance of such a state is at least 0, (ii) for an unbalanced protocol without randomness there is a state and strategies such that $\epsilon_a = 1$ and $\epsilon_r = 1$, that is, 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. As a consequence, 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^i_X for $X \in \{O, R\}$ and $i \in \{1, \ldots, n+1\}$. First, the *originator*, O, sends CMT^1_O, then the *responder*, R, sends CMT^1_R, and so on, the last commitment being CMT^{n+1}_R. The pair of the last two commitments, $\langle \mathsf{CMT}^{n+1}_O, \mathsf{CMT}^{n+1}_R \rangle$, is a *valid contract*. The commitments are defined by

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

where text is the document the two parties want to sign, sid is a session identifier, 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^i_O with $i \in \{1, \ldots, n\}$ can be used by R to form a resolve request, RR^i_R, addressed to the TTP. Similarly, a commitment CMT^i_R can be used by O to form a resolve request, RR^i_O. The precise format is

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

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): $\mathsf{RR}^0_O = [\mathsf{CMT}^1_O, \mathsf{abort}]_O$, where abort is a fixed token.

Possible replies of T to $m = \mathsf{RR}^i_X$ are the *replacement contract*, denoted $\mathsf{R\text{-}CTR}^i_X$, and defined by $\mathsf{R\text{-}CTR}^i_X = [m]_T$, which is recognised as a *valid contract*, or a *rejection*, $\mathsf{RT}^i_X = [m, \mathsf{rejected}]_T$, 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: $\mathsf{RR}^i_R < \mathsf{RR}^j_O$ if $i < j$ and $\mathsf{RR}^i_O < \mathsf{RR}^j_R$ 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 $m = \mathsf{RR}^i_X$, 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.

We will prove (see Theorem 8.3 and Corollary 8.5) that GCP is timely and its unbalance is $1/n$, even if the adversary may delay messages sent over the secure channel. We mention without proof that, in addition, the protocol is TTP-accountable and TTP-secure, see [GJM99] for definitions.

3 The Symbolic Protocol Model

In this section we describe our symbolic protocol model, which the remainder of the paper is based on.

3.1 Variables, Terms, and Messages

We fix a finite set *Ids* of *identities* and a number k of *entities* or *roles* that participate in a protocol session. The adversary is denoted by $\mathscr{A}$. For each identity $A \in \mathit{Ids} \cup \{\mathscr{A}\}$, and for each $i \in \mathbb{N}$,

there is a *message nonce* $N_m^{A,i}$, and a randomization nonce $N_r^{A,i}$. We also fix a finite set Cons of constants.

To allow syntactic checks on incoming terms, we use variables that have a type and a depth restriction. Let *types* be the finite set containing the elements id *(identity)*, mnonce *(message nonce)*, rnonce *(randomization nonce)*, pair *(pair)*, const *(constant)*, empty *(empty)*, and sig *(signature)*. Let $\mathscr{V}$ be a finite set of variables, and let $type\colon \mathscr{V} \to types$ and $maxdepth\colon \mathscr{V} \to \mathbb{N}$ be functions assigning a type and a maximal depth to each variable.

Randomization nonces are used to explicitly capture randomness used by signature schemes. Note that [CHW07] proves that for many cases, explicitly capturing randomness in the symbolic model is not necessary. Their work, however, only covers trace properties, and protocols where equality tests are not permitted. We define terms as usual, and define their type and depth in the obvious way:

Definition 3.1 (terms, types, depth).
- *The empty term* ϵ *is a term with* $depth(\epsilon) = 0$ *and* $type(\epsilon) = \text{empty}$,
- *an identity* $A \in Ids$ *is a term with* $depth(A) = 0$ *and* $type(A) = \text{id}$,
- *a message nonce* N *is a term with* $depth(N) = 0$ *and* $type(N) = \text{mnonce}$,
- *a randomization nonce* N *is a term with* $depth(N) = 0$ *and* $type(N) = \text{rnonce}$,
- *a constant* $c \in \mathsf{Cons}$ *is a term with* $depth(c) = 0$ *and* $type(c) = \text{const}$,
- *if* t_1 *and* t_2 *are terms, then* $\langle t_1, t_2\rangle$ *is a term with* $depth(\langle t_1, t_2\rangle) = \max(depth(t_1), depth(t_2)) + 1$ *and* $type(\langle t_1, t_2\rangle) = \text{pair}$,
- *if* $x \in \mathscr{V}$, *then* x *is a term with* $depth(x) = 0$ *(and of type* $type(x)$*)*,
- *if* t' *is a term,* $A \in Ids$ *is an identity, and* N *is a term with* $type(N) = \text{rnonce}$, *then* $\mathsf{sig}(A, N, t')$ *is a term with* $depth(\mathsf{sig}(A, N, t')) = depth(t') + 1$ *and* $type(\mathsf{sig}(A, N, t')) = \text{sig}$.

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. We demand that $dom(\sigma)$ is finite for every substitution. 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 $\sigma(x)$.

Definition 3.2 (matching substitution). *For a term* t, *substitutions* σ *and* σ', *and a message* m, *we say that* m matches with t and σ via σ' *if* $t\sigma' = m$, $\sigma'(x) = \sigma(x)$ *for every* $x \in dom(\sigma)$, *and* $type(x) = type(\sigma'(x))$ *as well as* $depth(\sigma'(x)) \leq maxdepth(x)$ *for every* $x \in dom(\sigma')$.

We now define how the adversary can derive messages from its current knowledge. The following definition captures that the adversary can perform the canonical operations on terms to construct new messages. In order for our model to be computationally sound, we allow the adversary to derive all possible constants (acknowledging that these might be hard-coded into an algorithm). Although it would be compatible with usual security notions of signature schemes to allow the adversary to learn the randomness used in the construction of a signature, we do not model this explicitly—our protocols will ensure that if a random string is used for the construction of a signature, then this string will not be used for any other purpose during a protocol run, hence this knowledge would be of no use for the adversary: Instead of re-using a randomization received in the protocol run, the adversary might as well use one of its own nonces, and achieve the same effect.

Definition 3.3 (derivable messages). *Let* $\mathscr{I}$ *be a set of messages and let* C *be a set of identities. The set* $d(\mathscr{I}, C)$ *(the messages* derivable *from* $\mathscr{I}$ *with corrupted* C*) is the smallest set satisfying*

– $\mathsf{Cons} \cup \mathscr{I} \cup \mathit{Ids} \subseteq d(\mathscr{I}, C)$,
– *for all* $i \in \mathbb{N}$, $N_m^{\mathscr{A},i} \in d(\mathscr{I}, C)$,
– $\langle t_1, t_2 \rangle \in d(\mathscr{I}, C)$ *if and only if* $t_1, t_2 \in d(\mathscr{I}, C)$,
– *if* $\mathsf{sig}(A, N, t) \in d(\mathscr{I}, C)$, *then* $t \in d(\mathscr{I}, C)$,
– *if* $A \in C$, $t \in d(\mathscr{I}, C)$, *and* $i \in \mathbb{N}$, *then* $\mathsf{sig}(A, N_r^{\mathscr{A},i}, t) \in d(\mathscr{I}, C)$.

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 reply 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 so-called *role tree* and its vertices are the *(local) states* of the role. We demand that every variable occurring on the right-hand side of $\ell(v, v')$ occurs on the left-hand side of a label of an edge on the path from v_0 to v'. We additionally require that for a role Π, there is at most one identity A such that a (message or randomization) nonce of the form $N_m^{A,i}$ or $N_r^{A,i}$ appears in the receive-send rules. We say that A is the *identity of* Π.

For technical reasons, we only allow *randomized local states* and *strategic local states*, defined as follows. 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 the outgoing edges are all 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. (One could also express this property for each protocol rule individually, for ease of notation we fix this behaviour for the entire role.)

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.*

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 nonces when signing different terms. The straight-forward formalizations of these requirements can be found in Appendix A.

3.3 Symbolic Protocol Execution

We define how a protocol $Pr = (\Pi_1, \ldots, \Pi_k, \mathscr{I}_0)$ is executed in our model. We first use renaming to ensure that different roles use disjoint sets of variables and local states. In the following, we refer to all of the involved labeling functions as ℓ (after renaming, the individual labelling functions have disjoint domains). 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, $C \subseteq \mathit{Ids} \cup \{\mathscr{A}\}$, 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 determined 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 $(\mathscr{S}, \emptyset, v_0^1, \ldots, v_0^k, \mathscr{I}_0, C', \epsilon)$. These are the *only* successor states of q.
This expresses that after key generation, the adversary may corrupt identities, but may not perform another action without the scheduler being active before.

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 $(i, \sigma, v_1, \ldots, v_k, \mathscr{I}, C, m')$.
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. Assume that $a = i$ for some $i \in \{1, \ldots, k\}$. For each successor v_i' of v_i such that $\ell(v_i, v_i')$ contains the rule $(r \rightarrow_d s)$ or $r \longrightarrow_d^p s$, and there is a substitution σ' such that s matches with r and σ via σ', there is a successor state $(d, \sigma', v_1, \ldots, v_{i-1}, v_i', v_{i+1}, \ldots, v_k, \mathscr{I}, C, s\sigma')$ of q, if $d \neq \mathscr{A}$ 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 message sent by a role is activated next to process the message. If the message is empty and addressed to the adversary, or if the receiver cannot process the incoming message, control is returned to the scheduler (and m is ignored). Note that if the rule is a randomized rule (i.e., of the form $r \longrightarrow_d^p s$), then the matching-requirement is fulfilled trivially, as by the protocol model, it follows that $r = \epsilon$.

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 at any time, including the adversary.

The key generation step does not have a function in the symbolic protocol execution. We include it here because in the later computational model such a step is obviously necessary. Note that the above structure of the protocol execution allows certain "dead end loops:" When the adversary delivers a message m to a role that cannot receive m in the current state (due to lack of a matching receive-send rule), then the scheduler is activated next. The scheduler then can activate the adversary again; this leads to an infinite cycle where essentially, nothing happens. We therefore require the following: If the adversary did not produce any output that was received by a principal (in the sense that the principal changed local state upon processing the adversary's message), then from the resulting state, the adversary may not be activated. (Formally, the states are extended with a flag that indicates whether the adversary has delivered a "useless" message. Since this flag is not used elsewhere, we do not make it explicit in our state description above). Similarly, we disallow the scheduler to activate a principal machine with incoming message ϵ if the principal in the current state does not have a $\epsilon \longrightarrow_d s$-rule available (these rules are used to explicitely express that a principal is ready to act without receiving an incoming message).

This requirement results in a certain fairness condition that the scheduler needs to satisfy (see remarks at the end of Section 5.2).

The above definition of protocols and protocol execution 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 direct links,

completely disabling the adversary from taking any relevant action in a protocol run. However, such a protocol is clearly unrealistic. The reason why we allow these direct links is the observation that many security goals cannot be achieved when no guarantees can be made about the delivery of messages. An example is optimistic contract signing, which 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. This is not unrealistic: Such a party might be reachable via telephone, papermail or other fallback methods in the case of a network problem (where of course, message delivery might take a long time).

