Computational Semantics for Basic Protocol Logic – A Stochastic Approach

Gergei Bana *, Koji Hasebe, and Mitsuhiro Okada

[1] Dept of Computer Science, University of California at Davis, Davis, CA, USA
[2] Research Center for Verification and Semantics,
National Institute of Advanced Industrial Science and Technology Osaka, Japan
[3] Department of Philosophy, Keio University, Tokyo, Japan
bana@math.upenn.edu k-hasebe@aist.go.jp mitsu@abelard.flet.keio.ac.jp

Abstract. This paper is concerned about relating formal and computational models of cryptography in case of active adversaries when formal security analysis is done with first order logic. We first present some criticism of the way Datta et al. defined computational semantics to their Protocol Composition Logic, concluding that problems arise from focusing on occurrences of bit-strings on individual traces instead of occurrences of probability distributions of bit-strings across the distribution of traces. We therefore introduce a new, fully probabilistic method to assign computational semantics to the syntax. We present this via considering a simple example of such a formal model, the Basic Protocol Logic of [23] by K. Hasebe and M. Okada, but the technique is suitable for extensions to more complex situations such as PCL. The idea is to make use of the usual mathematical treatment of stochastic processes, hence be able to treat arbitrary probability distributions, non-negligible probability of collision, causal dependence or independence, and so on.

1 Introduction

In the past few years, linking the formal and computational models of cryptography has become of central interest. Several different methods have emerged for both active and passive adversaries. In this paper we would like to consider the relationship of the two models when formal security analysis is done with first order logic. In the formal approach of our interest, protocol correctness is analyzed by defining a syntax with axioms and inference rules and then proving some property. A logical proof then ensures that the property will be true in any formal model (semantics) of the syntax. The link to the computational world then is done by assigning a computational semantics (instead of formal) to the syntax, proving that the axioms and inference rules hold there, and hence a property correct in the syntax must be true in the computational model. However, as it turns out, it is not unambiguous how to define the computational semantics, and when a property should be deemed "true" computationally.

Recently, Datta et al. in [17] gave a computational semantics to the syntax of their Protocol Composition Logic of [19, 16] (cf. also [1] for *a protocol composition logic project overview*). In their treatment, every action by the honest participants is recorded on each execution trace (which have identical probabilities), and bit strings emerging later are checked whether they were recorded earlier and to what action they corresponded (the only actions of the adversary that are recorded are send and receive). This way, they first define whether a property is true on a particular trace, and they say the property is true in the model if it is true on an overwhelming number of traces. This method however relies on negligible collision probabilities, because otherwise there would be a large probability of identifying bit strings with the wrong actions. Moreover, as the comparisons are done on each trace separately it is not possible to track correlations.

* Partially supported by a Packard Fellowship.

Our approach puts more emphasis on probabilities. Instead of defining what is true on each trace, we say that a property is true in the model if a "cross-section" of all traces provides the right probabilities for computational realizations of the property in question. An underlying stochastic structure ensures that we can detect if something depends on the past or does not. It is not coincidences on traces that we look for, but correlations of probability distributions.

We introduce our method on a rather simple syntax, namely, a somewhat modified version of Basic Protocol Logic (or BPL, for short) of [23] by K. Hasebe and M. Okada and leave extensions to more complex situations such as the Protocol Composition Logic to future work. The reason for this is partly the limited space, partly to avoid distraction by an elaborate formal model from the main ideas, but also that a complete axiomatization of the syntax used by Datta et al. for their computational PCL has not yet been published anywhere, only fragments are available. We would like to emphasize though that our point is not to give a computational semantics to BPL but to provide a technique that works well in much more general situations as well.

BPL is a logical inference system to prove correctness of a protocol. Originally, it included signatures as well, but for simplicity, we leave that out from this analysis. BPL was defined to give a simple formulation of a core part of the protocol logics of [19, 16, 14] for proving some aimed properties within the framework of first order logic. It concentrates on several types of agreement properties in the sense of [29, 26]. It does not deal with the secrecy property of nonces or session keys.

We first give the axiomatic system in first-order predicate logic for proving the agreement properties. A message is represented by a first-order term that uses encryption and pairing symbols, an atomic formula is a sequence of primitive actions (as send, receive and generate) of principals on terms. We set some properties about nonces and cryptographic assumptions as non-logical axioms, and give a specific form of formulas, called *query form*, which has enough expressive power to specify our intended authentication properties. The main aspects in which we modified the original BPL is that the original axioms were not all computationally sound so we left out some that are in fact not used in proofs anyway. Furthermore, instead of denoting encryptions as $\{m\}_A$, we have decided to indicate the random seed of the encryption as $\{m\}^r_A$ (as Datta et al. do and also as in [24]) since computation interpretation becomes much easier this way.

We then define the computational semantics. This involves giving a stochastic structure that results when the protocol is executed. Principals output bit strings (as opposed to terms) with certain probability distributions. The bit strings are then recorded in a trace as being generated, sent or received by some principal. This provides a probability distribution of traces. We show how to answer whether a bit string corresponding to a term was sent around with high probability or not. For example a formal term $\{M\}^r_A$ was sent around in the computational model if a cross-section of all traces provides the correct probability distribution that corresponds to sending $\{M\}^r_A$. Or, a nonce N was generated, if another cross-section provides the right probabilities, and that distribution must be independent of everything that happened earlier. This way we define when a certain formula in the syntax is true in the computational semantics. We then analyze whether the axioms of the syntax are true in the semantics, and if they are, then we conclude that a formula that can be proved in the syntax is also true in the semantics.

Criticism of the computational semantics of Datta et al. We point out the main aspects in which we think out method works better then that of Datta et al.:

1. They rely on counting equiprobable traces, hence their method only applies to executions when the number of possible computational traces for a given security parameter is finite. Our method on the other hand works for infinite number of traces and arbitrary probability distributions.

2. As Datta et al. derive the validity of a formula in the model from validity of the formula on individual traces, they have to make sure that there are not too many accidental coincidences. As a

result, their method only works when collision probability is negligible, but, more importantly, this results in a weaker set of syntactic axioms then what would otherwise be possible in our method. For example, they postulate that $\neg\mathrm{Send}(\hat{X},t)[b]_X\neg\mathrm{Send}(\hat{X},t)$ is an axiom whenever for all σ evaluation of variables by bit-strings, $\sigma(b) \neq \sigma(\mathrm{Send}(\hat{X},t))$. Here, $\hat{X}$ is a principle, t is a term, $[b]_X$ is an action b carried out by principal $\hat{X}$ in thread X assuming also that nothing else is carried out. In other words, it is an axiom that if $\hat{X}$ did not send t before action b, then it did not send it even after action b as long as no σ evaluates b and $\mathrm{Send}(\hat{X},t)$ the same way. However, if there is even one coincidence in their evaluations, that prevents the axiom. We think this is an unnecessary restriction. As long as the probability distributions are different (up to negligibility) for any computational interpretation of b and $\mathrm{Send}(\hat{X},t)$, we can include $\neg\mathrm{Send}(\hat{X},t)[b]_X\neg\mathrm{Send}(\hat{X},t)$ in our axioms (however, in this paper we do not consider modal formulas).

3. A further problem, that even makes the soundness proofs of Datta et al. questionable is the following: They define a formula (e.g. $\mathrm{Send}(\hat{X},t)$) to be true in the model if it holds on all traces except for some with negligible probability. They ignore the fact that the position of $\mathrm{Send}(\hat{X},t)$ on the traces may vary badly from trace to trace, for example, may depend on the future of the trace. Such counterexamples can easily be created. Maybe it is possible to prove that if there is a bad choice of the positions then there is a good choice as well, but we see no indication of such concerns in the papers of Datta et al. As we suggest to use the standard tool of *filtration* in stochastic processes, this problem is taken care of in our semantics.

4. Finally, ignoring probability distributions and correlations give rise to pathologies like this one, putting further doubts at the correctness of their soundness proofs: Suppose that the encryption scheme is such that for any n_1, n_2 bit-strings generated randomly as nonces, any public key bit-string k_2 and any random seed r_2 for the encryption, there is another public key bit-string k_1 and a random seed r_1 such that $\{n_1\}_{k_1}^{r_1} = \{n_2\}_{k_2}^{r_2}$. This does not contradict CCA-2 security. Suppose principal A generates randomly nonce n_1, and then principal B receives $\{n_2\}_{k_2}^{r_2}$ from the adversary. In such a case, it will be true according to the semantics of Datta et al., that $\exists N \exists R \exists K.\mathrm{New}(A,N) \wedge \mathrm{Receive}(B,\{N\}_K^R)$. This is however pathologic, and is a consequence of ignoring the fact that k_1, if created by the adversary, cannot correlate with n_1, which was not yet sent around. Furthermore, this seems to contradict their axiom saying that $\mathrm{FirstSend}(X,t,t') \wedge a(Y,t'') \rightarrow \mathrm{Send}(X,t') < a(Y,t'')$ where $X \neq Y$ and t subterm of t'' (meaning in our case that the first send action of A sending N had to occur before B could do anything with N.

Related Work. Formal methods emerged from the seminal work of Dolev and Yao [18], whereas computational cryptography grew out of the work of Goldwasser and Micali [20]. The first to link the two methods were Abadi and Rogaway in [3] "soundness" for passive adversaries in case of so-called type-0 security. A number of other papers for passive adversaries followed, proving "completness" [27, 5], generalizing for weaker, more realistic encryptions schemes [5], considering purely probabilistic encryptions [22, 5], including limited models for active adversaries [25], addressing the issue of forbidding key-cycles [4], considering algebraic operations and static equivalence [9, 2]. Other approaches including active adversaries are considered by Backes et al. and Canetti in their *reactive simulatability* [8, 6] and *universal composability* [12, 13] frameworks, respectively. Non trace properties were investigated in [15] and [7], however, not in the context of first order logic.

