INSTITUT FÜR INFORMATIK

Symbolic Verification of Computational Security for Branching-Time Properties

Henning Schnoor

Bericht Nr. 8009
September 2008

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

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

Symbolic Verification of Computational Security for Branching-Time Properties

Henning Schnoor

Bericht Nr. 8009
September 2008

e-mail: schnoor@ti.informatik.uni-kiel.de

Symbolic Verification of Computational Security for Branching-Time Properties

Henning Schnoor
schnoor@ti.informatik.uni-kiel.de

September 2008

Abstract

Two different models for security of cryptographic protocols have been developed: *Symbolic security* is an abstract notion which can often be verified automatically. *Computational security* is defined in a realistic concurrent model against arbitrary, randomized polynomial-time attacks. A recent research trend is to prove that often, these security notions coincide, thereby transferring the decidability results from the abstract setting into the more realistic computational model. Previous results in this area are only concerned with *trace properties*, i.e., security goals that can be characterized as properties of single protocol runs. We prove the first equivalence result for a more complex class of goals, which include *balance* for contract signing protocols. Our result shows that computational security for these protocols can be verified automatically. The proof relies on a careful "derandomization" of realistic attacks.

1 Introduction

The design of cryptographic protocols is difficult and subject to many errors—often, problems in protocols are found only years after publication. The most famous example for this is the attack found by Gavin Lowe on a the Needham-Schroeder authentication protocol [Low96]. To formally prove security of cryptographic protocols, different models have been developed:

Symbolic models use abstraction from cryptographic primitives and represent messages and protocols over an algebraic structure. One of the most prominent examples of such models is the Dolev-Yao-model [DY83], which has been adopted and generalized for many scenarios. Security in

these models is often decidable when considering a finite number of sessions [RT03]. Decidability is sometimes preserved even when adding algebraic operators to the protocols, like exclusive or [CLS03] or Diffie-Hellman-exponentiation [CKRT03, Shm04]. This allows automatic security analysis of cryptographic protocols. The downside of these models is that while an attack on a abstract symbolic model usually translates into an attack on a concrete implementation, the other direction does not follow automatically: A security proof in the symbolic model only shows that there is no attack on the chosen level of abstraction.

Computational models (also called *cryptographic models*) consider protocols at the Turing machine level; principals are machines running concurrently. Security means that no probabilistic polynomial-time adversary can "break" the protocol with relevant probability. One of the first security proofs in such a model was achieved by Bellare and Rogoway for mutual authentication and key distribution in [BR93].

The security guarantees established by these types of models differ. Computational models arguably give stronger security, since the adversary can be any probabilistic polynomial-time algorithm, while in the symbolic models, the adversary may only perform actions expressible in the algebraic model. Also, the computational adversary can use randomness. On the other hand, the symbolic adversary is usually modelled to be nondeterministic, or alternatively to have knowledge about the internal states of honest principals that cannot be obtained by monitoring their input/output behaviour, and hence is unavailable to the adversary in the cryptographic model.

In the present paper, we contribute to the attempt to "bridge the gap" [AR02] between these viewpoints: We show that for a class of complex security goals, the symbolic and computational model give the same guarantees. Previous work in this area has focused on *trace properties*, where "bad" behaviours of protocols can be defined as properties of single executions of protocols (e.g., a security property disallows all executions where the adversary learns a certain secret, an authentication property demands that in any protocol run, if one party A "believes" that it has communicated with a party B, then B "believes" that it communicated with A, etc.). Our work deals with properties that cannot be expressed in this way. A prominent example for a more complex requirement occurs in *contract signing protocols* (see, e.g., [ASW98]). Here, a central security requirement is *balance* [CKS01]: If a party A negotiates with another party B, there should not be an "unbalanced" state where B has both a strategy to obtain a valid (i.e., signed) contract, and another strategy that prevents A from obtaining a valid contract. Such a state would allow B to enter negotiations with an outside party C, using that B can obtain a contract with A but is not bound to the

contract, to negotiate a better deal with C. Balance cannot be defined as a simple trace property, since it is acceptable for a protocol run to end with B getting a contract, and it is also admissible that there are protocol runs where A does not get a contract. Hence both outcomes are allowed, but B should not be able to unilaterally decide which alternative happens.

Balance can be expressed in a branching-time framework: The protocol is not allowed to *reach* a point such that B can control *all* possible outcomes of the protocol run. Cortier, Küsters, and Warinschi introduced a computational model for branching-time security in [CKW07], which can be used to express balance. On the other hand, Kähler, Küsters and Wilke defined a symbolic model allowing to express similar properties in [KKW05].

In this paper, we show that security in these two models is equivalent: A protocol is balanced in the symbolic model if and only if it is balanced in the computational model (the result holds for a bounded number of sessions). This proves that the symbolic model is a sound abstraction of the computational one, and most importantly implies that the decision procedure from [KKW05] can be used to prove computational security. To the best of our knowledge, this is the first result showing equivalence of symbolic and computational security that is not limited to trace properties.

Our proof first establishes a close relationship between states of the symbolic protocol and states of a system of Turing machines which realize the protocol. We then prove a "Mapping Lemma," which relates runs of the protocol in both models, and is similar to techniques used in proofs for trace properties (see [CH06]). The main difference is that our lemma deals with a non-idealized concurrent model, and addresses cryptographic primitives which may use randomness. The technically most intereresting part of our result is the proof of Theorem 4.2, which uses a trade-off between nondeterminism and randomness to "convert" adversaries between the symbolic and computational model: The computational adversary uses randomness to mirror the nondeterministic decision of the symbolic adversary, with high probability. To construct a symbolic adversary for an existing computational one, we show that in most protocol runs, the cryptographic primitives can be assumed to not use randomness at all. The symbolic adversary then can use nondeterminism to deal with the remaining randomness of the computational adversary.

Related Work: The first result relating symbolic to computational security was an influential paper by Abadi and Rogaway [AR02], who proved equivalence for encryption-scheme security. Further results established equivalence even for cases where the symbolic model is equipped with algebraic operators like Diffie-Hellman exponentiation [LM05]. Equivalence results have also been obtained in Canetti's UC-model, which ensures computational in-

distinguishibility in arbitrary contexts [CH06]. However, not all results have been positive: In [BP08], Backes and Pfitzmann prove that security in standard symbolic models does not imply cryptogaphic security for protocols using algebraic properties of exclusive or.

In [KKW05], it is explained how the ASW-protocol from [ASW98] can be defined in the symbolic model.

2 Prerequisites

2.1 The Symbolic Model

We briefly explain the symbolic model defined in [KKW05]. As usual, messages are defined over a term algebra containing a set of variables, a set of constants (e.g., names and random strings, also called nonces), a set of adversary constants $\mathcal{C}_{\mathcal{A}}$ (nonces generated by the adversary), and a set $\mathcal{K}$ of keys (all these sets are disjoint). We assume that there is a bijection $(.)^{-1}$ on $\mathcal{K}$ that maps a public (private) key to the corresponding private (public) key. *Plain terms* are defined as follows: Variables, (adversary) constants, and keys are plain terms. If t_1 and t_2 are plain terms, α is a key or a nonce, and $k \in \mathcal{K}$, then $<t_1, t_2>$ (pairing of terms), $\{t_1\}_k^a$ (asymmetric encryption of t_1 with key k), $\{t_1\}_\alpha^s$ (symmetric encryption of t_1 with key α) and $\mathsf{sig}_k(t_1)$ (term t_1 signed with respect to the public key k) are plain terms as well.[1] A *secure channel term* is of the form $\mathsf{sc}(i, j, m)$, where i and j are principal names and m is a plain term without variables. This term indicates that the principal with name i sends a message m to the principal named j on a secure channel, i.e., a channel that cannot be written to or blocked by the adversary (we allow the adversary to read secure channel messages).[2] These terms may only be generated by the principal i. A *term* is a plain term or a secure channel term. A (secure channel) term without variables is also called a *(secure channel) message*. A *(ground) substitition* is a function σ that assigns terms (without variables) to variables. We write $t\sigma$ for the term obtained from t by simultanously replacing every variable x with $\sigma(x)$, for all x in the domain of σ. A *principal rule* is of the form $R \Rightarrow S$, where R is a term or ϵ, and S is a term. A *principal* $\Pi = (V, E, r, \ell)$ is a finite tree