The most realistic way to express this situation in our model is to introduce a special party which only serves as a "buffer" between other principals and the secure channel, and whose only function is to relay received messages to the TTP. Hence all other honest players will be able to directly send messages to the buffer, which may directly send messages to TTP. Since we model the buffer as a principal on its own, we can express statements about strategies of buffers. Hence we can use pATL*-formulas to talk about the possibly allowed behaviour of these parties in detail, rather than fixing the way these channels work once and for all in our model. By using different specifications for these "buffer principals," one can also specify whether messages may be delivered out of sequence, and other details. Note that for a realistic protocol model, these channels should be specified such that when receiving an incoming term, they do not immediately forward it to the intended recipient, but use a $r \rightarrow_{\mathscr{A}} r$-rule to add the received term to the adversary's knowledge, and return control to the scheduler (if one wants to model buffers that conceal messages from the adversary, one can use a $r \rightarrow_{\mathscr{A}} \epsilon$ instruction instead, whose only result is storing the incoming message and activating the scheduler). Delivery-actions then should be ϵ-transitions, which can be initiated by the buffer-principal whenever it is activated by the scheduler.

However, our model also allows to specify (somewhat unrealistic) protocols where the principals can perform long sequences of distributed actions using their direct connections, without the adversary or the scheduler being able to interfere. The degree of "realism" of a security property proved in our model depends on the assumptions that the protocol makes about the communication model (i.e., the usage of direct links): A direct link used in the symbolic protocol description results in a shared tape between the corresponding machines in the symbolic model that the principals use to exchange their messages. For an example, of a protocol expressed in our model and the security guarantees obtained, see Sections 4 and 8.

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 5.2.

4 Modeling the Gradual Commitment Protocol

We now specify GCP, informally described in Section 2, within the framework developed thus far. There are two different scenarios in which we study the protocol, namely the one where the originator is honest while the responder is not and the one where the originator is dishonest while the responder is not. Both are very similar; we only model the first one.

Parameters and Basic Terms. The set of identities and the set of constants are defined by

$$Ids = \{O, R, T\} \ , \qquad \mathsf{Cons} = \{\mathsf{text}, \mathsf{abort}, \mathsf{rejected}, \underline{1}, \ldots, \underline{n+1}\} \ .$$

Here, the constants of the form $\underline{i}$ model the round numbers. The variables are denoted by x, y, and z, possibly decorated with sub and superscripts. The initial adversary knowledge, $\mathscr{I}_0$, is the empty set. Formally, the randomization nonces are of the form $N_r^{a,i}$. For notational convenience, we denote them by L, M, and N, possibly decorated with sub and superscripts. It will be obvious which role the nonces belong to.

All together there are 4 roles:

- role 1 is the originator, (by abuse of notation) also denoted by O,
- role 2 is the trusted third party, also denoted by T,
- role 3 is a buffer for a message from the originator to the TTP, denoted B_O^T, and
- role 4 is a buffer for a message from the TTP to the originator, denoted B_T^O.

All roles except for the buffers are network-accepting, the buffers are network-ignoring. The buffers are symmetric; we describe only B_O^T. Its role tree (V_B, E_B) is given by $V_B = \{b_0, b_1, b_2\}$ and $E_B = \{(b_0, b_1), (b_1, b_2)\}$, and the labeling function is defined by

$$l(b_0, b_1) = x \underset{\mathscr{A}}{\longrightarrow} \epsilon \ , \qquad\qquad l(b_1, b_2) = \epsilon \underset{T}{\longrightarrow} x \ .$$

Hence buffers do not deliver messages immediately, but only when then are activated (by the scheduler). Also, the adversary can read messages sent via buffers, but since they are network-ignoring, cannot write onto them. The terms that we use in our formal description are very similar to the description in Section 2, except for the following changes: (i) tuples are replaced by nested pairing, (ii) session id's are not modeled because we analyze a single session, and (iii) randomization nonces are made explicit. We let, for $i \in \{1, \ldots, n+1\}$ and $j \in \{1, \ldots, n\}$,

$$\mathsf{CMT}_X^i = \mathsf{sig}(O, x_X^i, \langle \mathsf{text}, \langle X, \langle R, \langle T, \underline{i} \rangle \rangle \rangle \rangle) \ , \qquad \mathsf{RR}_O^0 = \mathsf{sig}(O, y_O^0, \langle \mathsf{CMT}_O^1, \mathsf{abort} \rangle) \ ,$$
$$\mathsf{RR}_O^j = \mathsf{sig}(O, y_O^j, \langle \mathsf{CMT}_O^{j+1}, \mathsf{CMT}_R^j \rangle) \ , \qquad \mathsf{RR}_R^j = \mathsf{sig}(O, y_R^j, \langle \mathsf{CMT}_O^j, \mathsf{CMT}_R^j \rangle) \ .$$

Formal Model of the Originator. The underlying tree, (V_O, E_O), is defined by

$$V_O = \{s_1, \ldots, s_{n+3}, r_2, \ldots, r_{n+2}, a_2, \ldots, a_{n+2}, c_2, \ldots, c_{n+2}\} \ ,$$
$$E_O = \{(s_i, s_{i+1}) \colon 1 \le i \le n+3\} \cup \{(s_i, r_i), (r_i, c_i), (r_i, a_i) \colon 2 \le i \le n+2\} \ .$$

So s_1 is the root of the tree. The labeling is defined in what follows, where the first three equations describe how the originator works if nothing goes wrong:

$$l(s_1, s_2) = \epsilon \underset{\mathscr{A}}{\longrightarrow} \mathsf{CMT}_O^1[x_O^i / N_O^1] \ ,$$
$$l(s_i, s_{i+1}) = \mathsf{CMT}_R^{i-1} \underset{\mathscr{A}}{\longrightarrow} \mathsf{CMT}_O^i[x_O^i / N_O^i] \ ,$$
$$l(s_{n+2}, s_{n+3}) = \mathsf{CMT}_R^{n+1} \underset{\mathscr{A}}{\longrightarrow} \epsilon \ ,$$
$$l(s_2, r_2) = \epsilon \underset{B_O^T}{\longrightarrow} \mathsf{RR}_O^0[y_O^0 / M_O^0] \ ,$$
$$l(s_j, r_j) = \epsilon \underset{B_O^T}{\longrightarrow} \mathsf{RR}_O^{j-2}[y_O^{j-2} / M_O^{j-2}] \ ,$$
$$l(r_k, c_k) = \mathsf{sig}(T, z_O^{k-2}, \mathsf{RR}_O^{k-2})[y_O^{k-2} / M_O^{k-2}] \underset{\mathscr{A}}{\longrightarrow} \epsilon \ ,$$
$$l(r_k, a_k) = \mathsf{sig}(T, z_O^{k-2}, \langle \mathsf{RR}_O^{k-2}, \mathsf{rejected} \rangle)[y_O^{k-2} / M_0^{k-2}] \underset{\mathscr{A}}{\longrightarrow} \epsilon \ ,$$

where $i \in \{2, \ldots, n+1\}$, $j \in \{3, \ldots, n+2\}$, and $k \in \{2, \ldots, n+3\}$. The notation $[x/N]$ indicates a substitution which replaces all occurrences of x by N.

Formal Model of the TTP. We define the role of the TTP by induction on the structure of its role tree. The meta variable X stands for R or O. In some rules, a message will be sent to B_T^X. If $X = R$, then $B_T^X = \mathscr{A}$, because we model the dishonest responder by the adversary. The empty sequence is the root of the tree. For every pair (i, X), the root ϵ has one successor, (i, X), which itself has two successors, (i, X, res) and (i, X, rej), with edge labeling defined by

$$\begin{aligned} l(\epsilon, (i, X)) &= \mathsf{RR}_X^i \xrightarrow[T]{} \epsilon \ , \\ l((i, X), (i, X, res)) &= \epsilon \xrightarrow[B_T^X]{i/n} \mathsf{sig}(T, L_X^i, \mathsf{RR}_X^i)[y_X^i / M_X^i] \ , \\ l((i, X), (i, X, rej)) &= \epsilon \xrightarrow[B_T^X]{\frac{n-i}{n}} \mathsf{sig}(T, L_X^i, \langle \mathsf{RR}_X^i, \mathsf{rejected} \rangle)[y_X^i / M_X^i] \ . \end{aligned}$$

A vertex of the form (i, X, res) has $n + 1$ successors, namely (i, X, res, j) for $j \in \{1, \ldots, n + 1\}$, and the labeling of the corresponding edges is given by

$$l((i, X, res), (i, X, res, j)) = \mathsf{RR}_{\bar{X}}^j \xrightarrow[B_T^X]{} \mathsf{sig}(T, L_{\bar{X}}^j, \mathsf{RR}_{\bar{X}}^j)[y_{\bar{X}}^j / M_X^j] \ .$$

A vertex of the form (i, X, rej) has, for every j such that $i < j \leq n$ and $X = R$, or $i+1 < j \leq n$ and $X = O$, one successor of the form (i, X, rej, j), which, in turn, has two successors, (i, X, rej, j, res) and (i, X, rej, j, rej), where the labelings are determined by:

$$\begin{aligned} l((i, X, rej), (i, X, rej, j)) &= \mathsf{RR}_{\bar{X}}^j \xrightarrow[T]{} \epsilon \ , \\ l((i, X, rej, j), (i, X, rej, j, res)) &= \epsilon \xrightarrow[B_T^X]{j/n} \mathsf{sig}(T, L_{\bar{X}}^j, \mathsf{RR}_{\bar{X}}^j)[y_{\bar{X}}^j / M_X^j] \ , \\ l((i, X, rej, j), (i, X, rej, j, rej)) &= \epsilon \xrightarrow[B_T^X]{\frac{n-j}{n}} \mathsf{sig}(T, L_{\bar{X}}^j, \langle \mathsf{RR}_{\bar{X}}^j, \mathsf{rejected} \rangle)[y_{\bar{X}}^j / M_X^j] \ . \end{aligned}$$

Further, for every j such that $j \leq i$ and $X = R$, or $j \leq i + 1$ and $X = O$, the vertex has the successor (i, X, rej, j, rej):

$$l((i, X, rej), (i, X, rej, j, rej)) = \mathsf{RR}_{\bar{X}}^j \xrightarrow[B_T^X]{} \mathsf{sig}(T, L_{\bar{X}}^j, \langle \mathsf{RR}_{\bar{X}}^j, \mathsf{rejected} \rangle)[y_{\bar{X}}^j / M_X^j] \ .$$

Intuitively, if the TTP is in a state (i, X, rej, j, res), this means that the first resolve request that T received was RR_X^i, and was rejected. The second resolve request was $\mathsf{RR}_{\bar{X}}^j$, and was resolved. Note that there is no need to make $\bar{X}$ explicit in the notation, as by construction, T only receives one request from each X and $\bar{X}$. Prefixes of (i, X, rej, j, res) correspond to states in which only a subset of the above-mentioned events have occured, variations of exchanging rej and res have their natural interpretation.