Organization of this paper. In Section 2, we outline the syntax of Basic Protocol Logic. In Section 3, we give a computational semantics to Basic Protocol Logic, and discuss soundness. Finally, in Section 4, we conclude and present directions for future work.

Special Thanks. We would like thank Stéphane Glondu, Jesus Almansa, Arnab Roy and John Mitchell for the valuable discussions on the topic as well as Matthew Franklin.

2 Basic Protocol Logic

In this section we summarize the syntax of Basic Protocol Logic. We first give the language and the axioms, then explain how to describe our aimed correctness properties in BPL.

2.1 Language

Sorts and terms. Our language is order-sorted, with sorts $\texttt{coin}$, $\texttt{name}$, $\texttt{nonce}$ and $\texttt{message}$ such that terms of sorts $\texttt{name}$ and $\texttt{nonce}$ are terms of sort $\texttt{message}$. Let $\mathcal{C}_{\texttt{name}}$ be a finite set of constants of sort $\texttt{name}$ (which represent principal names), and $\mathcal{C}_{\texttt{nonce}}$ a finite set of constants of sort $\texttt{nonce}$. For each $A \in \mathcal{C}_{\texttt{name}}$ let $\texttt{coin}_A$ be a sort such that any term of sort $\texttt{coin}_A$ is of sort $\texttt{coin}$, and let $\mathcal{C}_{\texttt{coin}_A}$ be a finite set of constants of sort $\texttt{coin}_A$. Let $\mathcal{C}_{\texttt{coin}} := \bigcup_{A\in\mathcal{C}_{\texttt{name}}} \mathcal{C}_{\texttt{coin}_A}$. We require countably infinite free variables and countably infinite bound variables for each sort. We will use $A, B, \ldots, A_1, A_2, \ldots$ ($Q, Q', \ldots, Q_1, Q_2, \ldots$, resp.) to denote constants (variables, resp.) of sort $\texttt{name}$, $N, N', \ldots, N_1, N_2, \ldots$ ($n, n', \ldots, n_1, n_2, \ldots$, resp.) denote constants (variables, resp.) of sort $\texttt{nonce}$, $r^A, \ldots, r_1^A, r_2^A, \ldots$ ($s^A, \ldots, s_1^A, s_2^A, \ldots$, or $s, s', \ldots, s_1, s_2, \ldots$, resp.) denote constants of sort $\texttt{coin}_A$ (variables of sort $\texttt{coin}_A$, or variables of sort $\texttt{coin}$, resp.). The symbols $m, m', \ldots, m_1, m_2, \ldots$ are used to denote variables of sort $\texttt{message}$ and $M, M', \ldots, M_1, M_2, \ldots$ to denote constants of sort $\texttt{message}$ (that is, either $\texttt{name}$ or $\texttt{nonce}$). Let $P, P', \ldots, P_1, P_2, \ldots$ denote any term of sort $\texttt{name}$, and let $\rho, \rho', \ldots, \rho_1, \rho_2, \ldots$ denote anything of sort $\texttt{coin}$. Compound terms of sort $\texttt{message}$ are built from constants and free variables, and are defined by the grammar:

$$t \ ::= \ M \mid m \mid \langle t, t\rangle \mid \{t\}_P^\rho.$$

Where again, $M \in \mathcal{C}_{\texttt{name}} \cup \mathcal{C}_{\texttt{nonce}}$, m is any free variable of sort $\texttt{message}$, P is any constant or free variable of sort $\texttt{name}$, and ρ is any constant or free variable of sort $\texttt{coin}$. Hence, for example, $\langle\langle A_1, \{\langle n, A_2\rangle\}_Q^{r^A}\rangle, m\rangle$ is a term. We will use the shorter $\{n, A_2\}_Q^{r^A}$ instead of $\{\langle n, A_2\rangle\}_Q^{r^A}$. We will use the meta-symbols $t, t', \ldots, t_1, t_2, \ldots$ to denote terms, and $\nu, \nu', \ldots$ to denote any term of sort $\texttt{nonce}$.

Formulas. We introduce five binary predicate symbols: $P\ \mathit{generates}\ \nu$, $P\ \mathit{receives}\ t$, $P\ \mathit{sends}\ t$, $t = t'$ and $t \sqsubseteq t'$, which represent "P generates a fresh value ν as a nonce", "P receives a message of the form t", "P sends a message of the form t", and equality and subterm relation for "t is identical with t'" and "t is a subterm of t'"), respectively. The first three are called *action predicates*, and the meta expression acts is used to denote one of the action predicates: $\mathit{generates}$, $\mathit{receives}$ and sends. Atomic formulas are either of the form $P_1\ \mathit{acts}_1\ t_1; P_2\ \mathit{acts}_2\ t_2; \cdots ; P_k\ \mathit{acts}_k\ t_k$, or $t = t'$, or $t \sqsubseteq t'$. The first one is called *trace formula*. A trace formula is used to represent a sequence of the principals' actions such as "P sends a message m, and after that, Q receives a message m'". We also use $\alpha_1; \cdots ; \alpha_k$ (or $\boldsymbol{\alpha}$, for short) to denote $P_1\ \mathit{acts}_1\ t_1; \cdots ; P_k\ \mathit{acts}_k\ t_k$ (where k indicates the *length* of $\boldsymbol{\alpha}$). When every P_i is identical with P for $1 \leq i \leq k$, then $\boldsymbol{\alpha}^P$ denotes such a trace formula. For $\boldsymbol{\alpha}$ ($\equiv \alpha_1; \cdots ; \alpha_m$) and $\boldsymbol{\beta}$ ($\equiv \beta_1; \cdots ; \beta_n$), we say $\boldsymbol{\beta}$ *includes* $\boldsymbol{\alpha}$ (denoted by $\boldsymbol{\alpha} \subseteq \boldsymbol{\beta}$), if there is a one-to-one, increasing function $j : \{1, ..., m\} \to \{1, ..., n\}$ such that $\alpha_i \equiv \beta_{j(i)}$. Formulas are defined by

$$\varphi \ ::= \ \boldsymbol{\alpha} \mid t_1 = t_2 \mid t_1 \sqsubseteq t_2 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \forall m \varphi' \mid \exists m \varphi'$$

where m is some bound variable, and φ' is obtained from φ by substituting m for every occurrence in φ of a free variable m' of the same sort as m. We use the meta expression $\varphi[\boldsymbol{m}]$ to indicate the list of all variables $\boldsymbol{m}$ occurring in φ. Substitutions are represented in terms of this notation.

Finally, we introduce the notion of (*strict*) *order-preserving merge of trace formulas* $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$: An order-preserving merge of $\boldsymbol{\alpha}$ ($\equiv \alpha_1; \cdots ; \alpha_l$) and $\boldsymbol{\beta}$ ($\equiv \beta_1; \cdots ; \beta_m$) is a trace formula

$\boldsymbol{\delta}$ ($\equiv \delta_1; \cdots; \delta_n$) if there are one-to-one increasing functions $j^\alpha : \{1, ..., l\} \to \{1, ..., n\}$, $j^\beta : \{1, ..., m\} \to \{1, ..., n\}$ such that $\alpha_i \equiv \delta_{j^\alpha(i)}$, $\beta_i \equiv \delta_{j^\beta(i)}$, and the union of the ranges of j^α and j^β cover $\{1, ..., n\}$. δ is called a *strict order-preserving merge* if, furthermore, the ranges of j^α and j^β are disjoint.

For example, both $\alpha_1; \alpha_2; \alpha_2; \alpha_3$ and $\alpha_2; \alpha_1; \alpha_3; \alpha_2$ and $\alpha_1; \alpha_2; \alpha_3$ are order preserving merges of $\alpha_1; \alpha_2$ and $\alpha_2; \alpha_3$, while the last one is not a strict order preserving merge.

Description of roles. A *protocol* is a set of *roles*, and each role for a principal (say, Q) is described as a trace formula of the form $\boldsymbol{\alpha}^Q \equiv Q\ acts_1\ t_1; \cdots; Q\ acts_k\ t_k$, where the terms $t_1, ..., t_k$ are built from variables of sort `nonce` and of sort `name`.

As an example, here we consider the Needham-Schroeder public key protocol [28], whose informal description is as follows.

1. $A \to B$: $\{n_1, A\}_B^{r_1^A}$
2. $B \to A$: $\{n_1, n_2\}_A^{r_1^B}$
3. $A \to B$: $\{n_2\}_B^{r_2^A}$

Initiator's and responder's roles of the Needham-Schroeder public key protocol (denoted by $Init_{NS}$ and $Resp_{NS}$, respectively) are described as the following formulas.

Example 1. **(Roles of the Needham-Schroeder protocol)**

$Init^A_{NS}[Q_2, n_1, m_2, s^A_1, s_2, s^A_3] \equiv$
$A\ generates\ n_1; A\ sends\ \{n_1, A\}^{s^A_1}_{Q_2}; A\ receives\ \{n_1, m_2\}^{s_2}_A; A\ sends\ \{m_2\}^{s^A_3}_{Q_2}$

$Resp^B_{NS}[Q_1, m_1, n_2, s_1, s^B_2, s_3] \equiv$
$B\ receives\ \{m_1, Q_1\}^{s_1}_B; B\ generates\ n_2; B\ sends\ \{m_1, n_2\}^{s^B_2}_{Q_1}; B\ receives\ \{n_2\}^{s_3}_B$

The brackets indicate the variables that occur in the formula.