[1] Note that we only allow nonces and atomic keys as keys for symmetric encryption. If we allow arbitrary terms, then an adversary might have partial knowledge about a key of the form $<k_1, k_2>$

[2] It was shown in [PG99] that a contract signing protocol needs a *trusted third party* (TTP), which usually is involved only when problems occur. Communication with the TTP must use secure channels, otherwise the adversary can block the TTP. Hence secure channels are necessary for contract signing protocols.

with root $r \in V$ where ℓ maps every edge $(v, v') \in E$ to a principal rule $\ell(v, v')$ such that every variable occurring on the right-hand side of $\ell(v, v')$ occurs on the left-hand side of a rule on the path from r to v'. A *protocol* $P = ((\Pi_1, \ldots, \Pi_n), \mathcal{I}_0)$ consists of a finite set of principals and a finite set $\mathcal{I}_0$ of messages, the *initial adversary knowledge*. We require that $\mathcal{I}_0$ contains all public keys and names of principals. We also require that different principals have disjoint sets of variables, and that adversary constants do not appear in P.

Given a set $\mathcal{I}$, the adversary can derive messages as follows: $d(\mathcal{I})$ is the smallest set that contains $\mathcal{I}$ and all adversary constants, is closed under pairing and splitting up pairs, symmetric and asymmetric encryption and signature generation (if both key and message are known), and decryption with known keys, i.e., $\mathcal{I} \cup \mathcal{C}_{\mathcal{A}} \subseteq d(\mathcal{I})$, $<t_1, t_2> \in d(\mathcal{I})$ if and only if $t_1, t_2 \in d(\mathcal{I})$, further if $t, k \in d(\mathcal{I})$, then $\{t\}^a_k \in d(\mathcal{I})$, and if $t_1, \alpha \in d(\mathcal{I})$ for a key or nonce α, then $\{t_1\}^s_\alpha \in d(\mathcal{I})$. Decryption and signatures are modelled as follows: If $\{m\}^a_k, k^{-1} \in d(\mathcal{I})$, then $m \in d(\mathcal{I})$, if $\{t_1\}^s_\alpha \in d(\mathcal{I})$ and $\alpha \in d(\mathcal{I})$, then $t_1 \in d(\mathcal{I})$. Finally, if $m, k^{-1} \in d(\mathcal{I})$, then $\mathsf{sig}_k(m) \in d(\mathcal{I})$. Also, we assume that both public key and message can be derived from the signature, i.e., if $\mathsf{sig}_k(m) \in d(\mathcal{I})$, then $m, k \in d(\mathcal{I})$.

A *symbolic state* is of the form $((v_1, \ldots, v_n), \sigma, \mathcal{I}, \mathcal{SC})$, where σ is a ground substitution, each v_i is a node in Π_i, $\mathcal{I}$ is a finite sequence of messages (the *adversary knowledge*), and $\mathcal{SC}$ is a finite multi-set of secure channel messages, the *secure channel*. We write $\mathcal{I}(q)$ to denote the adversary knowledge in state q. The *initial state* of a protocol $((\Pi_1, \ldots, \Pi_n), \mathcal{I}_0)$ is $((r_1, \ldots, r_n), \sigma, \mathcal{I}_0, \emptyset)$, where for all i, r_i is the root of Π_i, and σ is the substitution with the empty domain. We now define transitions between the states. There is a transition from a state $((v_1, \ldots, v_n), \sigma, \mathcal{I}, \mathcal{SC})$ in the following three cases (for a complete formal definition, see [KKW05]):

Adversary Transitions The adversary can deliver a message $m \in d(\mathcal{I})$ to a principal Π_j which has a *matching* outgoing edge from its current node, i.e., an edge (v_j, v'_j) such that $\ell(v_j, v'_j) = R \Rightarrow S$ and there is a substitution σ' agreeing with σ on the domain of σ such that $R\sigma' = m$. In the case that there are several edges, one is picked nondeterministically. The new state of Π_j is v'_j, the new substitution is σ', and $S\sigma'$ is added to the adversary knowledge (and to the secure channel if it is a secure channel message).

Secure Channel Delivery A message $\mathsf{sc}(i, j, m)$ currently in the secure channel is removed from the secure channel and delivered to the principal Π_j, which has a matching outgoing edge in its current node (with

respect to the current substitution σ). This is handled analogously to the above case of an adversary-delivered message.

ϵ-Transitions If a principal Π_j is in a state that has an outgoing edge $\epsilon \Rightarrow S$, then Π_j can change state and send $S\sigma$ over the network or the secure channel, where sending is as in the above two cases.

The ϵ-transitions are used to model the realistic ability of a party to perform an action without being "triggered" from outside. A state can have outgoing transitions of all three kinds. For example, in a contract signing protocol, if a party does not receive an answer, then it can contact the trusted third party without waiting for a new incoming message. The graph induced by the transitions explained above is called the *transition graph* $\mathcal{G}_P$ of the protocol P. For a state q, $\mathcal{G}_{P,q}$ denotes the subgraph induced by q in $\mathcal{G}_P$. We call edges in $\mathcal{G}_P$ corresponding to adversary transitions *adversial edges*, and other edges *scheduler edges*. Finally, note that in the symbolic model, only a finite number of interleaving sessions can be expressed.

As an example for a principal A, consider a tree with just three nodes v_1, v_2, and v_3, edges (v_1, v_2) and (v_2, v_3), where $\ell(v_1, v_2) = \epsilon \Rightarrow N_A$ (N_A is a symbol for a nonce generated by A), and $\ell(v_2, v_3) = <\mathsf{sig}_{k_B}(N_a), x> \Rightarrow \mathsf{sig}_{k_A}(x)$ (where k_A and k_B are the public keys of A and B). This principal sends out a random number, and if it receives tis number signed with B's public key and some other input x, it will sign x with its private key and send the signature over the network. Hence A is willing to sign any text that it believes to come from B (we will not address the obvious security issues in this protocol).

2.2 The Computational Model

The computational model that we use has been defined by Cortier, Küsters, and Warinschi in [CKW07]. Here principals are modeled as interactive Turing machines (ITMs) which, in addition to work tapes, have a tape holding the security parameter, a tape for storing random coins, and input and output tapes. The security parameter influences the strength of cryptographic primitives. For many algorithms this is simply the bitlength of the involved keys, and of randomly generated strings. Each I/O tape has a name, enabling communication between machines sharing tapes with the same name. Additionally, there is an adversary machine $\mathcal{A}$ and a scheduler machine $\mathcal{S}$ (both of these are polynomial time ITMs, the adversary may be probabilistic). All these machines run concurrently. The runtime per activation of every ITM in the system is polynomially bounded in the content of all tapes. To allow

the adversary control over the network, the machines modelling principals do not communicate directly with each other, but hand their outgoing messages to the adversary, in writing the message to a special input tape of the adversary (the adversary machine does not necessarily get activated after receiving a message from a principal)[3]. To model the secure channel, for each pair of principal ITMs (M_i, M_j), there is a machine $\mathbf{RC}_j^i$ that relays the secure channel messages sent from the principal with name i to the one with name j. The adversary does not have write-access to the secure channel machines (but can read messages on the secure channels as well as messages sent over the network).

The scheduler machine is used to resolve nondeterminism, to schedule the secure channels, and to activate the adversary. It is described in [CKW07] as "an imaginary entity that is only needed to model how things are potentially scheduled in a real protocol run": Quantifying over all possible schedulers is quantifying over all possible behaviours of the system with regard to activation schedules, delivery schedules of the secure channel, and non-deterministic choices of principals in symbolic protocols. When the cryptogtaphic system is started, control is given to the scheduler, and control is also returned to the scheduler when a machine finishes its computation (except when a message is delivered to a principal, this principal is activated next). The scheduler communicates *directly* with the principal machines using their I/O tapes (the adversary does not have access to the internal behaviour of principal machines). The scheduler also controls when a secure channel delivers the message(s) stored, and has access to the complete internal configuration of the principal machines and the adversary, including so-far used random coins. A *computational state* is the configuration of all involved Turing machines, including random coins used so far.