5 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].

A *probabilistic function* $f\colon A \to B$ is a function that for each element $a \in A$ returns a probability distribution on B. We often simply write $f(a)$ for the result of the random experiment of choosing an element b from B, each with the probability $(f(a))(b)$.

5.1 Game Structures and Strategies

Definition 5.1 (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*.*

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 called the* principal in q *and denoted with* $Pr(q)$*. 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 5.2 (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*.*

Note that strategies depend on the *state* only, and not on the history of the computation. History-aware strategies can be defined analogously. For our application in the setting of cryptographic protocols, our definitions ensure that all relevant aspects of the history can be derived from the state, hence for ease of notation we only consider "memoryless" strategies.

Let $S_{PR} = \{S_a \mid a \in PR\}$ be a strategy for PR, and let $P = p_0 p_1 p_2 \ldots$ be a path, i.e., a sequence of states in $\mathscr{G}$. With $|P|$, we denote the number of states in P (which might be ∞). We now define

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

i.e., the probability that when the principals follow their strategies from S_{PR}, the resulting play follows the path P. With $P[i]$ for $i \geq 0$ we denote the ith state on P. With $P[i, \infty]$ we denote the sub-path of P starting at $P[i]$.

Next, we define the set of pATL*-formulas. Our syntax and semantics are almost identical to the ones used in [CL07].

Definition 5.3 (probabilistic alternating-time temporal 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.*
- *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.*

The semantics of pATL*-formulas is given in the next definition.

Definition 5.4 (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*
  - *$\mathscr{G}, q \models \langle\langle A\rangle\rangle^{\geq\alpha}\varphi$ if there is a strategy S_A for A such that for all strategies $S_{\overline{A}}$ for $\overline{A}$, $\sum_{\mathscr{G},P\models\varphi,P[0]=q} prob_{S_A\cup S_{\overline{A}}}(P) \geq \alpha$, ($\leq \alpha$ analogously)*
  - *$\mathscr{G}, q \models [[A]]^{\geq\alpha}\varphi$ if for all strategies S_A for A, there is a strategy $S_{\overline{A}}$ for $\overline{A}$ such that $\sum_{\mathscr{G},P\models\varphi,P[0]=q} prob_{S_A\cup S_{\overline{A}}}(P) \geq \alpha$, ($\leq \alpha$ analogously)*
- *For a path P and a state formula φ, $\mathscr{G}, P \models \varphi$ if $\mathscr{G}, P[0] \models \varphi$.*
- *For a path P and path formulas φ, ψ, we have $\mathscr{G}, P \models \varphi \mathsf{U} \psi$ if there is some $i \geq 0$ such that (i) $\mathscr{G}, P[i,\infty] \models \psi$, and (ii) for all $0 \leq j < i$, we have that $\mathscr{G}, P[j,\infty] \models \varphi$.*
- *for a path P and path formulas φ, ψ, we have $\mathscr{G}, P \models \varphi \mathsf{R} \psi$ if for all $i \geq 0$, we have (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. In the above definition, the (possibly uncountable) sums $\sum_{\mathscr{G},P\models\varphi} \mathrm{prob}_{S_A\cup S_{\overline{A}}}(P)$ can be rewritten as sums over countably many elements (there can only be a countable number of paths with a non-zero probability). Hence the value of the sums is well-defined. Note that by construction, $\neg\langle\langle A\rangle\rangle^{\geq\alpha}\neg\varphi$ is equivalent to $[[A]]^{\geq 1-\alpha}\varphi$, and $\langle\langle A\rangle\rangle^{\leq\alpha}\varphi$ is equivalent to $\langle\langle A\rangle\rangle^{\geq 1-\alpha}\neg\varphi$. Analogously, U is the dual operator of R.

We now consider a formula of the form $[[A]]^{\geq\alpha}\varphi$. Using the same proof as in [Kuh53] (see also [BL69]), one can show that

1. The "game" in which the coalition A tries to lower the probability of φ below α (and hence the coalition $\overline{A}$ attempting to ensure the probability is at least α) is determined: When $[[A]]^{\geq\alpha}\varphi$ is satisfied (i.e., A cannot "win" the game), then $\langle\langle\overline{A}\rangle\rangle^{\geq\alpha}$ is satisfied (i.e., $\overline{A}$ has a strategy to "win"). Hence $[[A]]^{\geq\alpha}\varphi$ implies $\langle\langle\overline{A}\rangle\rangle^{\geq\alpha}\varphi$. Since the other direction is trivial, the formulas are equivalent (this equivalence is already mentioned for sequential, non-randomized game structures in [AHK02]).
2. The optimal strategies in the above game are pure, i.e., not randomized.

The second point above shows that our restriction to determined strategies can be made without loss of generality; the first point shows that in our formulas, we can avoid the [[.]]-operator. We say that a formula is in *[[.]]-free positive normal form* if no [[.]]-operator appears, negation only appears directly before propositional variables, and no $\langle\langle.\rangle\rangle^{\leq\alpha}$-operator appears. Using the above-mentioned dualities and the usual one between the Boolean operators $\wedge$ and $\vee$, we conclude:

Proposition 5.5. *Every* pATL*-*formula is equivalent to an* pATL*-*formula in [[.]]-free positive normal form.*

5.2 Game Structures for Protocol Analysis

We now define the PGS canonically induced by a protocol and our symbolic protocol model. This PGS canonically represents the states and actions described in Section 3.3. Note that instead of introducing a principal for the key generator in the GS, we let the scheduler perform the corresponding move (obviously, this does not change the strategies present in the structure).

Definition 5.6 (protocol game structure). *Let* $Pr = (\Pi_1, \ldots, \Pi_k, \mathscr{I}_0)$ *be a 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 of moves is as below,*
- *for each local state* v *of a rule in* $\Pi_1, \ldots, \Pi_k$ *there is a propositional variable* x_v*, for each principal* $a \in Ids$ *there is a variable* c_a*, and for each* $d \in \{1, \ldots, k, \mathscr{S}, \mathscr{A}\}$ *there is a variable* a_d*.*
- *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, except for the following cases:*

1. *If* $a \in \{1, \ldots, k\}$*, and its current local state is randomized. In this case:*
   - *the principal* a *has a single move available in* q*,*
   - *the outcome of* q *specified by* δ *results from choosing each possible successor state with the corresponding probability from the protocol description*
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 present in $\mathscr{G}_{Pr}$. Also note that we do not make any explicit requirements about "fairness" of scheduling: Fairness requirements may be specified by the pATL*-formula defining the security requirements of the protocol. For an example of how to treat fairless, see our formal definition of symbolical unbalance in Section 8.

However, the model does demand a minimal amount of "fair scheduling" of the scheduler: A principal for a protocol role Π_i that is in a local state where it expects to either receive a message (from the network or directly with a direct connection), or can act without receiving a message (this is modeled by a rule with an empty incoming term in the role definition), then during the protocol run, if no message is delivered to the principal, the scheduler eventually has to activete the principal (without an incoming message). This requirement is inherent in the model, as by construction, the scheduler is not allowed to activate the adversary twice in a row when the protocol execution has not progressed in the first activation (i.e., no principal changed local state). Hence if the role Π_i never receives an incoming message, there is a point when there is no other option for the scheduler than to activate the role Π_i (note that our protocols allow only finitely many steps). Thus a principal will not remain in a state forever, when in that state he has an outgoing edge labeled with a protocol rule that has an incoming empty term. Similarly, from our model it follows immediately that the scheduler needs to activate the adversary when other principals "run out of actions," using the $a_{\mathscr{A}}$-variable, it is possible to demand additional properties of $\mathscr{A}$-activation in the formulas.

From the protocol model it follows that in a final state of $\mathscr{G}_{Pr}$ (i.e., a state in which none of the players has an available action), no principal is waiting for an incoming term that is derivable by the adversary, and no principal is in a current local state in which it can act without input from the adversary (i.e., with a protocol rule where the incoming term is ϵ).

6 The Computational Model

We now explain our computational model, which is based on standard cryptographic models as [BR93]. Our main addition is to reason about strategies explicitely. With a computational strategy we mean an algorithm that, given information about the current state, determines the next move of the principal. Our principals access a *strategy machine* which is a strategy in the above sense. The adversary is an algorithm which includes its strategy as well as any attacks it might use against the signature scheme. Since strategies are polynomial-time computable functions, we fix the set of relevant strategies (Turing machines) before a protocol run—quantifying over Turing machines during a protocol execution would allow the machines to non-uniformly depend on the security parameter: A machine chosen in an protocol run with security paramater η could have a hard-coded table of prime factorizations of all integers with bitlength up to η, compromising the security of signature schemes relying on the hardness of the factorization problem. Hence we demand that a strategy must be "successful" for *every* security parameter. This is a natural assumption, and one that is used (with respect to the adversary) in virtually all definitions of security of computational realizations of cryptographic primitives. The set of strategies (and the strategies themselves) naturally may depend on the security criterion that a protocol is supposed to satisfy, i.e., on the pATL*-formula φ under consideration. Also, an adversary typically is a specific algorithm "breaking" a specific security feature of a protocol. Hence the adversary may depend on the formula φ as well.

For executing a protocol, the machines use a signature scheme that consists of a triple of algorithms:

- a key generator keygen, which on input $(1^\eta, r)$ (where η is the security parameter and r is a sequence of bits, interpreted as a random bitstring) produces a pair of public and secret keys,
- a signature algorithm sign, which on input (s, r, sk) (where s is a bitstring, r is a sequence of bits, and sk is a secret key) returns a bitstring t,
- a verification algorithm verify, which on input (s, t, pk) (where s and t are bitstrings and pk is a public key) produces as output a single bit.

These algorithms operate in the expected manner, details see Appendix B.

We now define how protocols are executed in a computational model. The machines in the computational model share tapes to enable direct communication. For the computational execution of a protocol $Pr = (\Pi_1, \ldots, \Pi_k, \mathscr{I}_0)$, the following Turing machines are present in the system, and each pair of machines shares a tape:

Principal machines. For each $i \in \{1, \ldots, k\}$, a machine M_i, which simulates the role Π_i. It parses incoming messages and computes responses according to the protocol description in the obvious way. Messages adressed to another principal are written on the shared tape for the corresponding principal, messages to be sent to the adversary are written on the tape shared with $\mathscr{A}$. A machine simulating a network-ignoring role ignores all messages received on the tape shared with the adversary machine. To resolve strategic choices, a principal machine may communicate with its strategy machine (see below). In a randomized local state, the principal chooses each of the possible actions with the probability according to the protocol description (recall that randomized states only have outgoing edges in which no term is received, hence all of the successor edges in the protocol role can be performed). In randomized local states, a machine simulating a protocol rule does not activate its strategy machine. In both cases, the adversary is informed of the move executed by the principal machine (i.e., $\mathscr{A}$ receives an input specifying the current vertex in the protocol description that the principal machine now is in). In order to create and verify signatures, the principal machines use the algorithms from the computational signature scheme

in the obvious way. More details on the (straight-forward) way in which these machines operate are given in Appendix C. After the principal machine performed its computation, the machine representing the receiver of the outgoing message is activated next, unless the message is empty and the recepient is the adversary, in which case the scheduler is activated.