2.2 The Axioms of Basic Protocol Logic

We extend the usual first-order predicate logic with equality by adding the following axioms (I), (II) and (III). This axiomatic system is called *Basic Protocol Logic*.

The first set of axioms postulate what properties we want to require from the equality and subterm relations. When we chose these axioms, we keep in mind that we want them to be computationally sound. The set of axioms here is not the same as it was in the original formulation of BPL, as those axioms were not all computationally sound. The original axioms required that when a finite set of literals $\{t_1 = t'_1, \ldots, t_n = t'_n, s_1 \sqsubseteq s'_1, \ldots s_j \sqsubseteq s'_j, u_1 \neq u'_1, \ldots, u_k \neq u'_k, v_1 \not\sqsubseteq v'_1, \ldots, v_l \not\sqsubseteq v'_l\}$ is unsatisfiable by elements of $\bar{A}$ (for $\bar{A}$ see below), then $\forall \boldsymbol{m} \neg(t_1 = t'_1 \wedge \cdots \wedge s_1 \sqsubseteq s'_1 \wedge \cdots \wedge u_1 \neq u'_1 \wedge \cdots \wedge v_1 \not\sqsubseteq v'_1 \wedge \cdots)$ is an axiom. So, for example, the original axioms required that $\{m\}^s_A$ and $\{m\}^{r^B}_Q$ are equal only if $s = r^B$ and $Q = A$. However, if the principal Q did not generate its public key properly, but he did it with some smart trick, and if the randomization of s is not honest, then the interpretations of these terms may turn out to be equal. The axioms we present here were sufficient for the protocols we have checked, but other protocols may need additional axioms as much more can be defined that are also computationally sound.

(I) Term axioms. Consider any set $\bar{C}$ of countably infinitely many elements of sort `name`, countably infinitely many elements of sort `nonce` and countably infinitely many elements of sort `coin` such that it includes all elements of $\mathcal{C}_{\texttt{name}}$, $\mathcal{C}_{\texttt{nonce}}$ and $\mathcal{C}_{\texttt{coin}}$. Let $\bar{\mathcal{A}}$ be the free algebra constructed from $\bar{\mathcal{C}}$ via $\langle\cdot,\cdot\rangle$ and $\{\cdot\}^{\cdot}_{\cdot}$ (with the appropriate sorts in the indexes of the encryption terms). The elements of $\bar{\mathcal{A}}$ are of sort $\texttt{message}$.

We postulate the following axioms for $=$ and $\sqsubseteq$. Let $\boldsymbol{m}$ be all variables occurring in the corresponding terms. We require these for all $A, B \in \mathcal{C}_{\texttt{name}}$:

- If $t = t'$ is true in $\bar{\mathcal{A}}$, then $\forall \boldsymbol{m} t = t'$ is axiom. If $t \sqsubseteq t'$ is true in $\bar{\mathcal{A}}$, then $\forall \boldsymbol{m} t \sqsubseteq t'$ is axiom.
- $\forall \boldsymbol{m}(t=t)$, $\forall \boldsymbol{m}(t_1=t_2 \rightarrow t_2=t_1)$, $\forall \boldsymbol{m}(t_1=t_2 \wedge t_2=t_3 \rightarrow t_1=t_3)$, $\forall \boldsymbol{m}(t_1=t_2 \rightarrow t_1\sqsubseteq t_2)$, $\forall \boldsymbol{m}(t_1\sqsubseteq t_2 \wedge t_2\sqsubseteq t_3 \rightarrow t_1\sqsubseteq t_3)$
- $\forall \boldsymbol{m} Q s s^B(\{t_1\}^{s^B}_A=\{t_2\}^s_Q \rightarrow t_1=t_2 \wedge Q=A \wedge s=s^B)$
- $\forall \boldsymbol{m}(\langle t_1,t_2\rangle=\langle t_3,t_4\rangle \rightarrow t_1=t_3 \wedge t_2=t_4)$
- $\forall \boldsymbol{m}(\{t\}^{s^B}_A \neq \langle t_1,t_2\rangle)$, $\forall \boldsymbol{m}(\{t\}^{s^B}_A \neq n)$, , $\forall \boldsymbol{m}(\{t\}^{s^B}_A \neq Q)$
- $\forall \boldsymbol{m} n(\langle t_1,t_2\rangle \neq n)$, $\forall \boldsymbol{m} Q(\langle t_1,t_2\rangle \neq Q)$
- $\forall \boldsymbol{m} s^B(t_1\sqsubseteq\{t_2\}^{s^B}_A \rightarrow t_1\sqsubseteq t_2 \vee t_1=\{t_2\}^{s^B}_A)$, $\forall \boldsymbol{m}(t \sqsubseteq \langle t_1,t_2\rangle \rightarrow t \sqsubseteq t_1 \vee t \sqsubseteq t_2 \vee t_1=\langle t_1,t_2\rangle)$
- $\forall mn(m\sqsubseteq n \rightarrow m=n)$, $\forall m s^A(m\sqsubseteq s^A \rightarrow m=s^A)$, $\forall mQ(m\sqsubseteq Q \rightarrow m=Q)$

(II) Rules for trace formulas. We introduce the following axioms (1) and (2) for trace formulas, where $\boldsymbol{\gamma_i}$'s in (2) are the list of order-preserving merges of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

(1) $\boldsymbol{\beta} \rightarrow \boldsymbol{\alpha}$ (for $\boldsymbol{\alpha} \subseteq \boldsymbol{\beta}$)
(2) $\boldsymbol{\gamma_1} \vee \cdots \vee \boldsymbol{\gamma_n} \leftrightarrow \boldsymbol{\alpha} \wedge \boldsymbol{\beta}$

These axioms express the intuition that if a trace "happens", then a subtrace of it also happens, and two traces happen if and only if one of their possible merges happen.

(III) Axioms for relationship between properties. We introduce the following set of formulas as non-logical axioms. These axioms represent some properties about nonces and cryptographic assumptions.

(1) Ordering:

$$\forall Q_1 Q_2 nm(Q_1\ \textit{generates}\ n\ \wedge\ Q_2\ \textit{sends/receives}\ m\ \wedge\ n \sqsubseteq m$$
$$\rightarrow \neg(Q_2\ \textit{sends/receives}\ m; Q_1\ \textit{generates}\ n))$$

Let $|\overline{m_1 \sqsubseteq m_2 \sqsubseteq m_3}|$ mean that $m_1 \sqsubseteq m_2 \sqsubseteq m_3$ and the only way m_1 occurs in m_3 is within m_2. That is:

$$|\overline{m_1 \sqsubseteq m_2 \sqsubseteq m_3}| := \forall m(m_1 \sqsubseteq m \sqsubseteq m_3 \rightarrow$$
$$m_2 \sqsubseteq m\ \vee\ (m \sqsubseteq m_2 \wedge \forall m_4(m_2 \sqsubseteq m_4 \rightarrow \langle m, m_4\rangle \not\sqsubseteq m_3 \wedge \langle m_4, m\rangle \not\sqsubseteq m_3))).$$

(2) Nonce verification 1: For each A, B constants of sort $\texttt{name}$ and r^A constant of sort $\texttt{coin}_A$, we postulate

$$\forall Q n_1 m_2 m_5 m_6(A\ \textit{generates}\ n_1; A\ \textit{sends}\ m_2; Q\ \textit{receives}\ m_5$$
$$\wedge\ |\overline{n_1 \sqsubseteq \{m_6\}^{r^A}_B \sqsubseteq m_2}|\ \wedge\ n_1 \sqsubseteq m_5\ \wedge\ \neg|\overline{n_1 \sqsubseteq \{m_6\}^{r^A}_B \sqsubseteq m_5}|$$
$$\wedge\ \forall m_7(A\ \textit{sends}\ m_7 \wedge n_1 \sqsubseteq m_7 \rightarrow |\overline{n_1 \sqsubseteq \{m_6\}^{r^A}_B \sqsubseteq m_7}|)$$
$$\rightarrow \exists m_3 m_4(A\ \textit{sends}\ m_2; B\ \textit{receives}\ m_3; B\ \textit{sends}\ m_4; Q\ \textit{receives}\ m_5$$
$$\wedge\ \{m_6\}^{r^A}_B \sqsubseteq m_3 \wedge n_1 \sqsubseteq m_4))$$

(3) Nonce verification 2: For each A, B C of sort $\texttt{name}$ (where A and C may coincide), r^A constant of sort $\texttt{coin}_A$ and r^B constant of sort $\texttt{coin}_B$, we postulate

$$\forall n_1 m_2 m_5 m_6 m_8 m_{10}(A\ \textit{generates}\ n_1; A\ \textit{sends}\ m_2; C\ \textit{receives}\ m_5$$

$$
\begin{aligned}
&\wedge \overline{|n_1 \sqsubseteq \{m_6\}_B^{r^A} \sqsubseteq m_2|} \ \wedge\ n_1 \sqsubseteq m_5 \ \wedge\ \neg\overline{|n_1 \sqsubseteq \{m_6\}_B^{r^A} \sqsubseteq m_5|} \\
&\wedge \forall m_7(A\ sends\ m_7 \wedge n_1 \sqsubseteq m_7 \rightarrow \overline{|n_1 \sqsubseteq \{m_6\}_B^{r^A} \sqsubseteq m_7|}) \\
&\wedge B\ sends\ m_4 \ \wedge\ \overline{|n_1 \sqsubseteq \{m_8\}_C^{r^B} \sqsubseteq m_4|} \\
&\wedge \forall m_9(B\ sends\ m_9 \wedge n_1 \sqsubseteq m_9 \rightarrow \overline{|n_1 \sqsubseteq \{m_8\}_C^{r^B} \sqsubseteq m_9|}) \\
&\wedge(\neg(C\ sends\ m_{10} \wedge n_1 \sqsubseteq m_{10}) \vee A = C) \\
\rightarrow\ &\overline{|n_1 \sqsubseteq \{m_8\}_C^{r^B} \sqsubseteq m_5|}
\end{aligned}
$$

There are other possible axiomatizations, but the authors of [23] found this particularly useful (more exactly a somewhat less general version). The meaning of the Ordering axiom is clear. Nonce verification 1 and 2 are based on the idea of the authentication-tests [21]. Nonce verification I means that if A sent out a nonce n_1 encrypted with the public key of B that was not sent in any other way, and Q received this nonce in some other form, then the encrypted nonce had to go through B. The reason that we require A and B to be names and not arbitrary variables is that we do not want to require any principals in an arbitrary run to encrypt securely. A message may look like an encryption, but if the key was not generated properly or if the randomization of the encryption was not done well, then the information might leak.