Note that our case is a slight simplification of the model from [CKW07]: Their model allows to parametrize the knowledge of adversary and scheduler with *view oracles*. In the current paper, we are only interested in systems realizing symbolic protocols, we thus have to choose these view oracles reproducing the capabilities of scheduler and adversary in the symbolic world. Since the computational scheduler corresponds to symbolic nondeterminism, in our setting the scheduler has complete information over the entire state of the system (including local configuration of the principal TMs). The adversary, as in the symbolic model, has access to all messages sent over network and secure channels. With the more fine-grained approach in [CKW07],

[3]In the model as defined in [CKW07], there is a network buffer machine between any two principals, these machines are controlled by the adversary. In our formulation, we assume that messages are given directly to the adversary; this expresses that in the symbolic protocols, messages usually do not have a uniquely specified receiver.

it is possible to obtain slightly stronger security guarantees than the symbolic model can express. On the other hand, the symbolic model as defined in [KKW05] allows the corruption of secure channels, which is not possible in the computational model. To introduce this in the latter, one could use only a single secure channel machine, on which all principals (and the adversary) have write-access, and ensure authenticity of messages with signatures, where public/private keys for secure channel signatures are not used elsewhere. Then corrupting the secure channel messages for some principal can be modeled as obtaining that principal's secure channel signature key.

3 Protocol Mapping

We now show how to execute symbolic protocols in the computational model. In order to do this, we first introduce necessary restrictions of symbolic protocols to ensure they can be implemented in polynomial time (Section 3.1), and then show how to implement them in such a way that the computational system essentially has the same states as the symbolic model (Section 3.2). In Section 3.3, we prove that the computational realization in fact closely mirrors the behaviour of the symbolic model.

3.1 Executable Symbolic Protocols

In order to "translate" symbolic protocols to a concrete implementation using probabilistic polynomial-time Turing machines, we need to make a few assumptions about the protocol. Syntactically, we can assume that the initial intruder knowledge contains only a single (otherwise unused) constant symbol `init`, and that there is a principal that, called with this constant, sends the constant and the actual initial adversary knowledge over the network. We can therefore identify the sequence of messages sent so far and the adversary knowledge. We also require that there is a constant `fail` contained in the initial adversary knowledge such that when this constant is sent to a principal, the principal enters a mode (modeled as a subtree that is added to every node in the original tree) where it does not accept any incoming messages from the network anymore (but does continue to accept secure channel messages, and can perform ϵ-transitions).

Obviously, the cryptographic framework is in many ways more powerful than the symbolic one, and not every cryptographic protocol has a symbolic counterpart. On the other hand, there are also symbolic protocols which cannot be implemented in polynomial time. We restrict ourselves to protocols in which decryption and signature generation only is applied with keys that

the principal "knows:" Formally, we require that each principal i has an assigned set $\mathcal{K}_i$ of keys that it "owns," i.e., has access to the corresponding secret key (in typical Alice-and-Bob-notation, keys belonging to Bob are often called pk_B or similar). In such a protocol, a principal i may only use rules $R \Rightarrow S$ where for any term of the form $\{t_1\}_k^a$ appearing as a subterm of R, k is a key from $\mathcal{K}_i$. This expresses that asymmetric decryption is only possible when the secret key is known, and we do not consider the possibility that principals send their secret keys over the network (which is a reasonable assumption). However, if they do so, then our model allows the adversary to learn the key (if it was sent over the network unencrypted or encrypted with keys previously learned by the adversary). For symmetric encryption, this is different, since here keys can be sent over the network (usually encrypted with a public key). Hence for symmetric encryption, we demand that for a rule $R \Rightarrow S$ of a principal, if R contains a subterm $\{t_1\}_x^s$, where x is a variable, then x must appear on the left-hand side of a previous rule in the tree. This formalizes that in the state where the rule $R \Rightarrow S$ is applied, the principal i "knows" which key it will use to decode the message. This condition is necessary, since otherwise a rule $\{x\}_y^s \Rightarrow x$ could be used to match with any incoming ciphertext, where the recipient learns both the symmetric key and the message. Such a protocol is clearly not implementable in polynomial time if the used symmetric encryption algorithm is reasonably secure. (If a principal in a protocol is supposed to learn both a secret key and a message encrypted with the secret key, this message can be split up into two messages.) Similarly, if a term $\mathsf{sig}_k(t)$ is in the right-hand term of a rule of a principal, we demand that the principal owns the key k. Note that these rules are sufficient to ensure that protocols can be implemented in polynomial time, but clearly, these can be generalized (for example, one could formalize that principals are able to "learn" secret keys for asymmetric encryption that they recieve over the network). Our result holds for all variants of this definition that ensure that the protocols can be executed in polynomial time.

We call protocols that satisfy the restrictions introduced above *executable symbolic protocols*.

3.2 Computational Realizations of Symbolic Protocols

In a concrete implementation, the cryptographic primitives (a signature scheme, asymmetric and symmetric encryption, and nonce generation) have to be instanciated with algorithms, and symbolic principals have to be simulated by Turing machines. For the realization of symbolic protocols, we fix a signature scheme resistant against existential forgery (i.e., a polynomial-

time adversary equipped with a signing oracle has only a negligible chance of constructing a pair (m, s) where s is a valid signature for m that has not been obtained from the signature oracle), and asymmetric and symmetric encryption schemes that are IND-CCA secure (that is, when the adversary can perform an adaptive chosen ciphertext attack, its probability of guessing correctly whether some string is the encryption of a plaintext chosen by the adversary or a random string of the same length, is only neglibibly higher than one half). We also assume that all keys and nonces are generated randomly and independantly of each other, and all have the same bitlength (which depends on the security parameter).

In order to process messages from the term algebra by Turing machines, we fix a representation of these messages as bit-strings, and assume that our Turing machines can generate and parse this representation. The representation is as follows: We represent messages of the form $\{t\}^a_k$ with a bitstring that encodes a triple $(\texttt{aenc}, k, m)$, where `aenc` is a label that specifies that the term is the result of an asymmetric encryption, k is the public key used for the encryption, and m is the actual bitstring obtained from calling the encryption algorithm with the bitstring representing t and the value of k as input. In particular, although m is a subterm of $\{m\}^a_k$, the bitstring representing m is not readable from the bitstring representation of $\{m\}^a_k$. This expresses that it is not possible to extract m from $\{m\}^a_k$ without access to k^{-1}. Symmetric ciphertexts are modeled as pairs $(\texttt{senc}, m)$, where m is the encrypted message (obviously, there is no information about the used key). Nonces are modeled as triples $(\texttt{nonce}, N, m)$, where N is the name of the nonce, and m is the actual randomly-generated bitstring. We deal with signatures in a very similar way: A representation of a term $\mathsf{sig}_m(k)$ is a bitstring of the form $(\texttt{sig}, k, m, s)$, where k is the public key, m is the bitstring representation of the message, and s is the actual signature obtained from the signature algorithm.

In the following, we identify representation and messages from the algebra, hence our Turing machines process terms from the algebra directly. Note however, that string representations are only unique modulo randomness used in the application of cryptographic primitives.

We now define the computational system that executes a given symbolic protocol $P = ((\Pi_1, \ldots, \Pi_n), \{\texttt{init}\})$: We construct the following system of ITMs, which we call the *computational realization of* P.

- For each principal Π_i, there is an ITM M_i (called *principal machine*) that simulates Π_i as follows: it maintains a *current node* v_i in the tree Π_i and a *current substitution* σ. When receiving a message m over the network or the secure channel, or the empty string (when

activated by the scheduler without an incoming message), it determines a matching outgoing edge (v_i, v_i'), obtaining a new substitution σ'. Note that matching may include signature verification and decryption. In the case that there are several matching edges, one is picked with help of the scheduler (which communicates with principals directly using their I/O tapes). Now M_i hands the string representation of $S\sigma'$ to a secure channel machine and/or the adversary, depending on whether $S\sigma'$ is a secure channel message or not. To compute the representation of $S\sigma'$, cryptographic primitives may be called, including generating a random number when a nonce symbol appears for the first time. The new current state of the principal is v_i', its new current substitution is σ'. If an incoming message cannot be matched, then M_i from now on ignores all adversary-delivered messages, but still accepts messages from the secure channel and ϵ-messages from the scheduler.[4]

- Secure Channel machines: For each pair (Π_i, Π_j) of principals, we introduce a secure channel machine $\mathbf{RC}_j^i$ which acts as described in Section 2.2.