Scheduler machine. A scheduler machine $\mathscr{S}$, which plays the same role as the symbolic scheduler: It may activate machines for principals as well as the adversary machine.

Adversary machine. An adversary machine $\mathscr{A}$. This machine can perform any probabilistic polynomial-time computation, and in each activation, either process an incoming message (if there is an unread message on one of its incoming tapes), or write a bitstring to a single incoming tape of another machine (if there is no incoming message currently). If the adversary produces output to another machine, this machine is activated next. Otherwise, the scheduler machine is activated. Additionally, at the beginning of the protocol run the adversary machine can be called with a special parameter which leads to the printing of the list of identities that $\mathscr{A}$ wants to corrupt on a special output tape (the adversary does not have write access to any shared tapes with principal machines in this step).

Strategy machines. For each $a \in \{1, \ldots, k, \mathscr{S}\}$, a machine M_a^s deciding which action the corresponding principal performs. These machines have access to the *entire* configuration of the computational system, i.e., to the configuration of all involved Turing machines as well as randomness used so far in the protocol run. Strategy machines operate as follows: Whenever the principal a is active and has more than one possible action, the machine operating the role of A sends a symbol ? to its corresponding strategy machine, which is then activated. The strategy machine then replies with a number uniquely identifying the move. The communication between the principal machine for a and its strategy machine happens using a shared tape, i.e., the adversary may not interfere with this communication. Strategy machines are not activated by the scheduler, but only used by principals during their activation.

In the same way as in the symbolic model, if the adversary delivered a message to a principal machine that the latter did not accept (i.e., the machine did not change local state), the scheduler may not activate the adversary again in the following step, but has to activate a principal machine.

Note that the assumption that strategy machines have complete information about the current computational state and that the adversary is informed of internal decisions of the principals is required to transfer symbolic strategies to the computational model—see the Conclusion (Section 9) for a discussion of these issues.

6.1 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 $a \in Ids$, start keygen on input 1^η, store the secret key in sk_a and the public key in pk_a. Initialize machines for every role $1, \ldots, k$, $\mathscr{A}$, and $\mathscr{S}$, and strategy machines for each role $1, \ldots, k$ and $\mathscr{S}$. Each machine gets access to all public keys, the private key of the identity of the corresponding role (for machines simulating a protocol principal), and to the security parameter 1^η. The machines may only perform local computations in this step.
2. *Corruption.* The adversary prints a set C of identities and receives the private key of all $a \in C$.
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 (note that there is only a finite number of actions that can be performed after the adversary stops, hence the experiment will terminate).

It is clear that the experiment runs in polynomial time. A *step* in the computational run of the protocol is the action of one of the principal machines, the scheduler machine, or the adversary (a step may involve communication with a strategy machine). The first step is key generation and initialization, the next one is the corruption step, after which the scheduler machine controls the next activation. A *computational state* q of Pr consists of the configurations of all involved Turing machines in the computational execution of a protocol directly before the execution of a step (i.e., no machine is currently running), plus an indicator for the next machine to be activated (a principal machine, the scheduler, or the adversary), plus the so-far used random bits. An identity is *corrupted* in q if in the corruption phase of a protocol run leading up to q, the adversary printed a list C containing the corresponding identity. Note that the security parameter and the set of corrupted identities can be derived from the current state and the specification of the adversary machine. 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: Note that 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. By this we mean to replace, in the current global configuration of the computational system, the configuration of the currently active strategy machine with the initial configuration of the new strategy machine (note that strategy machines have access to the local configuration of all running Turing machines in the system, hence it is not necessary to give the newly introduced machine access to the configuration of its predecessor).

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, with $\mathrm{prob}_{q_c, S_A, \varphi}(P_c)$, 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).

6.2 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 6.1 (strategy set). *Let φ be a* pATL**-formula in* $[[.]]$*-free positive normal form. A* strategy set *for φ is a pair $(\mathscr{A}, S)$ consisting 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 will be executed in a protocol run will be addressed below. Due to Proposition 5.5, it suffices to define satisfaction for formulas in $[[.]]$-free positive normal form.

The following definition is straight-forward, 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 that are 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 $\mathscr{A} \notin A$, this models the usual assumption in pATL*, that if a coalition A works to achieve some goal, then the remainder of the players tries to achieve the opposite (and hence needs to be aware of the goal attempted by A).

Definition 6.2 (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 $\psi = a_i$ for a variable a_i, then $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, q_c \models \psi$, iff in q_c, the principal i is activated next.*
- *If $\psi = c_a$ for a variable c_i, then $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, q_c \models \psi$, iff a is corrupted in q.*
- *If $\psi = x_v$, for a variable x_v, then $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, q_c \models \psi$, iff in q_c, the simulation of 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 $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, q_c \models \psi$, iff* $\sum_{P_c\colon \mathscr{C}^{\eta}_{\mathrm{Pr}}, St, P_c \models \chi} prob_{q_c, S(\psi), \psi}(P_c) \geq \alpha$.
- *If ψ is a state formula, then $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, P_c \models \psi$ iff $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, P_c[0] \models \psi$.*
- *If $\psi = \chi \mathsf{U} \phi$, then $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, P_c \models \psi$ iff there is some $i \geq 0$ such that $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, P_c[i, \infty] \models \phi$ and $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, P_c[j, \infty] \models \chi$ for all $0 \leq j < i$.*
- *If $\psi = \chi \mathsf{R} \phi$, then $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, P_c \models \varphi, v$ iff for all $i \geq 0$ $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, P_c[i, \infty] \models \phi$, or $\mathscr{C}^{\eta}_{\mathrm{Pr}}, St, P_c[j, \infty] \models \chi$ for some $j \leq i$.*

We now define which strategy machines will be running in the execution of a protocol. The quantification here canonically mirrors first-order logic, with the simplification that our strategies do not need to depend on strategies that previously were executed in the protocol run so far.

Let φ be a pATL*-formula for a protocol Pr, and let $\psi = \langle\langle A\rangle\rangle^{\geq\alpha}\chi$ be a subformula of φ. We say 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. We consider the existentially quantified and universally quantified principals separately.

Definition 6.3 (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 such that 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).

7 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 7.1 (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 as explained earlier leads to problems in the computational model. Hence we fix the set of available strategies before the protocol is actually run. 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 is a natural requirement, 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. We only consider security properties defined by $\mathscr{A}$-monotone formulas. 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 7.2 (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). See Section 8 for an application of Theorem 7.2 in the context of contract signing protocols, and an example of the guarantees that the theorem gives for executing a protocol in the computational model.

The remainder of Section 7 is dedicated to proving Theorem 7.2: In Section 7.1, we establish a canonical correspondance between symbolic and computational states, Section 7.2 then proves Theorem 7.2.

7.1 Relating Computational and Symbolic States

We now define a correspondance between computational and symbolical states. In the following, let q_c be a computational state. Note that randomization is only used by the principals when determining the values of (message and randomization) nonces, and when choosing a move in a randomized local state. Additionally, the adversary and the key generator make use of randomness. To define the symbolic state corresponding to q_c, we first define a symbolic representation of each bitstring message present in the state. This representation is defined analogously to the parsing of messages (see Appendix C), we only note the non-trivial cases. In order to simplify the definitions, we assume that before the protocol run is started, the adversary randomly chooses values for the (message and randomization) nonces it will use in the protocol run, and when constructing bitstrings to send to principals during the protocol execution, the adversary only chooses nonces from this list and nonces that it received from the principals during the protocol execution. Since we only study a single session of a protocol in which each term that can be accepted by a principal has bounded length, there is only a finite number of nonces that the adversary can use in the creation of messages, and a bound on this number can be computed based on the protocol description alone. Hence the above assumption on the adversary can be made without loss of generality. We now define the symbolic representation of messages, where we only cover the non-trivial cases (the others are dealt with in a straight-forward manner, analogous to the definition of matching in Appendix C). In the following, let m_c be a message present in the system (i.e., a sub-message of a bitstring message accepted or written by a principal in the protocol run so far).

– If m_c is a nonce $(\mathtt{nonce}, s)$, then since s appears in a message present in q_c, and we assumed that the adversary and all principals determine random values for all nonces at the beginning of the protocol run, the value s appears in a principal's or the adversary's list of nonce values.
  1. If s is the value chosen by the machine simulating the role i with identity A for the nonce $N_m^{A,i}$, then $\mathrm{symb}(m_c) = N_m^{A,i}$.
  2. Otherwise, due to the above, s appears as the i-th message nonce in the list of nonces created by the adversary. In this case, let $\mathrm{symb}(m) = N_m^{\mathscr{A},i}$.

  In the above case distinction, if more than one case applies, then choose a representation using some well-defined order on the principals. Note that this case only occurs when different nonces chosen by principals have the same value. This event occurs only with negligible probability in the security parameter.
– If m_c is of the form $(\mathsf{sig}, m'_c, s, k)$, then let A be an identity with public key k. Assume that $\mathsf{verify}(s, m'_c, k) = 1$. If m_c was obtained by a call of sign, then let N be the symbol representing the randomization nonce used in that call. Otherwise, let N be a symbol for an adversary randomization nonce. In both cases, let $\mathrm{symb}(m_c) = \mathsf{sig}(a, N, \mathrm{symb}(m'_c))$. Again, if there is more than one nonce N satisfying the condition, choose one according to some arbitrary order.

In the above, whenever none of the mentioned cases applies, the value $\mathrm{symb}(m_c)$ is undefined (note that such a term is never accecpted by a principal, as due to our protocols not using encryption, principals are able to scan the entire incoming messages before processing it, see Appendix C for the straight-forward details). Note that we do not define a symbolical term corresponding to a randomization nonce—by the definition of protocols, a randomization nonce is never received or sent as a message on its own, but only used in combination with a signature. We now can define the *symbolic state corresponding to* q_c as the state $\mathrm{symb}(q_c) = (a, \sigma_s, v_1, \ldots, v_k, \mathscr{I}, C, m)$, where

– a is the principal marked for activation in q_c,
– σ_s is the substitution defined as $\sigma_s(x) = \mathrm{symb}(\sigma_c(x))$, where σ_c denotes the union of substitutions maintained by the computational principal machines (note that these have disjoint domains, and symb(.) is defined for every term that has been received by a principal),
– for $i = 1, \ldots, k$, v_i is the local symbolic state of the role Π_i as simulated by the principal machine,
– $\mathscr{I}$ contains, for every message m_c written to the adversary's incoming tape during the protocol run so far, the term $\mathrm{symb}(m_c)$ (again note that these are all well-defined, since the messages m_c have been sent by principals),
– $m = \mathrm{symb}(m_c)$, where m_c is the bitstring produced as output by the adversary or principal which was last activated (note that by definition, a principal only writes to a single output tape during its activation). If $\mathrm{symb}(m_c)$ is not defined, then for m we choose a special garbage symbol that is not used anywhere else and does not match with any variable (formally, we extend the set of types with a garbage type that is used nowhere else, and principals may not use variables of this type. We do not formalize this further—note that $\mathrm{symb}(m_c)$ can only be undefined if m_c is not accepted by any principal machine).