2.3 Query form and correctness properties

Our aimed correctness properties are described in a special form of formulas, called *query form*. Let $\boldsymbol{\alpha}^A[\boldsymbol{Q}, \boldsymbol{m}, \boldsymbol{s}]$ be a role $A\,acts_1\,t_1; A\,acts_2\,t_2; \cdots ; A\,acts_k\,t_k$ where each $acts_i$ ($1 \leq i \leq k$) is one of $sends$, $receives$ and $generates$, t_i is a term built from messages in $\boldsymbol{m} = \{m_1, ..., m_h\}$ (some of which may be nonces) from coins in $\boldsymbol{s} = \{s_1, ..., s_i\}$ and from names A and $\boldsymbol{Q} = \{Q_1, ..., Q_l\}$. Let $\boldsymbol{\alpha}^A_{\leq i}[\boldsymbol{Q}, \boldsymbol{m}, \boldsymbol{s}]$ denote an initial segment of $\boldsymbol{\alpha}^A[\boldsymbol{Q}, \boldsymbol{m}, \boldsymbol{s}]$ ending with $A\,acts_i\,t_i$ (for $1 \leq i \leq k$), i.e., $\boldsymbol{\alpha}^A_{\leq i}[\boldsymbol{Q}, \boldsymbol{m}, \boldsymbol{s}] \equiv A\,acts_1\,t_1; \cdots A\,acts_i\,t_i$. Let $\boldsymbol{\alpha}^A_{\leq 0}[\boldsymbol{Q}, \boldsymbol{m}, \boldsymbol{s}] \equiv A = A$.

The query form includes a formalization of *principal's honesty* $Honest(\boldsymbol{\alpha}^A)$, which is defined as follows, the intuitive meaning being that A follows the role $\boldsymbol{\alpha}^A$ and does nothing else, but it may not complete it:

Definition 1. (Principal's honesty)
$Honest(\boldsymbol{\alpha}^A)[\boldsymbol{Q}, \boldsymbol{m}, \boldsymbol{s}]$
$\stackrel{def}{\equiv}\ \exists \boldsymbol{Qms} \bigvee_{i \in \{0\} \cup \{j \mid\ acts_j = sends\ \} \cup \{k\}}\ \boldsymbol{\alpha}^A_{\leq i}[\boldsymbol{Q}, \boldsymbol{m}, \boldsymbol{s}] \wedge Only(\boldsymbol{\alpha}^A_{\leq i}[\boldsymbol{Q}, \boldsymbol{m}, \boldsymbol{s}])$

For any role $\boldsymbol{\alpha}^A$ (or $\boldsymbol{\alpha}^A \equiv A$), $Only(\boldsymbol{\alpha}^A)$ denotes the following formula, whose intuitive meaning is "A performs only $\boldsymbol{\alpha}^A$ (or nothing)".

$$
\begin{aligned}
Only(\boldsymbol{\alpha}^A) \equiv\ & \forall n(A\ generates\ n_1 \rightarrow n \in Generates(\boldsymbol{\alpha}^A)) \\
& \wedge \forall m_1(A\ sends\ m_1 \rightarrow m_1 \in Sends(\boldsymbol{\alpha}^A)) \\
& \wedge \forall m_2(A\ receives\ m_2 \rightarrow m_2 \in Receives(\boldsymbol{\alpha}^A))
\end{aligned}
$$

Here, $Sends(\boldsymbol{\alpha}^A)$ denotes the set $\{t_j \mid A\ sends\ t_j \subseteq \boldsymbol{\alpha}^A\}$, and ($Receives(\boldsymbol{\alpha}^A)$, $Generates(\boldsymbol{\alpha}^A)$ are defined similarly. Set theoretical notation as $m \in Sends(\boldsymbol{\alpha}^A)$ (as well as $m \in Receives(\boldsymbol{\alpha}^A)$ and $m \in Generates(\boldsymbol{\alpha}^A)$) is an abbreviation of a disjunctive form: for example, if $Sends(\boldsymbol{\alpha}^A) = \{t'_1, \ldots, t'_j\}$, then $m \in Sends(\boldsymbol{\alpha}^A)$ denotes the formula $(m = t'_1) \vee (m = t'_2) \vee \cdots \vee (m = t'_j)$. (As a special case, if $Sends(\boldsymbol{\alpha}^A)$ is empty then $m \in Sends(\boldsymbol{\alpha}^A)$ denotes $A \neq A$, that is, impossible.)

Intuitively, each disjunct $\boldsymbol{\alpha}^A_{\leq i} \wedge Only(\boldsymbol{\alpha}^A_{\leq i})$ in $Honest(\boldsymbol{\alpha}^A)$ represents a historical record of P's actions at each step of his run: the sequence of actions $\boldsymbol{\alpha}^A_{\leq i}$ represents A's performance until this step, and $Only(\boldsymbol{\alpha}^A_{\leq i})$ represents that A performs only $\boldsymbol{\alpha}^A_{\leq i}$. $Only(\boldsymbol{\alpha}^A_{\leq 0})$ means that nothing was performed. Thus, $Honest(\boldsymbol{\alpha}^A[\boldsymbol{Q},\boldsymbol{m},\boldsymbol{s}])$ represents "A performs only a run of an initial segment of $\boldsymbol{\alpha}^A$ which ends with a sending action or the last action of $\boldsymbol{\alpha}^A$, and uses the data items $\boldsymbol{Q}$, $\boldsymbol{m}$ and $\boldsymbol{s}$ for each run".

As an example, we present the honesty of initiator A of the Needham-Schroeder protocol below.

Example 2. **(Initiator's honesty of the NS protocol)**

$$Honest(Init^A_{NS})[Q_1,n_1,m_2,s_1^A,s_2,s_3^A] \equiv$$

$$\exists Q_1 n_1 m_2 s_1^A s_2 s_3^A \left(\begin{pmatrix} \forall n_3 \neg(A\ \textit{generates}\ n_3) \\ \wedge \forall m_4 \neg(A\ \textit{sends}\ m_4) \\ \wedge \forall m_5 \neg(A\ \textit{receives}\ m_5) \end{pmatrix} \right.$$
$$\vee \begin{pmatrix} A\ \textit{generates}\ n_1; A\ \textit{sends}\ \{n_1,A\}^{s_1^A}_{Q_1} \\ \quad \wedge \forall n_3(A\ \textit{generates}\ n_3 \to n_3 = n_1) \\ \quad \wedge \forall m_4(A\ \textit{sends}\ m_4 \to m_4 = \{n_1,A\}^{s_1^A}_{Q_1}) \\ \quad \wedge \forall m_5 \neg(A\ \textit{receives}\ m_5) \end{pmatrix}$$
$$\left. \vee \begin{pmatrix} A\ \textit{generates}\ n_1; A\ \textit{sends}\ \{n_1,A\}^{s_1^A}_{Q_1}; A\ \textit{receives}\ \{n_1,m_2\}^{s_2}_{A}; A\ \textit{sends}\ \{m_2\}^{s_3^A}_{Q_1} \\ \quad \wedge \forall n_3(A\ \textit{generates}\ n_3 \to n_3 = n_1) \\ \quad \wedge \forall m_4(A\ \textit{sends}\ m_4 \to m_4 = \{n_1,A\}^{s_1^A}_{Q_1} \vee m_4 = \{m_2\}^{s_3^A}_{Q_1}) \\ \quad \wedge \forall m_5(A\ \textit{receives}\ m_5 \to m_5 = \{n_1,m_2\}^{s_2}_{A}) \end{pmatrix} \right)$$

Note that our formalization of honesty is stronger than the usual sense. That is, our definition of honesty is restricted to a single set of data items used for the honest principal's runs, whereas the usual sense of honesty means that P may perform multiple runs whose data items may differ from each other. However, by regarding a strict order-preserving merge of a certain number of the same role as a single role, we can represent the honesty with respect to any fixed finite number of the sets of data items which are used for all possible runs by the honest principal.

First-order formalization of correctness properties. We introduce a general form of formulas, called *query form*, to represents our aimed correctness properties. In order to make the discussion simpler, we consider only the case of two party authentication protocols, however our query form can be easily extended so as to represent the correctness properties with respect to other types of protocols which include more than two principals.