With $P(\mathcal{A}, \mathcal{S})$, we denote the computational realization of P when run with adversary $\mathcal{A}$ and scheduler $\mathcal{S}$, where the cryptographic primitives satisfy the above-mentioned security requirements. Note that if P is an executable symbolic protocol, $\mathcal{A}$ is a probabilistic polynomial-time adversary, and $\mathcal{S}$ a polynomial-time scheduler, then $P(\mathcal{A}, \mathcal{S})$ runs in probabilistic polynomial time in the security parameter: In computational realizations of symbolic protocols, each ITM is activated a constant number of times only, and the length of the messages is bounded by a polynomial in the security parameter.

We now establish a correspondance between symbolic and computational states. The configuration of an (inactive) principal machine is determined by its current node and substitution. The scheduler has access to the configuration of all machines, and does not have a relevant configuration on its own. Hence, a computational state is defined by:

- Current node and substitution of the principal machines,

[4]This behaviour mirrors the symbolic `fail` command introduced earlier. We require this, since otherwise a computational adversary can learn something from the fact that a message was not accepted, and get partial information about the keys. More formally, if the principal would continue its operation after recieving a non-matchable message, the computational realization would not realize the protocol with graph $\mathcal{G}_P$, but a protocol which extends $\mathcal{G}_P$ with certain self-loops. In particular, in such a protocol an unbounded number of principal actions can be performed, which is a significant difference to the protocols in the symbolic model.

- sequence of messages sent so far by the principals,
- messages stored in the secure channel machines,
- randomness used by the principals and the adversary,
- local configuration of the adversary machine.

For a fixed choice of randomness used so far, the encoding of a message as a bitstring is unique. Hence the state in the above sense is uniquely determined if we consider the messages sent and on the secure channel as terms from the term algebra. The adversary state is uniquely determined by the system properties observable by the adversary, i.e., its initial configuration, adversary randomness used so far, and replies sent by the principals (as reaction to message or secure channel delivery). Hence the last item in the above list is redundant, the computational state is uniquely determined by the first four. The first three of these directly constitute a symbolic state (since each principal uses different variables, the local substitutions of each principal correspond to a unique global substitution). We say that two computational states are *equivalent* if they only differ in the random coins used by principals and adversary, and identify symbolic states and equivalence classes of computational states. We write elements of the equivalence class corresponding to the symbolic state q as pairs $(q, \overrightarrow{r})$, where $\overrightarrow{r}$ is the sequence of random coins used in the computational realization up to that state. Note that not every state $(q, \overrightarrow{r})$ is consistent: The randomness $\overrightarrow{r}$ might make the adversary $\mathcal{A}$ choose a strategy that never reaches a state from the class q. We will ignore this technicality, and when we say "a randomly chosen state from the class q," we mean a state of the form $(q, \overrightarrow{r})$ that is consistent, i.e., such that there is a scheduler $\mathcal{S}$ such that when $P(\mathcal{A}, \mathcal{S})$ is started and the random coins used by the principals and the adversary are $\overrightarrow{r}$, then the protocol reaches a state from the class q.

A run of the computational realization $P(\mathcal{A}, \mathcal{S})$ induces a sequence of vertices in $\mathcal{G}_P$: A *step* in such a run is the (scheduler-triggered) action of either a secure channel machine, the adversary, or a principal (with an ϵ-transition), plus the following action of the involved principal. After the principal has finished its computation (with sending a reply message), control is returned to the scheduler and the protocol is in a well-defined symbolic state. Note however, that a step (q_1, q_2) in the execution of $P(\mathcal{A}, \mathcal{S})$ is not necessarily an edge in $\mathcal{G}_P$, since the computational adversary can perform actions not available to the symbolic one. However, we will show in the following Lemma 3.1 that, with overwhelming probability, the steps in an execution of $P(\mathcal{A}, \mathcal{S})$ do correspond to edges in $\mathcal{G}_P$.

3.3 Mapping Protocol Executions

The proof of the following lemma uses similar ideas to other proofs for related results, see e.g., [CH06]. The main difference is that our lemma states a result about a concrete computational model, and not about a hybrid model using ideal functionalities. As a consequence, we handle randomness directly in the proof of the lemma. Note that this proof is the only place where we actually need the security properties of our cryptographic primitives—this is not surprising, since we only need these to ensure that the adversary in the computational model is not more powerful than the one in the symbolic model.

Lemma 3.1 *Let P be an executable symbolic protocol, let $\mathcal{A}$ be an adversary, and $\mathcal{S}$ be a scheduler. Then with overwhelming probability, every step (q_1, q_2) in a run of $P(\mathcal{A}, \mathcal{S})$ started in a randomly chosen state from an equivalence class q is an edge in $\mathcal{G}_P$.*

Proof. First note that when considering computational realizations of symbolic protocols, we can restrict ourselves to schedulers that behave correctly in a syntactic way: We assume that a scheduler only delivers messages from secure channels to principals that, in their current state, can match these message with an applicable rule. This restriction is justified as follows: Since the scheduler has complete knowledge of the principal machines, it knows before delivering a message whether the principal will be able to match it. The scheduler does not gain anything from deliberately sending a non-matching message to the principal, since this cannot make the principal simulating the symbolic protocol change its state. Hence the only consequence of such a delivery would be that the message is removed from the current set of secure channel messages. Since the scheduler always has the option to never deliver a message on the secure channel, leaving the message on the secure channel does not limit its abilities. Similarly, the scheduler does not have any reason to send ϵ-messages to participants who do not have an outgoing ϵ-transition in the current state, hence we only consider schedulers which do not do this. Note that we cannot make a similar restriction for the adversary, since the adversary does not have access to the internal state of principal machines, and hence can gain information by observing whether an incoming message has been accepted. Deliveries by the adversary which cannot be matched will be dealt with by using the `fail` command in the symbolic model and the corresponding behaviour of the computational realization as introduced in Section 3.2. One could also make the principals treat an incoming "unmatchable" message on the secure channel in the same way as an incoming message from the adversary that cannot be matched, i.e., let

the principals go into a special state where some messages (either network or scheduler messages) are not accepted anymore. However, this is not natural in the models we consider: If the scheduler could, via delivering an unmatchable secure channel message, stop the principals from accepting adversary (network) messages in the future, this would give too much power to the scheduler: The purpose of the scheduler is to schedule the ordering of events that happen in the protocol, not to "disable" participants. (If one wants to equip the scheduler with such a power, this can be obtained by adding corresponding ϵ-transitions to the protocol principals). For the other alternative, there is no point in allowing the scheduler to make a principal go into a state where it does not accept incoming scheduler messages anymore—the scheduler can achieve the same effect by simply not delivering such messages to the principal in question anymore. Hence the only natural choice for our model is to assume that unmatchable messages from the secure channel do not have an effect on the principals, and thus, as explained above, we can assume that these events do not occur.

For the proof of the lemma, let $\overrightarrow{r_1}, \overrightarrow{r_2}$ be the randomness used by the principals and the adversary in the computational system in the computation up to states q_1 and q_2, i.e., let the computational states corresponding to q_1 and q_2 be $(q_1, \overrightarrow{r_1})$ and $(q_2, \overrightarrow{r_2})$, respectively. We make a case distinction depending on which action (triggered by the scheduler) lead from $(q_1, \overrightarrow{r_1})$ to $(q_2, \overrightarrow{r_2})$ in the execution of $P(\mathcal{A}, \mathcal{S})$.