For a computational path $P_c = q_1^c \ldots, q_n^c$, we define the corresponding symbolic path $\mathrm{symb}(P_c)$ as $\mathrm{symb}(q_1^c) \ldots \mathrm{symb}(q_n^c)$. We are interested in paths resulting from computational runs of the protocol. In order to relate protocol runs in the computational and symbolic models, we first show that with overwhelming probability, a run in the former is also possible in the latter.

Definition 7.3. *Let Pr be a protocol, φ an $\mathscr{A}$-monotone* pATL**-formula for Pr, let $(\mathscr{A}, S)$ be a strategy set for φ, let η be a security parameter, and let r be a sequence of random bits of*

length required in protocol run with security parameter η. Then $\mathrm{tr_c}(Pr, \eta, \mathrm{r}, (\mathscr{A}, \mathrm{S}))$ *is the sequence of global computational states in the experiment run of Protocol Pr with adversary $\mathscr{A}$, strategy machines as specified by S, and random coins r.*

We say that a trace $\mathrm{tr_c}(Pr, \eta, \mathrm{r}, (\mathscr{A}, \mathrm{S})) = (q_c^1, \ldots, q_c^n)$ is *symbolically valid* if $\mathrm{symb}(q_c^1) \ldots \mathrm{symb}(q_c^n)$ is a valid path in $\mathscr{G}_{Pr}$, i.e., for all $i < n$, there is a move m for $Pr(\mathrm{symb}(q_c^i))$ such that $prob(\delta(\mathrm{symb}(q_c^i), m) = \mathrm{symb}(q_c^{i+1})) > 0$. The following theorem is a key step in relating the symbolic and computational models; as it ensures that traces of computational protocol runs are symbolically valid with overwhelming probability.

Theorem 7.4. *Let Pr be a protocol, φ an $\mathscr{A}$-monotone* pATL**-formula for Pr, and let $(\mathscr{A}.S)$ be a strategy set for φ. Then the probability that* $\mathrm{tr_c}(Pr, \eta, \mathrm{r}, (\mathscr{A}, \mathrm{S}))$ *is not symbolically valid is negligible in η, when r is chosen uniformly at random among all bit sequences of the corresponding length, and the signature scheme is resistant against existential forgery.*

Proof. By construction, the bitlength of all message and randomization nonces is at least linear in the security parameter. Therefore, the probability that two nonces generated by different principals coincide is negligible in the security parameter. For the remainder of the proof, we therefore only consider protocol runs in which such a collision does not occur. In particular, this implies that each nonce has been generated by a well-defined principal (which might be the adversary).

From the construction of the computational and symbolic protocol executions, it is obvious that all actions performed in the computational system can also be performed in the symbolic system, as long as the computational adversary only produces messages representing symbolical terms that are derivable in the current state. It therefore remains to show that with overwhelming probability, the adversary only generates such bitstring messages.

Assume that this is not the case, and $\mathscr{A}$ constructs a message m_c in a state q_c such that m_c is accepted by a principal, and $m = \mathrm{symb}(m_c) \notin d(\mathrm{symb}(q_c))$ with non-negligible probability. Since in our model, every subterm of a term t can be derived from t, We can assume that m is minimal (i.e., every proper subterm of m is derivable in the state $\mathrm{symb}(q_c)$). It thus follows that m is either a nonce or a signature. If m is a nonce, then, since $m \notin d(\mathrm{symb}(q_c))$, it follows that all tapes that the adversary can read only contain bitstrings that are independant of the value of m. Hence the adversary did not obtain m from a principal-sent message, and thus m is a nonce produced by the adversary himself. This is a contradiction, since by the above assumptions values of nonces do not collide, and thus m is an adversary nonce (and hence derivable). Therefore, m is a signature, $m = \mathsf{sig}(A, N, m')$ for some derivable message m', an identity A, and a randomization nonce N. We make a case distinction.

N is an adversary nonce. Since we assumed that collision of nonce values does not occur, and since principals always use self-generated randomization nonces in the computation of signatures, it follows that the signature has been computed by the adversary. Since $m \notin d(\mathrm{symb}(q_c))$, and $m' \in d(\mathrm{symb}(q_c))$, we know that A is not corrupted in $\mathrm{symb}(q_c)$, otherwise m would be derivable. Since A is not corrupted in $\mathrm{symb}(q_c)$, by construction of the experiment it follows that $\mathscr{A}$ does not have access to the secret key of A in the computational run of the experiment. By definition of $\mathrm{symb}(m)$, we know that m_c contains a signature for m' that is accepted by the algorithm verify. Hence, in producing the message m_c, the adversary has successfully performed an existential forgery, which only occurs with negligible probability.

N is a principal nonce. Since we assumed that m is not derivable by the adversary, and we do not use encryption in the protocols, it follows that the adversary does not have access to a bitstring

that contains a value depending on N. Due to the definition of the signature schemes, different randomization nonces lead to different signatures. Since the length of randomization nonces is at least linear in the security parameter, the probability that the adversary produces the same signature as the one resulting from using N is negligible in the security parameter. This concludes the proof of the theorem.

7.2 Proof of Theorem 7.2

We now show Theorem 7.2.

Proof. As explained in Section 7.1, we can assume without loss of generality that upon initialization, the adversary machine $\mathscr{A}$ generates the nonces it uses in the protocol run. By construction, principal machines do the same. We now define the computational realization of a symbolic strategy. Let s_a be a symbolic strategy for $a \in \{1, \ldots, k, \mathscr{S}, \mathscr{A}\}$. Then the computational realization of s_a is an algorithm that in every computational state q_c, determines the corresponding symbolic state $q = \mathrm{symb}(q_c)$, and then performs the computational move corresponding to $s_{\mathscr{A}}(q)$ (this can be computed from the symbolic move in a straight-forward way). Note that both the adversary and the strategy machines have enough information during every stage of the protocol run to determine the current symbolic state. Since the adversary can (see Section 7.1) be assumed to use only a finite set of nonces and all terms have bounded depth, the appearing adversary and principal strategies are finite and hence can be encoded in a polynomial time Turing machine (it is obvious that during a protocol run, principals only need to distinguish whether two occurrances of nonces contain the same value or not, the concrete value of the random bitstring in the nonces is irrelevant).

Hence, for a subformula $\langle\langle A\rangle\rangle^{\geq\alpha}\psi$ of φ and a symbolic state q such that $\mathscr{G}_{\mathrm{Pr}}, q \models \langle\langle A\rangle\rangle^{\geq\alpha}\psi$, we fix a strategy S_A for A that is "successful," and hard-code the strategies for the principals in A into the strategy machines and the adversary (note that if $\mathscr{A} \in A$, then φ is $\mathscr{A}$-positive). Let $M_\varphi^{a,q}$ be the resulting strategy machines. Let $M^{a,q}$ be arbitrary syntactically correct strategy machines for the remaining combinations of a and q. We define the existential strategy enumeration E as follows: On input $(a, \langle\langle A\rangle\rangle^{\geq\alpha}\psi)$, it returns a strategy machine which, when the first state in which it is activated is the state q, from then on simulates the strategy machine $M_\varphi^{a,q}$. If φ is $\mathscr{A}$-positive, then let $\mathscr{A}$ be the adversary machine that, when given the input ψ in a state q_c corresponding to a symbolic state q, executes the above-mentioned hard-coded symbolic strategies successful for ψ in the state q. It follows that when performing the experiment with the strategy set obtained from E, U, and $\mathscr{A}$, the existentially quantified principals always perform the symbolic strategies that are "successful" in the corresponding symbolic state.

In the case that φ is $\mathscr{A}$-negative, we replace the machine $\mathscr{A}$ with a machine $\mathscr{A}'$ that performs the same operations as $\mathscr{A}$, except that when $\mathscr{A}$ produces a bitstring message corresponding to a term that it cannot derive symbolically, $\mathscr{A}'$ stops. Due to Theorem 7.4, $\mathscr{A}$ and $\mathscr{A}'$ behave identically with all but negligible probability. Let symb be the event that $\mathscr{A}$ does not behave differently from $\mathscr{A}'$, and no nonces collide, let $\overline{\mathsf{symb}}$ be the event that symb does not occur. If symb occurs, then the relationship between symbolic and computational states defined in Appendix C is always unique: When a principal sends a message containing a bitstring value chosen for the nonce $N_m^{a,i}$, this string will not be interpreted as a different symbolical nonce $N_m^{b,j}$ for $b \neq a$ or $j \neq i$. Since the length of nonces is chosen linearly in the security parameter, the collision probability of nonces is negligible. This and Theorem 7.4 implies that there is a negligible function ϵ such that the probability of symb in an experiment with security parameter η is at least $1 - \epsilon(\eta)$.

We claim that under the condition symb, the probabilities of the formulas satisfied in the symbolic model exactly translate to the computational model. More formally, let $\mathscr{C}_{Pr}^{\eta\,\prime}$ be the transition system obtained from $\mathscr{C}_{Pr}^{\eta}$ by replacing $\mathscr{A}$ with $\mathscr{A}'$, and allowing only protocol runs in which the generated nonces do not collide. Then in $\mathscr{C}_{Pr}^{\eta\,\prime}$, the event symb always occurs.

Let $U^{\mathscr{A}}$ be the universal strategy enumeration U when φ is A-positive, and let $U^{\mathscr{A}}$ be the set containing be the universal strategy enumeration U and the adversary $\mathscr{A}$ when φ is $\mathscr{A}$-negative. Let $S(U^{\mathscr{A}})$ be the strategy set $(\mathscr{A}, E \cup U)$. We now show:

1. Let q_c be a computational state, let $q = \mathrm{symb}(q_c)$ be the corresponding symbolic state, and let ψ be a subformula of φ. Then the following are equivalent: (a) for all $U^{\mathscr{A}}$, $\mathscr{C}_{Pr}^{\eta\,\prime}, S(U), q_c \models \psi$ (b) $\mathscr{G}_{Pr}, q \models \psi$.
2. Let P_c be a computational path, let $P = \mathrm{symb}(P_c)$, and let ψ be a subformula of φ. Then the following are equivalent: (a) for all $U^{\mathscr{A}}$, $\mathscr{C}_{Pr}^{\eta\,\prime}, S, P_c \models \psi$ (b) $\mathscr{G}_{Pr}, P \models \psi$.

Proof. We show both claims by induction. For point 1, if ψ is a propositional variable or a negation of one, the claim follows directly from the definition of truth for propositional variables in $\mathscr{G}_{Pr}$ and $\mathscr{C}_{Pr}^{\eta}$. Induction for Boolean conjunction and disjunction is trivial.