Definition 2. (Query form) *Query form is a formula of the following form.*

$$Honest(\boldsymbol{\alpha}^A) \wedge \boldsymbol{\beta}^B \wedge Only(\boldsymbol{\beta}^B) \to \boldsymbol{\gamma}$$

Our aimed correctness properties are described as a special case of the query form. For example, the non-injective agreement of the protocol $\Pi = \{\boldsymbol{\alpha}^A[B/Q_2,\boldsymbol{m},\boldsymbol{s}],\ \boldsymbol{\beta}^B[A/Q_1,\boldsymbol{m},\boldsymbol{s}]\}$ from responder's (B's) view can be described as the following formula.

$$\begin{aligned} &Honest(\boldsymbol{\alpha}^A[Q_2,\boldsymbol{m},\boldsymbol{s}]) \wedge \boldsymbol{\beta}^B[A/Q_1,\boldsymbol{N}/\boldsymbol{m},\boldsymbol{r}/\boldsymbol{s}] \\ &\qquad \wedge Only(\boldsymbol{\beta}^B[A/Q_1,\boldsymbol{N}/\boldsymbol{m},\boldsymbol{r}/\boldsymbol{s}]) \to \boldsymbol{\alpha}^A[B/Q_2,\boldsymbol{N}/\boldsymbol{m},\boldsymbol{r}/\boldsymbol{s}] \end{aligned}$$

The matching conversations and the injective agreement can be obtained by replacing the right hand side of the implication with the strict order-preserving merge of $\boldsymbol{\alpha}^A[B/Q_2,\boldsymbol{N}/\boldsymbol{m},\boldsymbol{r}/\boldsymbol{s}]$ and $\boldsymbol{\beta}^B[A/Q_1,\boldsymbol{N}/\boldsymbol{m},\boldsymbol{r}/\boldsymbol{s}]$, and with $\boldsymbol{\alpha}^A[B/Q_2,\boldsymbol{N}/\boldsymbol{m},\boldsymbol{r}/\boldsymbol{s}] \wedge Only(\boldsymbol{\alpha}^A[B/Q_2,\boldsymbol{N}/\boldsymbol{m},\boldsymbol{r}/\boldsymbol{s}])$, respectively.

Actually, our formalization of the agreement properties is weaker than the usual sense, because our honesty assumption is stronger than the usual sense. However, as we have explained in the definition of honesty (Definition 2), our query form can be extended so that the honest principal may use a finite number of sets of data items used for his/her runs.

Remarks. In this paper we choose the set of formulas (III) of Section 2.2 as non-logical axioms, which have enough power to prove our aimed agreement properties in the sense of [26] for various protocols with public keys, such as the Needham-Schroeder-Lowe protocol (cf. Protocol 4.19 in [11]), ISO/IEC 11770-3 Key Transport Mechanism 6 (cf. Protocol 4.16 in [11]), and so on. With additional syntax and non-logical axioms about shared keys or signature we can also prove correctness of the core part of Kerberos (cf. Protocol 3.25 in [11]) and the ISO 9798-3 protocol (used as an example in [16]). BPL however, does not provide flexible compositional treatment of proofs as Protocol Composition Logic does.

3 Computational Semantics

3.1 Computational Asymmetric Encryption Schemes

The fundamental objects of the computational world are strings, $\mathsf{strings} = \{0,1\}^*$, and families of probability distributions over strings. These families are indexed by a *security parameter* $\eta \in \mathbb{N}$ (which can be roughly understood as key-lengths).

Definition 3 (Negligible Function). *A function $f : \mathbb{N} \to \mathbb{R}$ is said to be* negligible*, if for any $c > 0$, there is an $n_c \in \mathbb{N}$ such that $|f(\eta)| \leq \eta^{-c}$ whenever $\eta \geq n_c$.*

Pairing is an injective *pairing function* $[\cdot,\cdot] : \mathsf{strings} \times \mathsf{strings} \to \mathsf{strings}$. We assume that changing a bit string in any of the argument to another bit string of the same length does not influence the length of the output of the pairing. An encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ with key generation $\mathcal{K}$, encryption $\mathcal{E}$ and decryption $\mathcal{D}$. Let $\mathsf{plaintexts}$, $\mathsf{ciphertexts}$, $\mathsf{publickey}$ and $\mathsf{secretkey}$ be nonempty subsets of $\mathsf{strings}$. The set coins is some probability field that stands for coin-tossing, *i.e.*, randomness.

Definition 4 (Encryption Scheme). *A* computational asymmetric encryption scheme *is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where:*

- $\mathcal{K} : \mathsf{param} \times \mathsf{coins} \to \mathsf{publickey} \times \mathsf{secretkey}$ *is a key-generation algorithm with* $\mathsf{param} = \mathbb{N}$*,*
- $\mathcal{E} : \mathsf{publickey} \times \mathsf{plaintexts} \times \mathsf{coins} \to \mathsf{ciphertexts}$ *is an encryption function, and*
- $\mathcal{D} : \mathsf{secretkey} \times \mathsf{strings} \to \mathsf{plaintexts}$ *is such that for all (e, d) output of $\mathcal{K}(\eta, \cdot)$ and $c \in$* coins

$$\mathcal{D}(d, \mathcal{E}(e, m, c)) = m \textit{ for all } m \in \mathsf{plaintexts}.$$

All these algorithms are computable in polynomial time with respect to the security parameter.

In this paper, we assume that the encryption scheme satisfies adaptive chosen ciphertext security (CCA-2) defined the following way:

Definition 5 (Adaptive Chosen Ciphertext Security). *A computational public-key encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ provides* indistinguishability under the adaptive chosen-ciphertext attack *if for all PPT adversaries A and the function*

$$\begin{array}{ll} |\Pr[\,(e,d) \longleftarrow \mathcal{K}(1^\eta); b \longleftarrow \{0,1\}\,; & \\ \quad m_0, m_1 \longleftarrow \mathsf{A}^{\mathcal{D}_1(\cdot)}(1^\eta, e); & \\ \quad c \longleftarrow \mathcal{E}(e, m_b); & \\ \quad g \longleftarrow \mathsf{A}^{\mathcal{D}_2(\cdot)}(1^\eta, e, c) : & \\ \quad b = g &] - \frac{1}{2}| \end{array}$$

is negligible in η. The oracle $\mathcal{D}_1(x)$ returns $\mathcal{D}(d,x)$, and $\mathcal{D}_2(x)$ returns $\mathcal{D}(d,x)$ if $x \neq c$ and returns $\perp$ otherwise. The adversary is assumed to keep state between the two invocations. It is required that m_0 and m_1 be of the same length. The probability includes all instances of randomness: key generation, the choices of the adversary, the choice of b, the encryption.

In other words, first a public key-private key pair is generated as well as a random bit b. Then, the adversary is given the public key, and a decryption oracle, which it can invoke as many times as wished, and at the end it comes up with a pair of bit strings m_0, m_1 of the same length, which it hands to an encryption oracle. Out of these two messages, the oracle encrypts the one determined by the initial choice of random bit b, and hands the ciphertext back to the adversary. The adversary can further invoke the decryption oracle (which decrypts everything except for the ciphertext computed by the encryption oracle. At the end, the adversary has to make a good guess for b. This guess is g, and the adversary wins if the probability of making a good guess significantly differs from $1/2$.

That is, an adversary should not be able to learn from a ciphertext whether it contains the plaintext m_0 or the plaintext m_1, even if:

- the adversary knows the public key used to encrypt,
- the adversary can choose the messages m_0 and m_1 itself, so long as the messages have the same length, and
- the adversary can request and receive the decryption of any *other* ciphertext.

It was shown, in [10], that the above definition is equivalent with another that seems stricter at first, namely, when an n-tuple of encryption and decryption oracles are given, each with separate encryption and decryption keys, but using the same bit b to choose from the submitted plaintexts. The adversary is allowed to invoke the oracles in any order but it cannot submit a message that was received from an encryption oracle to the corresponding decryption oracle.

3.2 Stochastic Model for the Computational Execution of BPL

In the following, we discuss the mathematical objects that we use to represent a computational execution of a protocol. Our plan is to define a computational semantics, show that the syntactic axioms hold if the encryption scheme is CCA-2 secure, and, as a result, if the query-form (or anything else) is provable in the syntax, it must be true in any computational model.

The main improvements from computational semantics proposed by Datta et al. were explained in the introduction. Our approach avoids the necessity to tell on each trace whether something is satisfied or not because we want to avoid pure coincidences. Since computational behavior is probabilistic, it is often not natural to ask whether something is satisfied on a particular trace or not. The natural question is whether the traces produce the right kind of probability distribution for a certain variable.

First, since probabilities and complexity are involved, we need a probability space for each value of the security parameter. Since time plays an important role in the execution, what we need is the probability space for a *stochastic process*. For the presentation here, we limit ourselves to finite probability spaces as explaining the notion of measurability and stochastic processes is much simpler this way, but for anyone familiar with these notions in infinite spaces it is near to trivial to generalize the method to allowing infinite steps (but polynomial expected run-time). So, here we assume that for each security parameter, there is a maximum number of execution steps n^η. The following notions that we introduce are standard in probability theory.

We will denote the finite probability space for an execution of a protocol with security parameter η by Ω^η, subsets of which are called events. Let $\mathcal{F}^\eta$ denote the set of all subsets of Ω^η (including the empty set). A subset containing only one element is called an *elementary event*. The set Ω^η is meant to include all randomness of an execution of the protocol (and perhaps some additional information). A *probability measure* p^η assigns a probability to each subset such that it is additive with respect to disjoint unions of sets (so it is enough to assign a probability to each element of Ω^η, then the probability of any subset can be computed). When it is clear which probability space we are talking about, we will just use the notation $\Pr$.