Case 1: Secure Channel machine Assume that the action between $(q_1, \overrightarrow{r_1})$ and $(q_2, \overrightarrow{r_2})$ was the delivery of a secure channel message m to some principal with name i. As explained above, we can assume that the principal was able to match the message with a rule applicable in its current state. Hence for its current node v_i and the current substitution σ_i, there exists an edge (v_i, v_i') such that $\ell(v_i, v_i') = R \Rightarrow S$, and there is a substitution σ_i', which agrees with σ_i on the domain of the latter, such that $R\sigma_i' = m$, and $S\sigma_i'$ is the message sent over the network or the secure channel by M_i. Thus q_2 is obtained from q_1 by the principal with name i changing its current node to v_i', extending its substitution to σ_i', removing m from the secure channel, and adding $S\sigma_i'$ to the adversary knowledge (and potentially the secure channel). This is an action also allowed by the symbolic protocol, and thus there is an edge (q_1, q_2) in $\mathcal{G}_P$ as required.

Case 2: Principal machine without incoming message Assume that the principal i was activated by the scheduler machine, without an incoming message. Since we only consider schedulers which send ϵ-messages only to principals expecting them, the current node v_i of the

principal i has an outgoing edge (v_i, v_i') such that $\ell(v_i, v_i')$ is of the form $\epsilon \Rightarrow S$, where $S\sigma_i$ is the message that i sent over the network, and σ_i is the current substitution of the principal i. Thus q_2 is obtained by the principal i changing its current node to v_i', and adding $S\sigma_i$ to the adversary knowledge (and the secure channel, if it is a secure channel message). This is a valid action in the symbolic protocol, and hence there is an edge (q_1, q_2) in $\mathcal{G}_P$.

Case 3: Activation of the adversary Now assume that the adversary was activated, and it delivered a message m to the principal i. First assume that the principal could not match the incoming message with an applicable role. By the definition of the computational realization, this means that the principal machine changed into a mode where it does not accept messages from the adversary anymore. The same effect can be achieved by the symbolic adversary by sending the special constant `fail` to the principal as introduced in Section 3.1. Hence in this case, there is an edge (q_1, q_2) in $\mathcal{G}_P$ as claimed. Now assume that the principal was able to match the message m in the tree Π_i. The proof of this case is identical to the case of a secure channel message above, except that we need to show that the delivered message m is an element of $d(\mathcal{I}(q_1))$, i.e., that the symbolic adversary is able to derive the message that the computational adversary sends. We show that the case $m \notin d(\mathcal{I}(q_1))$ occurs only with negligible probability.

In order to prove this, we first show that $\mathcal{A}$ has only negligible propability of producing the bitstring representation of a nonce or a key that it cannot derive from its current knowledge. For this, we say that the computational adversary $\mathcal{A}$ *can access* a key or a nonce α in a state $(q, \overrightarrow{r})$, if it has a probability of distinguishing α from a random string of the same length that non-neglibibly exceeds one half (remember that all keys and nonces have the same bitlength, which depends on the security parameter). We now show the following claim, which proves that a key or a nonce that is secret in the symbolic model is also secret in the computational model.

Claim *Let* α be a key or a nonce which is not an element of $d(\mathcal{I}(q))$ in a state q. Then $\mathcal{A}$ cannot access α in a randomly chosen state from the class q.

Proof. of the claim We first introduce some additional notation. We say that a set S of keys and nonces *reveals* a key or a nonce α in a

state q, if $\alpha \in d(\mathcal{I}(q) \cup S)$, and S is a minimal set with this property. In particular, this implies $S \cap d(\mathcal{I}(q)) = \emptyset$. Note that S is not uniquely determined by $\mathcal{I}(q)$: If, for example $\mathcal{I}(q) = \left\{\{\alpha\}^s_{k_1}, \{\alpha\}^s_{k_2}\right\}$, then both $\{k_1\}$ and $\{k_2\}$ reveal α in q. We say that α is *secret* in a state q if $\alpha \notin d(\mathcal{I}(q))$ (this is equivalent to asking that $\emptyset$ does not reveal α). For two secret keys or nonces α and β, we write $\beta \rightarrow_q \alpha$ if there is a set S such that S reveals α in q, and $\beta \in S$, i.e., if the knowledge of β "helps" the adversary to obtain α. Note that $\rightarrow$ is a reflexive relation. We further define the relation $\twoheadrightarrow$ to be the transitive closure of $\rightarrow$, and let $\leq$ be the partial order defined on the equivalence classes induced by $\twoheadrightarrow$.

Now assume that the claim does not hold. Then let β be a $\leq$-minimal key or nonce such that $\beta \leq \alpha$, and $\mathcal{A}$ can access β (in particular, due to the notation introduced above, this implies that β is secret). Since $\beta \leq a$ only holds for a finite number of keys and nonces β, and α itself satisfies the conditions except for the minimality, such a minimal β exists. Due to the definitions of the relations above, it follows that β was only sent over the network encrypted with keys that $\mathcal{A}$ cannot distinguish from random noise. Hence if $\mathcal{A}$ can distinguish β from random noise, $\mathcal{A}$ has won the IND-CCA-game against the encryption scheme. This is a contradiction, since due to our prerequisites, both the asymmetric and the symmetric encryption scheme are IND-CCA-secure. □

With this property, we can now finish the proof of Lemma 3.1. As mentioned above, it remains to prove that the case $m \notin d(\mathcal{I}(q_1))$ occurs only with negligible probability. Hence assume that $m \notin d(\mathcal{I}(q_1))$. Let $R \Rightarrow S$ be the rule chosen by the principal Π_i on receiving the message m, and let σ' be the substitution obtained by the matching. Since $R\sigma' = m$, we know that in particular, m is a bitstring representing a message from the term algebra. Hence we can regard m as a tree in the algebra. Let v be a node in m such that $v \notin d(\mathcal{I}(q_1))$ (we identify v with the term it represents), but all successors of v (corresponding to subterms of v) are members of this set. We make a case distinction.

First consider the case that v is a leaf, i.e., a constant, and assume that v has not been transferred over the network. Since names of the principals are public (and hence have been transferred over the network in the initialization phase generating the initial knowledge of the adversary), this constant must be a nonce which is not an adversary constant. Hence the adversary has successfully guessed a nonce created

by a principal, this event only happens with negligible probability. Now assume that v has been transferred over the network. Since $v \notin d(\mathcal{I}(q))$, due to the above claim, $\mathcal{A}$ has a chance that is negligibly better than one half to distinguish v from a random string. In particular, the probability that $\mathcal{A}$ generates v is negligible.

Now consider the case that v is not a leaf. If v represents the pairing of two terms, then we have a contradiction, because by choice of v, both successors of v are elements of $d(\mathcal{I}(q_1))$, and this set is closed under the pairing operator. Now assume that v is the asymmetric encryption $\{t\}_k^a$ of a term t under a key k. Due to the choice of v, it follows that both t and k are elements of $d(\mathcal{I}(q_1))$, and thus $\{t\}_k^a$ is also an element of $d(\mathcal{I}(q_1))$, a contradiction. Now assume that v is the symmetric encryption $\{t_1\}_\alpha^s$ of a message t_1, with the key or nonce α used as a symmetric encryption key. Due to the minimality of v, we know that both t_1 and α are elements of $d(\mathcal{I}(q_1))$. Since $d(\mathcal{I}(q_1))$ is closed under symmetric encryption, it follows that $v \in d(\mathcal{I}(q_1))$, a contradiction. Finally assume that the subterm at v is of the form $\mathsf{sig}_k(t)$ for a term t and a key k. Since all successors of v are elements of $d(\mathcal{I}(r_1))$, and t is a subterm of $\mathsf{sig}_k(t)$, it follows that $k^{-1} \notin d(\mathcal{I}(q_1))$ (otherwise, v would be an element of $d(\mathcal{I}(q_1))$). Hence k^{-1} is secret in q_1, and due to the above claim, the adversary's chance of distinguishing k^{-1} from random is neglibibly better than one half. Thus $\mathcal{A}$ has constructed a correct signature without access to the private key—since the signature scheme is resistant against existential forgery, this only happens with negligible probability.