Now assume that $\psi = \langle\langle A\rangle\rangle^{\geq\alpha}\chi$. First assume that $\mathscr{G}_{Pr}, q \models \psi$. Due to the construction of the machines operating the principals in A, in the resulting protocol run the principals in A perform the strategies S_A where s_A is a strategy for A that is "successful" against every strategy for $s_{\overline{A}}$ in the symbolic game structure. Since we only consider the conditioned experiment with condition symb, the adversary only performs symbolically valid moves. Note that since no symbolic state is visited twice during a protocol run, we can assume the adversary strategy to be deterministic (since the adversary might as well fix the random bits it is going to use in the protocol run in advance). Since we assumed that nonces do not collide, and different randomization nonces lead to different signatures, the function symb(.) constructed in Section 7.1 is a one-to-one correspondance between symbolical terms and their representation. Hence the matchings performed by the principals are exactly those that result in the symbolic protocol structure when the corresponding strategies are played. Also, the only relevant probabilistic events are the randomized choices by principals in their randomized local states. Let $S_{\overline{A}}$ be the symbolical strategies followed by the principals in $\overline{A}$ (the adversary follows a symbolical strategy due to the above, and the principal machines only make symbolically valid moves due to definition, hence such a strategy $S_{\overline{A}}$ exists). Then for a path P, we have $\mathrm{prob}_{S_{PR}}(P) = \sum_{\{P_c \mid \mathrm{symb}(P_c)=P\}} \mathrm{prob}_{q,S_A,\varphi}(P_c)$, i.e., the probability that a symbolical path P is followed in the symbolical model is the same as the probability that a computational path corresponding to P is followed in the computational model. By choice of S_A, the probability that a resulting path satisfies χ is at most α. By induction, a computational path P_c satisfies χ if and only if $\mathrm{symb}(P_c)$ satisfies χ. It therefore follows that the probability that χ is satisfied on the resulting path is the same in the symbolical and computational models, the claim follows. Now assume that $\mathscr{G}_{Pr}, q \models \psi$ is not true. Then for every strategy for A, there is a strategy $s_{\overline{A}}$ for $\overline{A}$ such that the probability of the resulting path satisfying χ is less than α. In particular, this is true for the strategies selected by E. Let $U^{\mathscr{A}}$ be the universal strategy enumeration containing these strategies (in the case that φ is a $\mathscr{A}$-negative formula, this contains the adversary strategy contained in $s_{\overline{A}}$). Then in the exact same way as the above direction, it follows that the strategy set obtained from S, $\mathscr{A}$, and $U^{\mathscr{A}}$ successfully ensures that the probability of the resulting path satisfying χ is less than α, and thus (for all U, $\mathscr{C}_{Pr}^{\eta\,\prime}, S(U), q_c \models \psi$) is not satisfied.

Point 2 now follows trivially: The case where ψ is a state formula is covered by the above, induction for propositional disjunction and conjunction is trivial. The case for operators U and R also follows by straight-forward induction.

We now show the statement of the theorem: We first show that if $\psi = \langle\langle A\rangle\rangle^{\geq\alpha}\chi$ is a subformula of φ such that $\mathscr{G}_{Pr}, q_0 \models \psi$, then $\mathscr{C}^{\eta}_{Pr}, S, q^{\eta}_{init} \models \langle\langle A\rangle\rangle^{\geq\alpha-\epsilon(\eta)}\chi$. Let s_A be the "successful" symbolic strategies followed by the principals in A when supplied with the formula ψ in q^{η}_{init}. Let sat be the event that the resulting computational path satisfies ψ. From the above claim, it follows that $prob(\mathsf{sat} \mid \mathsf{symb}) \geq \alpha$. Since $prob(\mathsf{symb}) \geq 1-\epsilon(\eta)$, it follows that $prob(\mathsf{sat} \cap \mathsf{symb}) \geq \alpha \cdot (1-\epsilon(\eta))$. Hence we have $prob(\mathsf{sat}) \geq prob(\mathsf{sat} \cap \mathsf{symb}) \geq \alpha(1-\epsilon(\eta)) \geq \alpha - \epsilon(\eta)$, as claimed.

Note that $\varphi^{\epsilon(\eta)}$ is obtained from formulas like the above with Boolean conjunction and disjunction. Hence the statement of the theorem follows by straight-forward induction on these operators.

8 Application to Contract Signing and the Gradual Commitment Protocol

In the following, let Pr be a contract-signing protocol with same structure as GCP, i.e., the roles in the protocol are an originator O, a responder R, a trusted third party T, and buffer principals B (for simplification, we treat the buffers as a single principal). All of these roles are operated by different identities, i.e., they do not share signature keys or access to nonces. For analysing 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 $\mathscr{S}$. The first move of the adversary is to corrupt $\overline{X}$, to be able to generate messages signed by $\overline{X}$ in the protocol run.

We now define pATL*-formulas describing the relevant security properties. $\varphi_{\mathsf{nc}} = \overline{c_X} \wedge \overline{c_T}$ expresses that the adversary corrupted neither X not T.

We further assume that from the local states of X, T, and B, it can be derived whether $\overline{X}$ (i.e., the adversary) or X have obtained a valid contract, and other facts about the protocol run: Let $\varphi_{\mathscr{A}\mathsf{c}}$ be a propositional formula that is true exactly in all states in which the adversary has obtained a valid contract, let $\varphi_{X\mathsf{c}}$ be a corresponding formula for X, and let φ_{dl} be a formula that is true exactly in those states where B has delivered all messages that it received.

We can 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, all buffers have delivered all messages, and neither X nor T have been corrupted.

We can now formally state what it means for a protocol to be unbalanced:

Definition 8.1 (balance). *A contract signing protocol Pr is symbolically (p_a, p_r)-unbalanced against X, if there is some $p > 0$ such that*

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

Recall from Section 5.2 that even though the scheduler works together with the adversary, our protocol model requires that whenever the honest principal X is in a state in which he can act without receiving an incoming message, then eventually the scheduler must activate X. As an example, in the protocol GCP, if X does not receive the next commitment from $\overline{X}$ (i.e.,

the adversary), then X eventually is allowed to send a resolve-request. In the following, with "unbalanced" we mean "unbalanced for R or O," and use "balanced" for "not unbalanced."

An additional property of CSPs is the following: A protocol is timely, if the honest signer always has a strategy that finishes the protocol run, i.e., reaches a state in which the honest principal has a contract, or the adversary will never obtain a contract. We let $\varphi_{X\mathsf{abr}} = \Box\neg\varphi_{\mathscr{A}\mathsf{c}}$, expressing a successful "abort" from the honest signer's point of view (the adversary will never be able to obtain a contract in the following protocol run), and stipulate:

Definition 8.2 (timeliness). *A contract signing protocol Pr is timely if its initial state satisfies the formula*

$$[[\mathscr{A}, T, B, X, \mathscr{S}]]^{\geq 1}\Box\langle\langle X\rangle\rangle^{\geq 1}\left(\Box(\Diamond\varphi_{\mathsf{dl}} \wedge \varphi_{\mathsf{nc}}) \rightarrow \Diamond(\varphi_{X\mathsf{c}} \vee \varphi_{X\mathsf{abr}})\right).$$

The formula expresses that for every possible behavior of the principals, in every state that is reached in the protocol run the honest signer has a strategy to finish the protocol run in the above sense (provided that the buffer principals eventually deliver their messages, and neither X not T are corrupted). Note that the [[.]]-free positive normal form equivalent of the above formula is $\mathscr{A}$-negative.

Our main result on GCP is:

Theorem 8.3 (balance and timeliness of GCP**).** *For all $n \geq 2$,* GCP_n *is timely and (p_a, p_r)-balanced for all $p_a + p_r \geq 1 + (1/n)$.*

The proof can be found in Appendix D.

As an illustration of the guarantees that our soundness result obtaines for CSPs in the computational model, consider the following: A computational definition of balance has been given in [CKW07]. Adepted to the setting with explicit probabilities, their definition states[1] that 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 did not receive 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 7.2, we obtain the following corollary:

[1] Note that we sligtly simplified their definition in omitting their polynomial-time "challenge"-function and fair scheduling—since as explained earlier, our model is essentially finite-state in the symbolic setting, it is clear that this function can be computed in polynomial time in our setting. Also, as mentioned earlier, fairness of scheduling is implicit in our model—hence we can regard the scheduler as working together with the adversary.

Corollary 8.4. *A contract signing protocol is computationally (p_a, p_r)-unbalanced if and only if it is symbolically (p_a, p_r)-unbalanced.*

The easy proof can be found in Appendix E. For GCP, we conclude

Corollary 8.5. *If $p_a + p_r \geq 1 + (1/n)$, GCP_n is computationally (p_a, p_r)-balanced.*

9 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 note that by replacing the randomized decision of the TTP with a strategic one, a protocol satisfying the usual non-probabilistic defintion of balance can be obtained (although with the significant drawback that the TTP is then behaving "strategically" which is an unnatural notion for a party which is supposed to act as an "impartial judge"). 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. Treating encryption requires different techniques: A strategy in the symbolic model is simply a function from the set of states to the set of possible moves, whereas encryption allows to hide information about the current state from the principals and the adversary, leading to a game with incomplete information. Formally treating incomplete information may also help in relaxing some of the conditions that we make in our computational model: Currently, both the adversary and the strategy machines are given access to enough information about the current state of the protocol run to determine in which "symbolic state" the system is. This is obviously necessary to transfer strategies from the symbolic to the computational model without significantly altering their success probability. In the current paper, the assumption that the adversary knows the internal decisions of the principals is not unrealistic: Since our model does not allow encryption, the adversary may read all network messages that the principals send, and hence obtains significant information about the internal state of the principals from their I/O-behaviour. Obviously, the situation is different when we allow encryption in cryptographic protocols.

Using a variant of ATL that allows to deal with incomplete information [JÅ06] therefore allows us to both treat more cryptographic primitives as well as study a more realistic computational model.

Also, contract signing in a concurrent model gives rise to interesting challenges [KKW06]. We believe that the challenges in a concurrent model and one with incomplete information are closely related.

References

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[Aiz08] Mihhail Aizatulin. A timely and balanced optimistic contract-signing protocol. Diploma thesis, University of Kiel, March 2008.

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

[ASW98] Nadarajah Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. In *In Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99. IEEE Computer Society Press, 1998.

[BDD+06] Michael Backes, Anupam Datta, Ante Derek, John C. Mitchell, and Mathieu Turuani. Compositional analysis of contract-signing protocols. *Theoretical Computer Science*, 367(1-2):33–56, 2006.

[BL69] J. Richard Buchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

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

[BR93] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In D. Stinson, editor, *Advances in Cryptology – Crypto '93, 13th Annual International Cryptology Conference*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 1993.

[CH06] Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer, 2006.