In order to describe what randomness was carried out until step $i \in \{0, 1, ..., n^\eta\}$, we assign a subset $\mathcal{F}_i^\eta \subseteq \mathcal{F}^\eta$ to each i, such that $\mathcal{F}_i^\eta$ is closed under union and intersection, and includes $\emptyset$ and Ω^η, and $\mathcal{F}_i^\eta \subseteq \mathcal{F}_{i+1}^\eta$. The set $\{\mathcal{F}_i^\eta\}_{i=1}^{n^\eta}$ is called *filtration*. Since everything is finite, $\mathcal{F}_i^\eta$ is *atomistic*, that is, each element of it can be obtained as a union of disjoint, minimal (with respect to inclusion) nonempty elements. The minimal nonempty elements are called *atoms*. We introduce the notation

$$\mathbf{Pr} = \{(\Omega^\eta, \{\mathcal{F}_i^\eta\}_{i=0}^{n^\eta}, p^\eta)\}_{\eta \in \mathsf{param}}.$$

We included $\mathcal{F}_0^\eta$ to allow some initial randomness such as key generation. A discrete *random variable* on Ω^η is a function on Ω^η taking some discrete value. Since $\mathcal{F}_i^\eta$ contains the events determined until step i, a random variable g^η depends only on the randomness until i exactly if g is constant on the atoms of $\mathcal{F}_i^\eta$; this is the same as saying that for any possible value c, the set $[g^\eta = c] := \{\omega \mid g^\eta(\omega) = c\}$ is an element of $\mathcal{F}_i^\eta$. In this case, we say that g^η is *measurable* with respect to $\mathcal{F}_i^\eta$. We will, however need a somewhat more complex dependence-notion. We will need to consider random variables that are determined by the randomness until step i_1 on certain random paths, but until step i_2 on other paths, and possibly something else on further paths. For this, we have to first consider a function $J^\eta : \Omega^\eta \to \{0, 1, ..., n^\eta\}$ that tells us which time step to consider on each ω. This function should only depend on the past, so for each $i \in \{0, 1, ..., n^\eta\}$, we require that the set $[J^\eta = i] \in \mathcal{F}_i^\eta$. We will call this function a *stopping time*. The events that have occurred until the stopping time J^η are contained in

$$\mathcal{F}_J^\eta := \{S \mid S \subseteq \Omega^\eta, \text{ and for all } i = 0, 1, ..., n^\eta,\ S \cap [J^\eta = i] \in \mathcal{F}_i^\eta\}.$$

Then, a random variable f^η depends only on the events until the stopping time J^η iff for each c in its range, $[f^\eta = c] \in \mathcal{F}_J^\eta$. Furthermore, a random variable h^η on Ω^η is said to be independent of what happened until J^η iff for any $S \in \mathcal{F}_J^\eta$ and a c possible value of h^η, $\Pr([h^\eta = c] \cap S) = \Pr([h^\eta = c]) \Pr(S)$. Finally, it is easy to see that for each random variable f^η, there is a stopping time J_f^η such that f^η is measurable with respect to $\mathcal{F}_{J_f}^\eta$, and J_f^η is minimal in the sense that f^η is not measurable with respect to any other $\mathcal{F}_J^\eta$ if there is an ω such that $J^\eta(\omega) < J_f^\eta(\omega)$.

Example 3. Suppose coins are tossed three times, one after the other. Then

$$\Omega = \{(a, b, c) \mid a, b, c = 0, 1\}.$$

Let $(1, \cdot, \cdot) := \{(1, b, c) \mid b, c = 0, 1\}$. $(0, \cdot, \cdot)$, etc. are defined analogously. At step $i = 1$, the outcome of the first coin-tossing becomes known. So,

$$\mathcal{F}_1 = \{\emptyset, (0, \cdot, \cdot), (1, \cdot, \cdot), \Omega\}.$$

At step $i = 2$, the outcome of the second coin becomes known too, therefore $\mathcal{F}_2$, besides $\emptyset$ and Ω, contains $(0, 0, \cdot)$, $(0, 1, \cdot)$, $(1, 0, \cdot)$ and $(1, 1, \cdot)$ as atoms, and all possible unions of these. $\mathcal{F}_3$ is all subsets. A function g that is measurable with respect to $\mathcal{F}_1$, is constant on $(0, \cdot, \cdot)$ and on $(1, \cdot, \cdot)$, that is, g only depends on the outcome of the first coin tossing, but not the rest. Similarly, an f

measurable on $\mathcal{F}_2$, is constant on $(0,0,\cdot)$, on $(0,1,\cdot)$, on $(1,0,\cdot)$ and on $(1,1,\cdot)$. A stopping time is for example the J that equals the position of the first 1, or 3 if there is never 1:

$$J(\ (a_1,a_2,a_3)\ ) = \begin{cases} i \text{ if } a_i = 1 \text{ and } a_k = 0 \text{ for } k < i \\ 3 \text{ if } a_k = 0 \text{ for all } k = 1,2,3 \end{cases}$$

The atoms of $\mathcal{F}_J$ are $(1,\cdot,\cdot)$, $(0,1,\cdot)$, $\{(0,0,1)\}$ and $\{(0,0,0)\}$.

For each value of the security parameter, an execution of the protocol involves some principals. Each principal has a distinct name, a bit-string not longer than the upper bound n^η. Each principal generates an encryption-key, decryption-key pair at the initialization. Hence, if $\mathbf{Pr} = \{(\Omega^\eta, \{\mathcal{F}_i^\eta\}_{i=0}^{n^\eta}, p^\eta)\}_{\eta\in\mathsf{param}}$ is the stochastic space of the execution of the protocol, let $\mathcal{P}^\eta$ be a set of (polynomially bounded number of) elements of the form $(A^\eta, (e_A^\eta, d_A^\eta))$ where $A^\eta \in \{0,1\}^{n^\eta}$, and (e_A^η, d_A^η) is a pair of probability distributions on Ω^η measurable with respect to $\mathcal{F}_0^\eta$ such that $\Pr[\omega : (e_A^\eta(\omega), d_A^\eta(\omega)) \notin Range(\mathcal{K}(\eta,\cdot))]$ is a negligible function of η. We assume that if $A = B$, then $(e_A^\eta, d_A^\eta) = (e_B^\eta, d_B^\eta)$. The set $\{\mathcal{P}^\eta\}_{\eta\in\mathsf{param}}$ describes all the principals, corrupted and uncorrupted, that take part in the execution at a given security parameter, along with their public and secret keys. Let $\mathcal{P} = \{\mathcal{P}^\eta\}_{\eta\in\mathsf{param}}$.

For nonces, we choose the following definition. Since CCA-2 security is length-revealing, we have to assume that nonces are always of some fixed length m^η for each security parameter η. Let $\mathcal{N}$ be a set of elements of the form $\{N^\eta\}_{\eta\in\mathsf{param}}$ where $N^\eta : \Omega^\eta \to \{0,1\}^{m^\eta}$ such that N^η is uniformly distributed over $\{0,1\}^{m^\eta}$, and for any two $\{N_1^\eta\}_{\eta\in\mathsf{param}}, \{N_2^\eta\}_{\eta\in\mathsf{param}} \in \mathcal{N}$, N_1^η and N_2^η are independent (i.e. for any two $s_1, s_2 \in \{0,1\}^{m^\eta}$, $\Pr[N_1^\eta = s_1 \wedge N_2^\eta = s_2] = \Pr[N_1^\eta = s_1]\Pr[N_2^\eta = s_2]$). This set describes the nonces that were generated with overwhelming probability during the execution of the protocol. The nonces have to be independent of each other, and have uniform distribution over the given length. The nonces also have to be independent of what happened earlier when they are being generated, but we will require this later.

Let $\mathcal{R}$ be a set of elements of the form $R = \{R^\eta\}_{\eta\in\mathsf{param}}$ where $R^\eta : \Omega^\eta \to \mathsf{coins}$. Let $\mathcal{R}_g$ be the subset of $\mathcal{R}$ which are properly randomized, that is, for which the values in coins have the distribution required for the encryption scheme. That is, they have good distribution (hence the g).

Messages: Let the set of messages be $\mathcal{M}$ elements of the form $M = \{M^\eta\}_{\eta\in\mathsf{param}}$, where $M^\eta : \Omega^\eta \to \{0,1\}^{n^\eta}$. For any two messages, M_1, M_2, we will denote that $M_1 \approx M_2$ iff $p^\eta[\omega : M_1(\omega) \neq M_2(\omega)]$ is a negligible function of η. This way, $\approx$ is an equivalence notion on the set of messages. Let $D_M := \mathcal{M}/\!\approx$, let $D_N := \mathcal{N}/\!\approx\ \subset D_M$, and let

$$D_P := \{A \in \mathcal{M} : (A^\eta, (e_A^\eta, d_A^\eta)) \in \mathcal{P} \text{ for some } (e_A^\eta, d_A^\eta)\}/\!\approx\ \subset D_M$$

We have to define what we mean by a computational pairing and encryption. For any $X, X_1, X_2 \in D_M$, we write that $X =_{\mathrm{C}} \langle X_1, X_2\rangle$, if for some (hence for all) $M_1 = \{M_1^\eta\}_{\eta\in\mathsf{param}} \in X_1$ and $M_2 = \{M_2^\eta\}_{\eta\in\mathsf{param}} \in X_2$, the ensemble of random variables $\{\omega \mapsto [M_1^\eta(\omega), M_2^\eta(\omega)]\}_{\eta\in\mathsf{param}}$ is an element of X. Further, if $A \in \mathcal{P}$, and $R \in \mathcal{R}$, then we will write that $X =_{\mathrm{C}} \{X_1\}_A^R$ if for any (hence for all) $M_1 = \{M_1^\eta\}_{\eta\in\mathsf{param}} \in X_1$, the ensemble of random variables $\{\omega \mapsto \mathcal{E}(e_A^\eta(\omega), M_1^\eta(\omega), R(\omega))\}_{\eta\in\mathsf{param}}$ is an element of X. This way, we can consider an element of the free term algebra $T(D_M)$ over D_M as an element of D_M. Let $\sqsubseteq_{T(D_M)}$ denote the subterm relation on $T(D_M)$. This generates a subterm relation $\sqsubseteq_{\mathrm{C}}$ on D_M by defining $X_1 \sqsubseteq_{\mathrm{C}} X_2$ to be true iff there is an element $X \in T(D_M)$ such that $X_1 \sqsubseteq_{T(D_M)} X$ and $X_2 =_{\mathrm{C}} X$.