Hence all cases where $m \notin d(\mathcal{I}(q_1))$ occur with negligible probability, and therefore $m \in d(\mathcal{I}(q_1))$ with overwhelming probability, as claimed. This finishes the proof of the lemma. □

The above lemma ensures that, with overwhelming probability, a run of the computational system induces a path in the protocol graph $\mathcal{G}_P$. One consequence of this result is that with overwhelming probability, the computational adversary does not "break" the cryptographic primitives. Hence we can restrict ourselves to adversaries that do not even attempt that. In the face of such an adversary, the strength of the implementation of the cryptographic primitives becomes irrelevant. In particular, this allows us to exchange the implementation of the cryptographic primitives with one that does not use randomness at all, *without* giving more power to the adversary. Hence we can fix one choice of random coins for the principals, and thus each equivalence class of computational states only contains a single state (except that we need to be careful about the randomness used by the adversary

outside of cryptographic primitives, e.g., to randomly decide which symbolic step to perform). We therefore obtain a real bijection between the states of the symbolic and of the computational system. This idea is formalized in the proof of Theorem 4.2.

4 Branching-Time Behaviour

We now show that the computational realization of symbolic protocols fulfill the same security requirements as their symbolic counterparts. Section 4.1 covers the "reachability" part of the result, Section 4.2 then proves that strategies can be transformed from one model to the other.

4.1 Reachability

We now state the first part of our transfer result, which concerns the case of reachability: A protocol can reach a state q in the symbolic model if and only if this state can be reached in the computational model with non-negligible probability. This first part is similar to previous results on trace properties: For example, the question whether a protocol preserves secrecy of a value can be phrased as the question whether some state (or set of states) is reachable. We first define what we mean with "reachable," in the computational and in the symbolic model:

Definition Let $P = ((\Pi_1, \ldots, \Pi_n), \mathcal{I})$ be a protocol, and let q_1 and q_2 be states of P. We say that

- q_2 is *symbolically reachable from* q_1 *in* P if there is a path from q_1 to q_2 in $\mathcal{G}_P$,
- q_2 is *computationally reachable from* q_1 *in* P if there are an adversary $\mathcal{A}$ and a scheduler $\mathcal{S}$ such that when $P(\mathcal{A}, \mathcal{S})$ is started in a randolmy chosen state from q_1, the probability of it reaching a state in q_2 is non-negligible (in the security parameter).

We now show that these reachability notions are equivalent. The result easily follows from the previous Lemma 3.1, which already established that runs of the symbolic protocol and its computational realization coincide with overwhelming probability.

Theorem 4.1 *Let P be a protocol, and let q_1 and q_2 be states in $\mathcal{G}_P$. Then q_2 is computationally reachable from q_1 in P if and only if q_2 is symbolically reachable from q_1 in P.*

Proof. First assume that q_2 is symbolically reachable from q_1 in P. We construct an adversary $\mathcal{A}$ and a scheduler $\mathcal{S}$ such that when $P(\mathcal{A}, \mathcal{S})$ is started in a state $(q_1, \overrightarrow{r_1})$, the probability of reaching a state from the class q_2 is non-negligible (where $\overrightarrow{r_1}$ is chosen randomly).

We define $\mathcal{A}$ and $\mathcal{S}$ as basically "mimicing" the path from q_1 to q_2 that exists in $\mathcal{G}_P$. This means that at each state on the path in $\mathcal{G}_P$, depending on the next edge in the path, either the scheduler activates the adversary, which delivers a message (and the scheduler makes the principal choose the matching edge corresponding to the path in $\mathcal{G}_P$, if there is more than one choice), the scheduler triggers the delivery of a secure channel message, or of an ϵ-transition of a principal (i.e., it activates a principal without an incoming message). Since on the path in $\mathcal{G}_P$, all adversary-sent messages are elements of the current adversary knowledge, the polynomial-time adversary can compute all these messages. Note that since adversary and scheduler follow a symbolic path, the randomness used by principals and adversary does not influence the success probability of the adversary: The adversary does not "guess" a message which he does not have access to, or forge signatures without having the corresponding secret key. Therefore, the adversary and the scheduler do not have to make any random choices (since the path from q_1 to q_2 is finite, the exact sequence of actions to perform can be hard-coded into the algorithms $\mathcal{A}$ and $\mathcal{S}$). Hence we can identify symbolic and computational states. In particular, since the symbolic protocol reaches the state q, the computational realization reaches a state of the form $(q_2, \overrightarrow{r_2})$ with probability 1. In particular, q_2 is computationally reachable from q_1 in P.

Now assume that q_2 is computationally reachable from q_1 in P, i.e., there is an adversary $\mathcal{A}$ and a scheduler $\mathcal{S}$ such that for a randomly chosen vector $\overrightarrow{r_1}$, when $P(\mathcal{A}, \mathcal{S})$ is started in $(q_1, \overrightarrow{r_1})$, the probability of reaching a state from the class q_2 is non-negligible. We show that there is a path from q_1 to q_2 in $\mathcal{G}_P$. From Lemma 3.1, we know that with overwhelming probability, the sequence of states visited by the run of $P(\mathcal{A}, \mathcal{S})$ corresponds to a path in $\mathcal{G}_P$. Hence there must be such a path from q_1 to q_2 in $\mathcal{G}_P$, as claimed. □

4.2 Strategies

In the previous section, we have seen that reachability is equivalent in the computational and the symbolic models, i.e., an adversary (with the help of a scheduler) can reach a state in one model if and only if it can do this in the

other. For security properties of protocols that rely on the set of reachable states alone (like secrecy or mutual authentication), this is enough to show that security in the symbolic and in the computational model coincide. In the case of more complex security properties that we are interested here, this is not sufficient: We have to prove that reaching a state is as "useful" to the adversary in the computational model as it is in the symbolic one. In the case of contract signing protocols, this means that we want to show that a state is "unbalanced" (cp. Introduction) symbolically if and only if it is unbalanced computationally. For contract signing protocols, there are two goals that the adversary wants to be able to reach: Having a contract himself, and preventing another principal from obtaining a contract. More generally, a *goal* is a pair (C, C'), where C and C' are sets of atoms. A symbolic state q *satisfies* (C, C'), if $C \subseteq d(\mathcal{I}(q))$, and $C' \cap d(\mathcal{I}(q)) = \emptyset$. In the example of contract signing protocols, one can specify the protocol in such a way that a special nonce is sent over the network (by the principal or the trusted third party) when a principal or the adversary obtains a contract. Hence goals like "the adversary obtains a contract" or "the principal i does not have a valid contract" can be defined in this way.

For a state q in the protocol P and a goal (C, C'), a *symbolic q-strategy for (C, C') in P* is a subgraph $\mathcal{H}$ of $\mathcal{G}_{P,q}$ obtained by removing adversial edges and nodes then unreachable from q such that every leaf in $\mathcal{H}$ satisfies (C, C'). Intuitively, such a strategy tells the symbolic adversary which action to perform in each state. We are only interested in leaves here, because the adversary has only achieved a goal of the form "a principal does not have a valid contract" if that principal cannot obtain the contract anymore in the protocol run (otherwise such a goal would always be reached at the beginning of a protocol). Since we only remove adversial edges, a leaf in $\mathcal{H}$ is a state in which no principal can perform an ϵ-transition, or receive a message from the secure channel anymore. The only party that still can perform an action is the adversary, and the strategy $\mathcal{H}$ might tell the adversary not to do so. Hence it is allowed to have leaves in $\mathcal{H}$ which are no leaves in $\mathcal{G}_P$.

In the computational world, a strategy is also a "set of instructions" for the adversary. Unlike in the first phase of the protocol where the adversary tries to reach a certain state with the help of the scheduler, a computational strategy has to work against all possible schedulers, that is, for all possible non-deterministic choices of the principals, and all possible schedules of the secure channel messages. Formally, we say that a *computational q-strategy for (C, C') in P* is an adversary $\mathcal{A}$ such that for every scheduler $\mathcal{S}$, the probability that $P(\mathcal{A}, \mathcal{S})$, when started in a randomly chosen state from the class q, ends in a state satisfying (C, C') is non-negligible.