[CHW07] Véronique Cortier, Heinrich Hördegen, and Bogdan Warinschi. Explicit randomness is not necessary when modeling probabilistic encryption. *Electronic Notes in Theoretical Computer Science*, 186:49–65, 2007.

[CKKW06] Véronique Cortier, Steve Kremer, Ralf Küsters, and Bogdan Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In S. Arun-Kumar and Naveen Garg, editors, *FSTTCS*, volume 4337 of *Lecture Notes in Computer Science*, pages 176–187. Springer, 2006.

[CKS01] Rohit Chadha, Max I. Kanovich, and Andre Scedrov. Inductive methods and contract-signing protocols. In *ACM Conference on Computer and Communications Security*, pages 176–185, 2001.

[CKW07] Véronique Cortier, Ralf Küsters, and Bogdan Warinschi. A cryptographic model for branching time security properties - the case of contract signing protocols. In Joachim Biskup and Javier Lopez, editors, *ESORICS*, volume 4734 of *Lecture Notes in Computer Science*, pages 422–437. Springer, 2007.

[CL07] Taolue Chen and Jian Lu. Probabilistic alternating-time temporal logic and model checking algorithm. In J. Lei, editor, *FSKD (2)*, pages 35–39. IEEE Computer Society, 2007.

[CMSS05] Rohit Chadha, John C. Mitchell, Andre Scedrov, and Vitaly Shmatikov. Contract signing, optimism, and advantage. *Journal of Logic and Algebraic Programming*, 64(2):189–218, 2005.

[DY83] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[GJM99] Juan A. Garay, Markus Jakobsson, and Philip D. MacKenzie. Abuse-free optimistic contract signing. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. Springer, 1999.

[JÅ06] Wojciech Jamroga and Thomas Ågotnes. What agents can achieve under incomplete information. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *AAMAS*, pages 232–234. ACM, 2006.

[KKT07] Detlef Kähler, Ralf Küsters, and Tomasz Truderung. Infinite state AMC-model checking for cryptographic protocols. In *LICS*, pages 181—192. IEEE Computer Society, 2007.

[KKW05] Detlef Kähler, Ralf Küsters, and Thomas Wilke. Deciding properties of contract-signing protocols. In Volker Diekert and Bruno Durand, editors, *STACS*, volume 3404 of *Lecture Notes in Computer Science*, pages 158–169. Springer, 2005.

[KKW06] Detlef Kähler, Ralf Küsters, and Thomas Wilke. A dolev-yao-based definition of abuse-free protocols. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 95–106. Springer, 2006.

[KR02] Steve Kremer and Jean-François Raskin. Game analysis of abuse-free contract signing. In *CSFW*, pages 206–. IEEE Computer Society, 2002.

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

[KSW08] Klaas Ole Kürtz, Henning Schnoor, and Thomas Wilke. Computationally secure two-round authenticated message exchange. Technical Report 0809, Institut für Informatik, Christian-Albrechts-Universität zu Kiel, 2008.

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

[LM05] Yassine Lakhnech and Laurent Mazaré. Computationally sound verification of security protocols using Diffie-Hellman exponentiation. Technical report, Verimag, 2005.

[PG99] Henning Pagnia and Felix C. Gartner. On the impossibility of fair exchange without a trusted third party. Technical report, Darmstadt University of Technology, 1999.

A Protocol Requirements

We require k-protocols to fulfill the following conditions in order to be implementable in polynomial time: First, we restrict the way in which randomization nonces may be used: For a variable x or a nonce N with type rnonce, we require that if x or N appears in a subterm of the form $\mathsf{sig}(A, x, t)$ or $\mathsf{sig}(A, N, t)$, then *all* occurrances of x or N are of this form, with the *same* subterm t and identity A. Additionally, terms with type rnonce are only allowed to appear as the corresponding argument of a signature operator. This models that random strings for constructing signatures are not used for any other purpose, but a signature received or sent once may be referred to in later steps of the protocol execution. Note that this does not prevent the principal from signing the same message more than once, with different randomization each time. Additionally, for a subterm $\mathsf{sig}(A, x, t)$ or $\mathsf{sig}(A, N, t)$ in a role Π, if A is not the identity of the role Π, we require that the first occurance of this subterm is on the left-hand side of a rule. This expresses that principals cannot generate signatures for others, but of course have access to signatures received previously. We also require that if a rule $r \longrightarrow_d s$ or $r \longrightarrow_d^p s$ appears in one of the Π_i where $d \in \mathbb{N}$, then $1 \leq d \leq k$, i.e., messages are only adressed to existing roles. Finally, in a protocol we require that different roles (even when they have the same identities) use disjoint sets of nonces (this models that processes running in parallel on the same machine should not use the same randomness).

B Signature Schemes

In the computational model, the signature scheme from the symbolic model has to be instanciated with a computational signature scheme, consisting of

- a key generator keygen, which on input $(1^\eta, r)$ (where η is the security parameter and r is a sequence of bits, interpreted as a random bitstring) produces a pair of public and secret keys,
- a signature algorithm sign, which on input (s, r, sk) (where s is a bitstring, r is a sequence of bits, and sk is a secret key) returns a bitstring t,
- a verification algorithm verify, which on input (s, t, pk) (where s and t are bitstrings and pk is a public key) produces as output a single bit.

We allow sign to report an error when supplied with a random string r that is longer than required for the computation of the signature with the specified security parameter (which without loss of generality can be derived from the keys). We require the length of accepted random sequences to be at least linear in the security parameter. We demand that for all pairs (pk, sk) returned by keygen on input 1^η, all bitstrings s, and all sequences of bits r, $\mathsf{verify}(s, \mathsf{sign}(s, r, sk), pk) = 1$, unless the call of sign reports an error. We also require that if $r_1 \neq r_2$, then $\mathsf{sign}(s, r_1, sk) \neq (s, r_2, sk)$ for every bitstring s and secret key sk, if both runs of the signature algorithm are successful. All of the algorithms above are required to be deterministic polynomial time (as the randomness is explicitely handed to the algorithms as the bitstring r).

An adversary against a signature scheme is a probabilistic polynomial-time algorithm A which as input receives 1^η and a public key as computed by keygen on input $(1^\eta, r)$ for some random string r of appropiate length, and has access to a signature oracle, to which it may send strings s and receive valid signatures corresponding to the public key. The goal of the adversary is to produce a signature which was not obtained from the oracle, and which is accepted by the verification algorithm.

A signature scheme is resistant against *existential forgery* if for any adversary, the success probability of A is negligible in η, provived that the string r is chosen uniformly at random.

Often, a weaker notion of security is considered in which the adversary is only successful when it constructs a signature for a message for which it did not obtain a signature from the oracle, i.e., often it is assumed that when the adversary has access to a signature s for a message m, it may generate bitstrings $s' \neq s$ which are also valid signatures for m. It is easy to see that if the signature scheme allows these computations, we need to add a rule allowing the adversary to derive $\mathsf{sig}(A, N_2, t)$ from $\mathsf{sig}(A, N_1, t)$, if the adversary can derive the randomization nonce N_2.

The (straight-forward) generalization of the above to the situation in which the adversary attacks more than one key can be found in [KSW08].

C Details on Operation of Principal Machines

We now describe the imlementation of the principal machines in detail. Upon initialization, a principal machine chooses random values for all (message and randomization) nonces it will use in the protocol run. The random bitstrings used for the message nonces have length η, the random bitstrings for randomization nonces have the number of bits that the signature scheme requires for the corresponding security parameter for randomization nonces. In the initialization phase, the principal machine is given access to (binary representations of) all constants, identities, and public keys. A principal machine keeps copies of all bitstring messages received and sent. When encoding a message to be sent as a bitstring, we use tagging in order to let the message contain complete type information: A message nonce is represented as a pair $(\mathtt{nonce}, s)$, where s is the actual random string, a pair is represented as a triple $(\mathtt{pair}, m_1, m_2)$ where m_1 and m_2 are the representations of components of the paired message, a constant is of the form $(\mathtt{constant}, c)$ where c is the bitstring representation of the constant, $(\mathtt{id}, p)$ represents the encoding of an identity where p is a string uniquely describing an identity, and finally a signature is represented with a quadruple $(\mathtt{sig}, m, s, k)$, where m is the message, s the signature (i.e., the return value of the signature algorithm), and k is a binary representation of the public key which can be used to verify the signature. A principal expects incoming messages to be fully typed in the same manner, and ignores all bitstrings which do not follow this convention. Note that since we do not use encryption in our protocols, a principal can scan the entire incoming message to perform this verification, also a principal is able to determine the depth of any subterm of a received message immediately. We define the depth of a bitstring message m_c in the obvious way.

A principal machine maintains, besides a "current state" in the form of a pointer to a vertex in the tree describing the role it simulates, a "local" substitution σ mapping variables to bitstrings. Since different protocol rules use different variables, we also use σ for the union of these local substitutions. By constuction, a principal will only have to evaluate or modify σ for its own variables. When receiving an incoming bitstring message m_c, a principal machine parses the message by recursively trying to determine a matching against a term r on the left-hand side of a principal rule as follows. Before applying any of the above rules, the principal scans the incoming message for syntactical correctness and validity of all included signatures (see the case $\mathsf{sig}(A, N, r')$ below for details on signature verification).

- Assume that r is a variable x. If $\sigma(x)$ is defined, then matching is successful if and only if $\sigma(x) = m_c$. If $\sigma(x)$ is undefined, then check whether m_c is of the correct type, and $\mathit{depth}(m_c) \leq \mathit{maxdepth}(x)$. If this is not true, reject. Otherwise, matching is successful, and the substitution σ is extended with $\sigma(x) = m_c$.
- Assume that r is a pair $\langle r_1, r_2 \rangle$. If m_c is not of the form $(\mathtt{pair}, m_c^1, m_c^2)$ for bitstring messages m_c^1 and m_c^2, then reject. Otherwise, recursively match m_c^1 with r_1 and m_c^2 with r_2.

– Assume that r is a constant c. If m_c is not of the form $(\texttt{constant}, c')$, reject. Otherwise, matching is successful if and only if c' is identical to the value that was provided for the constant c in the intialization. The case where r denotes an identity works in exactly the same way.
– Assume that r is a signature $\mathsf{sig}(A, N, r')$ for a term r', an identity A, and a nonce N. If this exact term has been processed as either a send- or a receive action in a previous step of the protocol execution, accept if and only if the value of m_c is the same binary string as the value stored as the bitstring representation of r in the previous step. Otherwise, first check that m_c is of the form $(\texttt{sig}, m'_c, s, k)$, reject otherwise. Now first match m'_c with r', and then accept if $\mathsf{verify}(s, m'_c, k) = 1$, and k is the public key for A which the principal machine received in the initialization phase (note that the principal does not assign a value to N—by construction of protocols, the variable N is only used in the remainder of the protocol execution to access the bitstring representation of m_c, which the principal stores after matching is successful).