Execution trace: The execution trace is defined as $Tr = \{Tr^\eta\}_{\eta\in\mathsf{param}}$ of the form

$$\omega \mapsto Tr^\eta(\omega) = P_1^\eta(\omega)\ acts_1^\eta(\omega)\ s_1^\eta(\omega); ...; P_{n^\eta(\omega)}^\eta(\omega)\ acts_{n^\eta(\omega)}^\eta(\omega)\ s_{n^\eta(\omega)}^\eta(\omega)$$

where for each η security parameter, $\omega \in \Omega^\eta$, $n^\eta(\omega)$ is a natural number less than n^η, $P_i^\eta(\omega) \in D_P$, $acts_i^\eta(\omega)$ is one of $generates$, $sends$, $receives$ and $s_i^\eta(\omega) \in \{0,1\}^*$. For each η, ω, and $i \in \{1, ..., n^\eta(\omega)\}$, let

$$Tr_i^\eta(\omega) = P_i^\eta(\omega)\; acts_i^\eta(\omega)\; s_i^\eta(\omega)$$

We also require that Tr_i^η be measurable with respect to $\mathcal{F}_i^\eta$ for all i. Moreover, we require that any of *Tr* is PPT computable from the earlier ones.

3.3 Computational Semantics

We now explain how to give computational semantics to the syntax, and what it means that a formula of the syntax is true in the semantics. For a given security parameter, an *execution* is played by a number of participants.

Assumptions. In a particular execution, we assume that the principals corresponding to names in the syntax (that is, they correspond to elements in $\mathcal{C}_{\mathtt{name}}$) are *regular* (non-corrupted). We assume that these participants generate their keys and encrypt correctly (that is, the keys are properly distributed, and also r^A is properly randomized) with a CCA-2 encryption scheme, and never use their private keys in any computation except for decryption. For other participants (possibly corrupted), we do not assume this. (Encrypting correctly is essential to able to prove the nonce verification axioms.) We further assume that pairing of any two messages differs from any nonce and from any principal name on sets of non-negligible probability. The network is completely controlled by the adversary. The sent and received bit strings are recorded in a trace in the order they happen. Freshly generated bit-strings produced by the regular participants are also recorded. The combined algorithms of the participants and the adversary are assumed to be probabilistic polynomial time.

Such a situation, with the definitions of the previous section, produces a *computational trace structure associated to the execution* of the form

$$\mathfrak{M} = (\Pi, [\cdot,\cdot], \mathbf{Pr}, \mathcal{P}, \mathcal{N}, \mathcal{R}_g, Tr, \Phi_{\mathcal{C}}),$$

where $\Phi_{\mathcal{C}}$ is a one-to-one function on $\mathcal{C}_{\mathtt{name}} \cup \mathcal{C}_{\mathtt{nonce}} \cup \mathcal{C}_{\mathtt{coin}}$ such that

- $\Phi_{\mathcal{C}}(A) \in D_P$ for any $A \in \mathcal{C}_{\mathtt{name}}$ such that $(e^\eta_{\Phi_{\mathcal{C}}(A)}, d^\eta_{\Phi_{\mathcal{C}}(A)})$ is measurable with respect to $\mathcal{F}_0$ and has the correct key distribution, and for different constants are independent of each other
- $\Phi_{\mathcal{C}}(N) \in D_N$ for any $N \in \mathcal{C}_{\mathtt{nonce}}$,
- $\Phi_{\mathcal{C}}(r) \in \mathcal{R}_g$ for any $r \in \mathcal{C}_{\mathtt{coin}}$.

An *extension* of $\Phi_{\mathcal{C}}$ to evaluation of free variables is a function Φ that is the same on constants as $\Phi_{\mathcal{C}}$, and for variables Q, n, m, s^A, s of sort $\mathtt{name}$, $\mathtt{nonce}$, $\mathtt{message}$, $\mathtt{coin}_A$ and $\mathtt{coin}$ respectively, $\Phi(Q) \in D_P$, $\Phi(n) \in D_N$, $\Phi(m) \in D_M$, $\Phi(s^A) \in \mathcal{R}_g$ and $\Phi(s) \in \mathcal{R}$ hold. Then, for any t term, $\Phi(t) \in D_M$ is defined on terms as

- $\Phi(\langle t_1, t_2\rangle) = \langle \Phi(t_1), \Phi(t_2)\rangle$;
- $\Phi(\{t\}_P^r) = \{\Phi(t)\}_{\Phi(P)}^{\Phi(r)}$; where, as we mentioned earlier, elements of $T(D_M)$ are considered as elements of D_M.

We say that a random variable M is a *realization* of the term t through Φ, which we denote $M \lll_\Phi t$, if $M \in \Phi(t)$, and if also $t = \{t'\}_P^{\rho^A}$, then we further require that there is an $M' \lll_\Phi t'$ such that $\Phi(\rho^A)$ is independent of $\mathcal{F}^\eta_{J^\eta_{M'}}$ (where for $J^\eta_{M'}$ see the paragraph before Example 3).

We now define when a formula φ is *satisfied* by Φ:

- For any terms t_1, t_2, $\varphi \equiv t_1 = t_2$ is satisfied by Φ, iff $\Phi(t_1) = \Phi(t_2)$, and $\varphi \equiv t_1 \sqsubseteq t_2$ is satisfied by Φ iff $\Phi(t_1) \sqsubseteq_{\mathrm{C}} \Phi(t_2)$.

- For any term u and $acts = sends/receives$, $\varphi \equiv P\, acts\, u$ is satisfied by Φ iff there are stopping times J^η such that apart from sets of negligible probability, $Tr^\eta_{J^\eta(\omega)}(\omega)$ is of the form $A^\eta\, acts\, M^\eta(\omega)$ where $M := \{M^\eta\}_{\eta \in \mathsf{param}} \lll_\Phi u$ and $A := \{A^\eta\}_{\eta \in \mathsf{param}} \lll_\Phi P$. We will denote this as $Tr_J \lll_\Phi P\ acts\ u$.
- If $acts = generates$ then (in the previous item) u is a nonce ν, and so $M := \{M^\eta\}_{\eta \in \mathsf{param}} \lll_\Phi u$ means $M \in \Phi(\nu)$ in this case, and we further require that M^η be *independent* up to negligible probability of $\mathcal{F}^\eta_{J-1}$ for all η. (That is there is an $N \approx M$ such that N^η is independent of $\mathcal{F}^\eta_{J-1}$.)
- $\varphi \equiv \beta_1, ..., \beta_n$ sequence of actions is satisfied by Φ if each of β_k ($k = 1, ..., n$) is satisfied by Φ, and if J_k is the stopping time belonging to β_k, then we require that $J_k < J_l$ whenever $k < l$ (that is, for each $\eta \in \mathsf{param}$ and $\omega \in \Omega^\eta$, $J^\eta_k(\omega) < J^\eta_l(\omega)$.
- For any formulas φ, φ_1, φ_2, $\neg\varphi$ is satisfied by Φ iff φ is not satisfied by Φ; $\varphi_1 \vee \varphi_2$ is satisfied by Φ iff either φ_1 is satisfied by Φ or φ_2 is satisfied by Φ; $\varphi_1 \wedge \varphi_2$ is satisfied by Φ iff φ_1 is satisfied by Φ and φ_2 is satisfied by Φ. $\varphi_1 \rightarrow \varphi_2$ is satisfied by Φ iff $\neg\varphi_1 \vee \varphi_2$ is satisfied by Φ.
- If φ is a formula, m a bound variable, and φ' is obtained from φ by substituting m for every occurrence in φ of some free variable m' of the same sort as m, then $\forall m\varphi'$ (or $\exists m\varphi'$, resp.) is satisfied by Φ iff φ is satisfied by each (or some, resp.) Φ' extension of $\Phi_\mathcal{C}$ that only differs from Φ on m'.

A formula φ is *true* in the structure $\mathfrak{M}$, iff φ is satisfied by every Φ extension of $\Phi_\mathcal{C}$.

If in a structure, the Basic Protocol Logic axioms are true (in which case the structure is called *model*), then by standard arguments of first order logic, it follows that everything provable in the syntax is true in the model. In particular, if the query form is provable in the syntax, then it must be true in any model. We now turn our attention to whether the axioms are satisfied by a structure.

Satisfaction of the Term axioms. Most of these axioms are trivially satisfied because of the properties of equality, subexpression relation, pairing and encryption, and our assumptions of the execution. The axioms containing encryptions (other then the first item) are true, because of CCA-2 security, the encryptions cannot produce distributions that are identical (except for negligible probability) with interpretations of other terms.