Again, we do not require the protocol to end in a leaf of $\mathcal{G}_P$, due to the

same reason as above: The adversary might choose not to perform an action anymore, although it could (like delivering a signed contract to a principal). Since we demand that the protocol run ends in a state satisfying the goal for *all* schedulers, this implies that no principal has an outgoing ϵ-transition, and no secure channel message can be delivered that makes the protocol reach a state not satisfying the goal, i.e., the principals and the schedulers cannot perform any relevant action anymore. Note that in [CKW07], the issue of how to ensure that every principal who still wants to perform an action has the opportunity to do so was adressed by using *fair schedulers*. Since in the current paper, we are interested only in procotols which induce a transition graph of finite depth, it suffices to restrict ourselves to nodes that do not have outgoing scheduler edges (since these are exactly those states in which none of the principals can still "take an action" in the sense of [CKW07]—again, the adversary might choose not to perform an action if it has achieved the goal in the current state). Also note that in the mentioned paper, goals were described by polynomial-time functions from the configurations of the involved Turing machines. It is easy to see that for the class of protocols we consider here, these notions of goals are equally expressive.

The fact that a state allows a stragegy for some goal is not considered harmful for a security protocol, since these goals might very well be some that the protocol is supposed to obtain (for example, in a contract signing protocol, every leaf where a principal obtained a contract also has a (trivial) strategy for that principal to obtain the contract). What a protocol should avoid is for a state to allow strategies for two *different* goals, since this would allow the adversary to unilaterally determine the outcome of a protocol. We call states which allow such strategies *unbalanced* with respect to a set of goals.

Formally, we say that a symbolic state q in the protocol P is (computationally or symbolically) *unbalanced* for a set of goals G, if for every goal (C, C') in G, there is a (computational or symbolic) q-strategy for (C, C') in P. It can easily be verified that this definition of computational unbalancedness is equivalent to the definition of unfairness in [CKW07]. We show the following theorem:

Theorem 4.2 *Let P be a protocol, and G a set of goals. Then a state q in P is symbolically unbalanced for G if and only if it is computationally unbalanced for G.*

Proof. Clearly it suffices to show that there is a symbolic q-strategy for a goal (C, C') in P if and only if there is a computational q-strategy for (C, C') in P.

First assume that there is a symbolic strategy $\mathcal{H}$. The computational strategy we construct simply mimics the symbolic one, which can be done since all messages sent by the symbolic adversary can be obtained from the messages sent by the honest principals in polynomial time. To be more formal, in a symbolic state q which is not a leaf, the computational adversary chooses (arbitrarily) one of the edges available in $\mathcal{H}$, and sends the corresponding message (which it can derive from its current knowledge, since $\mathcal{H}$ is a subgraph of $\mathcal{G}_P$). When the protocol reaches a leaf, this state satisfies (C, C') due to the choice of $\mathcal{H}$. It only remains to prove that the adversary can always guess the correct equivalence class of the current state of the protocol with high probability.

In order to prove this, note that if the adversary follows a fixed strategy, then the relevant part of the protocol graph $\mathcal{G}_P$ is only finitely branching, since the principals only have a finite number of nondeterministic choices in each step. Since the number of steps executed is bounded by a constant, the adversary can guess in each stage of the execution which of the (finitely many) possible symbolic states the execution is in. Since there are only finitely many guesses that the adversary needs to perform, and it has a non-negligible probability of guessing correctly in each step, the probability that it guesses correctly every time—and thus follows the symbolic strategy $\mathcal{H}$ correctly, thereby reaching a leaf satisfying the goal (C, C')—is non-negligible. Hence the adversary $\mathcal{A}$ constructed in this way is a computational q-strategy for the goal (C, C') in P.

Now assume that there is a computational q-strategy for (C, C') in P, i.e., an adversary $\mathcal{A}$ such that for all schedulers $\mathcal{S}$, the system $P(\mathcal{A}, \mathcal{S})$, when started in a randomly chosen state of the form $(q, \overrightarrow{r})$, reaches a leaf satisfying (C, C') with non-negligible probability for all possible schedulers $\mathcal{S}$. From Lemma 3.1, we know that with overwhelming probability, every step in a protocol run of $P(\mathcal{A}, \mathcal{S})$ corresponds to an edge in $\mathcal{G}_P$. This implies that the adversary can be replaced with one that does not attempt to guess a nonce created by the principals or decrypt a message where the decryption key is not available to the adversary, or forge a signature without access to the secret key, and this adversary still has non-negligible success probability. In particular, this means that the success probability of the adversary does not depend on the implementation of the cryptographic primitives used by the principals. Hence we can replace this implementation with one that does not use randomness, without changing the success probability of the adversary. We can therefore assume that only the adversary uses randomness.

We say that a state $(v, \overrightarrow{r_v})$ is (C, C')-*enabling* if when $P(\mathcal{A}, \mathcal{S})$ is started in $(v, \overrightarrow{r_v})$, the probability of ending in a leaf satisfying (C, C') is nonzero for all possible schedulers. In particular, due to the choice of q, there is a

randomness vector $\overrightarrow{r_q}$ such that $(q, \overrightarrow{r_q})$ is (C, C')-enabling. We define the symbolic strategy $\mathcal{H}$ by defining a series of subgraphs $\{\mathcal{H}_i\}_{n\in\mathbb{N}}$ of $\mathcal{G}_{P,q}$. We will also define, for every state v in $\cup_i \mathcal{H}_i$, a suitable randomness vector $\overrightarrow{r_v}$ such that the state $(v, \overrightarrow{r_v})$ is (C, C')-enabling. For $i = 0$, $\overrightarrow{r_q}$ has been defined above. Now assume inductively that $\mathcal{H}_i$ and $\overrightarrow{r_v}$ for all $v \in \mathcal{H}_i$, have been defined with this property. We construct the next partial strategy $\mathcal{H}_{i+1}$ in two steps: We first add scheduler edges, and in a second step add adversial edges. Since we want to construct a symbolic strategy, we need to add *all* possible scheduler successors, but can choose which of the adversial actions to add.

For this, let $\mathcal{H}'_i$ be the graph obtained from $\mathcal{H}_i$ by adding all symbolic states that can be reached from a state in $\mathcal{H}_i$ in $\mathcal{G}_P$ using only scheduler edges, and the scheduler edges leading to these vertices. For every state v' in $\mathcal{H}'_i \setminus \mathcal{H}_i$, let the corresponding randomness vector $\overrightarrow{r_{v'}}$ be the same as the randomness from one of the states in $\mathcal{H}_i$ from which v can be reached using only scheduler edges. We show that for every node $v' \in \mathcal{H}'_i$, the state $(v', \overrightarrow{r_{v'}})$ is (C, C')-enabling. Let v' be a node from $\mathcal{H}'_i$. By choice of v' and $\overrightarrow{r_{v'}}$, v' can be reached from some node v in $\mathcal{H}_i$ for which $\overrightarrow{r_{v'}} = \overrightarrow{r_v}$, on a path from v to v' which uses only scheduler edges. Therefore, there exists a scheduler $\mathcal{S}_{v'}$ which follows this path, i.e., when $P(\mathcal{A}, \mathcal{S}_{v'})$ is started in $(v, \overrightarrow{r_v})$, the execution reaches a state from the class v'. Due to the above, only $\mathcal{A}$ uses randomness. Hence no random choices have been made between v and v', and therefore, the execution reaches the state $(v', \overrightarrow{r_v})$. Since $(v, \overrightarrow{r_v})$ is (C, C')-enabling, the probability of $P(\mathcal{A}, \mathcal{S})$ reaching a leaf satisfying (C, C') when started in $(v, \overrightarrow{r_v})$ with an arbitrary scheduler $\mathcal{S}$ (including $\mathcal{S}_{v'}$) is non-zero. Since no random choices were made between v and v', the adversary $\mathcal{A}$ still has non-negligible probability of reaching a goal satisfying (C, C') when the computational realization of P is started with any scheduler and $\mathcal{A}$ in state $(v', \overrightarrow{r_v})$. Hence we can define $\overrightarrow{r_{v'}} = \overrightarrow{r_v}$, and $(v', \overrightarrow{r_v}) = (v', \overrightarrow{r_{v'}})$ is (C, C')-enabling as required.