When a principal machine receives an input string m_c, in the "local state" v_i of the principal role, with the above rules it determines the set of successors v'_i such that $\ell(v_i, v'_i) = (p, \alpha, r \to_d s)$ where m_c and r match (plus the new substitutions in the case that this matching is applied). From these possible edges, the machine then determines those with the lowest priority (i.e., with the lowest number p in the value of ℓ). If the current local state is not randomized, it sends a symbol `?` to its strategy machine, receives a number identifying the successor node v'_i, and then follows the corresponding move. If the current local state is randomized, then each move according to a successor node v'_i is chosen with the probability associated with v'_i. In both cases, the new local state is the thus-determined v'_i, the substitution σ is updated as described above. The adversary receives a string indicating the new symbolic local state. The reply is computed in the obvious way, where we note that the restrictions on the way in which signatures and randomization nonces may appear in a protocol description (see Section 3.2) ensure that a principal can always compute the bitstring representation of the outgoing term in polynomial time.

D Proof of Theorem 8.3

In this section, we show that our protocol GCP is in fact balanced and timely in the symbolic model.

D.1 Unforgeability Lemma

The following lemma is an important tool in the proof—it states that at each point in the protocol execution, the adversary cannot generate a resolve request that belongs to a future round. The lemma is only phrased for the case where the originator is honest, but the symmetric variant holds true as well.

Lemma D.1 (unforgeability). *Let $n \geq 2$ and q any reachable state in the GS model for* GCP_n *where R is dishonest such that s_i, r_i, a_i, or c_i is the local state of the originator role and $O \notin C$. Then $\mathsf{RR}^j_R \in d(q)$ only if $j < i$.*

The proof is a simple induction on the number of steps in a protocol execution. One uses that every message of the form RR^i_O is a message signed by O, but $O \notin C$, and Definition 3.3 only allows to derive messages signed by O if $O \in C$ or the message is sent to the adversary. The rest follows from the structure of the protocol.

D.2 Proof of Timeliness, Theorem 8.3

We start by showing that a reply of T to an honest X is final in the sense that the resulting state is resolved or aborted for X. Assume that q is such a state and T has just replied to a request RR^i_X of X. For instance, the local state of T in q could be (i, X, res).

If the reply of T was a replacement contract, we are done, because it is eventually delivered. So for the rest assume that T sends a rejection token RT^i_X, which implies $i < n$.

The local state of T can be (i, X, rej) or $(i, \bar{X}, rej, j, rej)$.

First assume that $\overline{X}$ does not possess CMT^{n+1}_X in q.

The fact that T sends a rejection token as a reply to the above request means that the local state of T is (i, X, rej) or $(i, \bar{X}, rej, j, rej)$. We proceed by a case distinction.

First case, the local state of T is $(i, \bar{X}, rej, j, rej)$. Then no replacement contract is ever sent and because of $i < n$ x never sends CMT^{n+1}_X. The state is aborted.

Second case, the local state of T is (i, X, rej). In particular, $\overline{X}$ does not possess a replacement contract. Since X does not send a commitment such that $\overline{X}$ can construct a resolve request m such that $\mathsf{RR}^i_X < m$ (unforgeability), T will not resolve any request by $\overline{X}$. Therefore, the state is aborted, too.

To show timeliness for any X-honestly reachable state q note that X always has a strategy to receive a reply from T: Just send the most powerful request that it can generate to T and wait for an answer. Since very message is eventually delivered, the above applies.

D.3 Proof of Balance

Let q be a reachable state and U_r and U_a the chances of X to resolve and abort the session in q and $\overline{U}_r$ and $\overline{U}_a$ the corresponding chances for $\overline{X}$. Here, by the chance of X to reach φ we mean the largest number p such that $\langle\!\langle X \rangle\!\rangle^{\geq p}\varphi$. (Note that this number exists, because our protocol model is finite.) The goal is to show $\overline{U}_r + \overline{U}_a \leq 1 + 1/n$. To this end, we prove $U_r + U_a \geq 1 - 1/n$ by constructing an appropriate strategy.

If T's local state in q is of the form (i, X) or a state in which T has already replied to some request of X then by the proof of timeliness we know that this state is terminated for X, so that $U_r = 1$ or $U_a = 1$.

In the following assume that T has not responded to any requests of X yet. Let S be a strategy for X in which it sends the most powerful request $m = \mathsf{RR}^i_X$ that it can generate (if such a request is already underway, do nothing). Let us evaluate the outcomes of S depending on q.

- If $\overline{X}$ possesses CMT^{n+1}_X in q then by timeliness q must be resolved for X, so that $U_r = 1$. In the following cases assume that $\overline{X}$ does not possess CMT^{n+1}_X in q.
- Assume T has rejected a request of m' of $\overline{X}$ before (i.e., the local state of T is a suffix of $(j, \overline{X}, rej$ for some $j)$, and $m' \not< m$. In this case T will reply to m with a rejection token and we have seen in the proof of timeliness that this indeed means that the session is aborted for X. Thus $U_a = 1$ and we are done.
- Assume T has rejected a request m' of $\overline{X}$ before and $m' < m$. In this case, according to TTP rule 4, $U_r = i/n$ and $U_a = 1 - i/n$, so that $U_a + U_r = 1$.
- Assume T has resolved a request m' of $\overline{X}$ before. In this case T will resolve m as well, so that $U_r = 1$.
- The only case left is that T has not replied to any messages from either $\overline{X}$ or X in q, so that it is in the initial state. Assume first that $X = O$ and $\overline{X} = R$. Let $\overline{S}$ be any strategy of $\{\overline{X}, B\}$ in q. We make a case distinction depending on which message first reaches the TTP. In all cases,

let $P_a(\alpha)$ and $P_r(\alpha)$ be the probabilities that an aborted or a resolved states are reached. Let us

- Assume the first message to reach the TTP is RR_O^i. Then $P_r(\alpha) = i/n$ and $P_a(\alpha) = 1 - i/n$.
- Assume the first message to reach the TTP is RR_R^i. Then again $P_r(\alpha) = i/n$ and $P_a(\alpha) = 1 - i/n$.
- Assume the first message to reach the TTP is RR_R^{i+1}. Then $P_r(\alpha) = (i+1)/n$ and $P_a(\alpha) = 1 - (i+1)/n$.
- Assume the first message to reach the TTP is RR_R^j with $j < i$. In this case $\alpha < m$ and thus m is going to be treated according to TTP rule 3b when it arrives, thus $P_r(\alpha) = i/n$ and $P_a(\alpha) = 1 - i/n$.

Hence in all cases, $P_r(\alpha) \geq i/n$. But this means $U_r \geq i/n$. By the same argument $U_a \geq 1 - (i+1)/n$, thus giving $U_a + U_r \geq 1 - 1/n$.

Note that due to Lemma D.1, the adversary cannot derive resolve requests for R with an index of more that $i+1$, hence the above list covers all possible messages to reach the TTP (note that the honest originator O does only send a single resolve request, and then does not send any more messages in the protocol run).

The case $X = R$ and $\overline{X} = O$ is similar. The only difference is that valid requests of $\overline{X} = O$ would be RR_O^i and RR_O^{i-1}.

E Proof of Corollary 8.4

The proof mainly is straight-forward translation between the semantics of the symbolic and the computational models. We only highlight the non-trivial points:

Proof Sketch. A minor difference between our definition of computational unbalance above and the computational interpretation of the pATL*-formula defining unbalance is that in the computational definition, the adversary is required to recognize when it is in an unbalanced state. In our setting, this difference is not significant: As argued earlier, our game structure is essentially finite, and hence we can hard-code the information about which symbolical states are unbalanced into the adversary algorithm (note that the adversary can always determine the current symbolic state). In the following we use that implementations of finitely many different machines can naturally be realized as a single machine simulating one of the machines depending on the input.

First assume that a protocol Pr is symbolically (p_a, p_r)-unbalanced.

Let $\varphi_A = \langle\langle \mathscr{A}, B, \mathscr{S}\rangle\rangle^{\geq p_a} \Diamond \varphi_{\mathsf{res}}$, and $\varphi_B = \langle\langle \mathscr{A}, B, \mathscr{S}\rangle\rangle^{\geq p_r} \Diamond \varphi_{\mathsf{abr}}$, and let $\varphi = \langle\langle \mathscr{A}, T, B, X, \mathscr{S}\rangle\rangle^{\geq p} \Diamond (\varphi_A \wedge \varphi_B)$ Since Pr is symbolically (p_a, p_r)-unabalanced, it follows that $\mathscr{G}_{Pr}, q^0_{Pr} \models \varphi$. Since φ is $\mathscr{A}$-positive, Theorem 7.2 implies that there is an adversary machine $\mathscr{A}_c$, and strategy machines $\mathscr{S}_\varphi$, B_φ, B_{φ_A}, B_{φ_B}, $\mathscr{S}_{\varphi_A}$, $\mathscr{S}_{\varphi_B}$, T_φ, and X_φ (which we group as the set S_1) such that for all strategy machines T_{φ_A}, X_{φ_A} if the protocol is started with adversary $\mathscr{A}_c$, then a state is reached in which upon special input `abort` (which we interpret as φ_A), an aborted state is reached with probability at least p_a, and for all strategy machines T_{φ_B}, and X_{φ_B}, on input `resolve` (interpreted as φ_B), a resolved state is reached. It is clear that this corresponds directly to the computational definition.

Now assume that the protocol is not symbolically (p_a, p_r)-unbalanced. Then for every $p \geq 0$, the intial state $q^0_{Pr} \models \varphi$ of $\mathscr{G}_{Pr}$ satisfies the formula $\neg\varphi$, which is $\mathscr{A}$-negative.

Note that $\neg\varphi = \neg\langle\langle \mathscr{A}, T, B, X, \mathscr{S}\rangle\rangle^{\geq p} \Diamond (\varphi_A \wedge \varphi_B)$ is equivalent to $\langle\langle \emptyset\rangle\rangle^{\geq 1-p} \neg\Diamond (\varphi_A \wedge \varphi_B)$. Hence for every p, the empty coalition has a strategy to ensure that with probability at least $1-p$, no unbalanced state (meaning a state satisfying $\varphi_A \wedge \varphi_B$) is reached. Since the empty coalition

has exactly a single strategy, the strategies for different values of p are the same. Since only finitely many random choices occur during a protocol run, and each randomized choice only has finitely many outcomes with non-zero probability, the occurring probabilities only take finitely many values. Since the strategy of the empty coalition has a success probability arbitrarily close to 1, its success probability is in fact 1. Due to Theorem 7.2, this implies that an unbalanced state in the above sense is reached with probability $\epsilon(\eta)$ for a negligible function η only: For every adversary machine, there are strategy machines for X and $\mathscr{T}$ that can stop the adversary from reaching its goal with the specified probability, assuming they are provided with this goal using the special input `about` and `resolve`. These machines straight-forwardly translate to strategies for the formulas φ_A and φ_B in the symbolic definition (recall that existential quantification in φ corresponds to universal quantification in $\neg\varphi$ and vice versa). This concludes the proof. □