In the second step, we add the adversary actions (and here we have a choice which successors to add). First define S_i to be the set of leaves in $\mathcal{H}'_i$. We will define a function $f_i \colon S_i \to \mathcal{G}_{P,q}$, which defines, for each $s \in S_i$, which successor $f_i(s)$ (representing the adversarial action to be taken in that state) to add in the next step. First assume that the symbolic state $s \in S_i$ satisfies (C, C'). Since s is a leaf in $\mathcal{H}'_i$, this implies that s does not have any outgoing scheduler-edges in $\mathcal{G}_{P,q}$. Hence the suitable strategy for the adversary is to do nothing, and remain in this state. We therefore define $f_i(s) = s$. Now assume that s does not satisfy (C, C'). Since $(s, \overrightarrow{r_s})$ is (C, C')-enabling, it follows that s has a descendant in $\mathcal{G}_P$ which satisfies (C, C'). In particular, s is no leaf in (C, C'). Since s is a leaf in $\mathcal{H}'_i$, and $\mathcal{H}'_i$ contains all of the outgoing

scheduler edges for all nodes it contains, we know that all outgoing edges of s in $\mathcal{G}_P$ are adversial edges. Due to the construction of $\mathcal{H}'_i$, we know that $(s, \overrightarrow{r_s})$ is (C, C')-enabling. In particular, there is an adversial action that $\mathcal{A}_s$ can perform in the state $(s, \overrightarrow{r_s})$ such that when reaching the successor state s', it still has a non-zero success probability against all schedulers. Since s does not satisfy (C, C'), we can choose s' with $s \neq s'$. Let $\overrightarrow{r_{s'}}$ be the random choices from $\overrightarrow{r_s}$ extended with the random choice that makes $\mathcal{A}$ perform the action leading to s'. Then, $P(\mathcal{A}, \mathcal{S})$ still has non-zero probability of reaching a leaf satisfying (C, C') when started in $(s', \overrightarrow{r_{s'}})$ for all schedulers $\mathcal{S}$, i.e., $(s', \overrightarrow{r_{s'}})$ is (C, C')-enabling. Note that due to the comments earlier, we can restrict ourselves to adversaries that only perform actions present in $\mathcal{G}_P$. Hence there is an edge $(s, s') \in \mathcal{G}_P$. We define $f_i(s) = s'$.

Note that by the above, for all $s \in S_i$, we have that $(f(s), \overrightarrow{r_{f(s)}})$ is (C, C')-enabling. By construction, this is also true for all $(s, \overrightarrow{r_s})$ for $s \in \mathcal{H}'_i$. Therefore we can define H_{i+1} as having all vertices from $H'_i \cup \{f(s) \mid s \in S_i\}$, and all edges from $\mathcal{H}'_i$, plus all edges of the form $(s, f_i(s))$, where s is an element of S_i and $f_i(s) \neq s$. In this way, we have ensured that for every state $v \in H_{i+1}$, $(v, \overrightarrow{r_v})$ is (C, C')-enabling. Hence $\mathcal{H}_{i+1}$ satisfies the required properties.

By construction, in the step from H_i to H_{i+1} each path is either extended by at least one node, or will not be extended in any future step. Since all paths in $\mathcal{G}_P$ are finite, there is some i such that $\mathcal{H}_i = \mathcal{H}_{i+1}$. We define $\mathcal{H} = \mathcal{H}_i$, and claim that $\mathcal{H}$ is a symbolic strategy for (C, C'). We first need to show that $\mathcal{H}$ is obtained from $\mathcal{G}_{P,q}$ by removing only adversial edges and vertices unreachable from q. Assume that this is not the case, then there is a vertex $v \in \mathcal{H}$ and an edge (v, v') in $\mathcal{G}_P \setminus \mathcal{H}$ which is not an adversarial edge, i.e., a scheduler edge. This is a contradiction, since $\mathcal{H} = \mathcal{H}_{i+1} = \mathcal{H}_i$, and since $\mathcal{H}_i \subseteq \mathcal{H}'_i \subseteq \mathcal{H}_{i+1}$, it also follows that $\mathcal{H} = \mathcal{H}'_i$. The latter is closed under adding successors reachable on paths using only scheduler edges, hence (v, v') is an edge in $\mathcal{H}'_i = \mathcal{H}$, a contradiction.

It remains to show that every leaf in $\mathcal{H}$ satisfies the goal (C, C'). Let s be a leaf in $\mathcal{H}$. Due to the construction, we know that s is an element of S_i. Since s is a leaf in $\mathcal{H}$, we know that $f_i(s) = s$, since otherwise, there is an edge $(s, f_i(s))$ in $\mathcal{H}_{i+1} = \mathcal{H} = \mathcal{H}_i$. Due to the construction, $f_i(s) = s$ only happens in the case that s satisfies (C, C'). Hence every leaf of $\mathcal{H}$ satisfies the goal, and thus $\mathcal{H}$ is in fact a symbolic q-strategy, which concludes the proof. □

5 Conclusion

In the previous section, we have shown that the computational realization of a symbolic protocol P in fact shares (with overwhelming probability) the most important properties of P. This immediately implies that a protocol is "secure" symbolically if and only if it is "secure" computationally, in the following formal way: We say that a protocol P is *(computationally or symbolically) unbalanced* with respect to a set G of goals if there is a state q that is (computationally or symbolically) unbalanced with respect to G and that is (computationally or symbolically) reachable from the starting state of P.

From the results in the previous sections, we obtain the following:

Corollary 5.1 *An executable symbolic protocol P is symbolically unbalanced if and only if it is computationally unbalanced.*

As far as we know, this result is the first result showing that a security property transfers from the symbolic to the computational world which is not a trace property. One of the most important applications of our result (and results that prove equivalence of secutrity in symbolic and computational models in general) is that decidability directly transfers from the symbolic case. Since decidability of symbolic unbalancedness was proven in [KKW05], we immediately obtain the following:

Corollary 5.2 *The question if an executable symbolic protocol is computationally unbalanced is decidable.*

There are many interesting directions for future research. An obvious open question is how to generalize our results to a richer class of protocols (e.g., recursive protocols as introduced in [Tru05]) and to an unbounded number of seesions. Security analysis of symbolic protocols becomes undecidable in this [DLMS04], but by over-approximations the adversary knowledge, automatic analysis of protocols might still be possible, with techniques similar as in [RK06]. Other interesting topics involve the handling of more cryptographic primitives, for example *private contract signatures* as introduced in [GJM99]. To incorporate these into our result, note that the only element of our proof that needs to be generalized is the proof of Lemma 3.1. Finally, it would be very interesting to consider more complex security goals, and study a uniform way to express them in—one candidate is a variant of ATL, used to describe properties of contract-signing protocols in [Käh08].

References

[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.

[BP08] Michael Backes and Birgit Pfitzmann. Limits of the BRSIM/UC soundness of dolev-yao-style xor. *International Journal of Information Security*, 7(1):33–54, 2008.

[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.

[CKRT03] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. Deciding the security of protocols with diffie-hellman exponentiation and products in exponents. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *FSTTCS*, volume 2914 of *Lecture Notes in Computer Science*, pages 124–135. Springer, 2003.

[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.

[CLS03] Hubert Comon-Lundh and Vitaly Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *LICS*, pages 271–280. IEEE Computer Society, 2003.

[DLMS04] Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[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.

[Käh08] Detlef Kähler. *Strategy Properties for Cryptographic Protocols*. PhD thesis, Christian-Albrechts-Universität Kiel, 2008.

[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.

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

[Low96] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Tiziana Margaria and Bernhard Steffen, editors, *TACAS*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 1996.

[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.

[RK06] Andrew W. Roscoe and Eldar Kleiner. Modelling unbounded parallel sessions of security protocols in CSP. 2006.

[RT03] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with a finite number of sessions, composed keys is NP-complete. *Theoretical Computer Science*, 1-3(299):451–475, 2003.

[Shm04] Vitaly Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In David A. Schmidt, editor, *ESOP*, volume 2986 of *Lecture Notes in Computer Science*, pages 355–369. Springer, 2004.

[Tru05] Tomasz Truderung. Selecting theories and recursive protocols. In Martín Abadi and Luca de Alfaro, editors, *CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pages 217–232. Springer, 2005.