Satisfaction of the Ordering axiom. Suppose that there is an extension Φ such that the formula $Q_2\, sends/receives\, m; Q_1\, generates\, n$ is satisfied as well as the formula $n \sqsubseteq m$. Then, there are stopping times J^η_1, J^η_2 such that $Tr_{J_1} \lll_\Phi Q\ sends/receives\ m$, and $Tr_{J_2} \lll_\Phi Q\ generates\ n$, and $J_1 < J_2$. Then, $Tr_{J_1} \lll_\Phi Q\ sends/receives\ m$ implies that there is $M \lll_\Phi m$ such that M^η is measurable with respect to $\mathcal{F}^\eta_{J_1}$ and since $n \sqsubseteq m$ is satisfied, some $N \in \Phi(n)$ can be obtained as a series of decryptions and breaking up pairs from M. Since there is no new randomness used there, N^η only depends on the randomness until J_1, so N^η is measurable with respect to $\mathcal{F}^\eta_{J_1}$. But, $Tr_{J_2} \lll_\Phi Q\ generates\ n$ implies that $\Phi(n)$ has an element N'^η measurable with respect to $\mathcal{F}^\eta_{J_2}$ and independent of $\mathcal{F}^\eta_{J_2-1}$, and hence independent of $\mathcal{F}^\eta_{J_1}$ and of N^η. So, N and N' only differ up to negligible probability, but N^η and N'^η are independent for all η, which is impossible.

Satisfaction of the Nonce verification axioms. In order to show that the axioms are satisfied, we use the assumption that regular participants encrypt with a CCA-2 secure encryption scheme. Suppose there is a Φ such that the premise of the axioms are satisfied by Φ, but the conclusion is not. Then, if the conclusion is not satisfied, that means that with non-negligible probability, $\{m_6\}^{r^A}_B$ does not go through B. The premise however says that n_1 shows up in m_5 later, which can be recovered from there up to negligible probability via a series of de-coupling and decryption such that $\{m_6\}^{r^A}_B$

does not have to be decrypted. We have to show that a PPT algorithm can be constructed that breaks CCA-2 security. The algorithm that breaks CCA-2 security is simply the protocol execution itself with the following modifications:

- The decryption oracle (that the algorithm may access according to the definition of CCA-2 security) does the job for all decryptions with the private key d_B.
- The algorithm generates two samples of n_1 when the protocol execution samples n_1.
- When the protocol execution is to produce $\{m_6\}_B^{r^A}$, compute two samples of the realization of m_6 using the two samples of n_1 and using the same samples for the other parts of m_6.
- Submit to the encryption oracle of the CCA-2 game the pair of samples of m_6, and use the ciphers that it outputs whenever $\{m_6\}_B^{r^A}$ occurs again.
- If the sample for $\{m_6\}_B^{r^A}$ goes through B, terminate. If not, continue until the Q receives the sample for m_5.
- Recover the sample for n_1 via de-coupling and decryption using the decryption oracle if necessary. The bit string hence obtained is the one that was in the plaintext encrypted by the oracle, so the bit value b of the game can be determined.

If the conclusion of the axiom is not satisfied, then this algorithm has non-negligible probability of winning the CCA-2 game.

In order to show the validity of the second nonce-verification axiom, we have to use the modified version of CCA-2 (equivalent to the original) when there are two encryption - decryption pairs of oracles, each corresponding to independently generated encryption key - decryption key pairs. The algorithm then is the following:

- The decryption oracles (that the algorithm may access according to the modified definition of CCA-2 security) do the job for all decryptions with the private keys d_B and d_C.
- The algorithm generates two samples of n_1 when the protocol execution samples n_1.
- When the protocol execution is to produce $\{m_6\}_B^{r^A}$, compute two samples of the realization of m_6 using the two samples of n_1 and using the same samples for the other parts of m_6.
- Submit to the first encryption oracle of the CCA-2 game the pairs of samples of m_6.
- Skip the step when B decrypts $\{m_6\}_B^{r^A}$.
- When $\{m_8\}_C^{r^B}$ is constructed, compute two samples of m_8 just as in the case of m_6. Stop if the samples have different length, otherwise submit the results to the second encryption oracle.
- Continue until C receives the sample for m_5.
- Recover the sample for n_1 via de-coupling and decryption using the decryption oracle if necessary. The bit string hence obtained is the one that was in the plaintext encrypted by the oracles, so the bit value b of the game can be determined.

This is again PPT algorithm given that the protocol execution was PPT, so it breaks CCA-2 security. Therefore, the Nonce-verification axioms hold.

Soundness Since the axioms are true in the structure $\mathfrak{M}$, by a standard argument of first order logic, the following theorem is true:

Theorem 1. *With our assumptions on the execution of the protocol, if the associated computational trace structure is $\mathfrak{M} = (\Pi, [\cdot,\cdot], \mathbf{Pr}, \mathcal{P}, \mathcal{N}, \mathcal{R}_g, Tr, \Phi_{\mathcal{C}})$, then, if a formula is provable in the syntax with first-order predicate logic and axioms (I), (II), (III), then it is true in $\mathfrak{M}$.*

We would like to reflect again on the four points in the introduction: 1. Removing the bound n^η from the length of executions is a trivial step (change the finite sequence of the filtration to an infinite one, and the definition of measurability to the standard one for infinite spaces) in our framework, only the presentation of the definition of measurability is more involved in this case, that is why we chose to stick to the bound. 2. We did not introduce modal formulas here in the syntax, and it is our work in progress to extend our approach to PCL. As we keep track of the actual probability distributions and correlations, it should be no problem to define the semantics of modal formulas so that these axioms hold as long as the interpretations (distributions, not bit-strings) of b and $\mathrm{Send}(\hat{X}, t)$ are different up to negligible probability. 3. As we use filtrations, according to which random variables have to be measurable, dependence on the future is taken care of by measurability. 4. We required that the distribution of keys are measurable with respect to $\mathcal{F}_0^\eta$, and generated nonces are independent of the past, so the anomaly mentioned in the introduction here cannot happen as N and K must have independent interpretations. The reader may be worried that we don't require that the generated R has to be dependent of N as R is generated by the adversary or a corrupted participant. It is true that we could introduce another filtration that indicates the knowledge of the adversary up to a certain time, which may be needed in a more complex syntax (for example if we allow corrupted participants to generate their keys sometime in the middle), however, in BPL this is not necessary as this does not result in undesired coincidences and the proofs work even without this tool.

4 Conclusions

We have given a computational semantics to Basic Protocol Logic that uses stochastic structures, and showed a soundness theorem. In order to show that the axioms of BPL were true in the semantics, we had to modify BPL as the original axioms were not all computationally sound. We showed our method on BPL as it is simple enough for a first, concise presentation. We argued why this semantics looks more promising than the one by Datta et al. Next, we would like to apply our methods to the much more complex formal syntax of Protocol Composition Logic. A formal completeness theorem for BPL has also been provided in [23], we would like to investigate completeness in the computational case as well.

References

1. Website of `www.stanford.edu/~danupam/logic-derivation.html`.
2. M. Abadi, M. Baudet, and B. Warinschi. Guessing attacks and the computational soundness of static equivalence. In *Proceedings of the 9th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS '06)*. Springer-Verlag, March-April 2006. To appear.
3. M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, January 2002. Preliminary version presented at IFIP TCS'00.
4. P. Adão, G. Bana, J. Herzog, and A. Scedrov. Soundness of formal encryption in the presence of key-cycles. In S. De Capitani di Vimercati, P. Syverson, and D. Gollmann, editors, *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS)*, volume 3679 of *Lecture Notes in Computer Science*, pages 374–396, Milan, Italy, September 12–14 2005. Springer.
5. P. Adão, G. Bana, and A. Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In *Proceedings of the 18th IEEE Computer Security Foundations Workshop (CSFW)*, pages 170–184, Aix-en-Provence, France, June 20–22 2005. IEEE Computer Society Press.
6. M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW)*, pages 204–218, Pacific Grove, CA, USA, June 28–30 2004. IEEE Computer Society Press. Full version available at IACR ePrint Archive, Report 2004/059.
7. M. Backes and B. Pfitzmann. Relating symbolic and cryptographic secrecy. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P)*, pages 171–182, Oakland, CA, USA, May 8–11 2005. IEEE Computer Society Press. Full version available at IACR ePrint Archive, Report 2004/300, November 2004.

8. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In S. Jajodia, V. Atluri, and T. Jaeger, editors, *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, pages 220–230, Washington D.C., USA, October 27–30 2003. ACM Press. Full version available at IACR ePrint Archive, Report 2003/015, January 2003.
9. M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580, pages 652–663. Springer-Verlag, July 2005.
10. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting. In *Proceedings of EUROCRYPT'00*, pages 258–274. Springer-Verlag, 2000.
11. C. Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.
12. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, Las Vegas, NV, USA, October 14–17 2001. IEEE Computer Society Press. Full version available at IACR ePrint Archive, Report 2000/067.
13. Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key exchange protocols. In *Proceedings, Theory of Cryptography Conference (TCC)*, March 2006.
14. I. Cervesato, C. Meadows, and D. Pavlovic. An encapsulated authentication logic for reasoning about key distribution protocols. In *Proceedings of the 18th IEEE Computer Security Foundations Workshop*, pages 48–61, 2005.
15. V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In M. Sagiv, editor, *Proceedings of the 14th European Symposium on Programming (ESOP)*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171, Edinburgh, UK, April 4–8 2005. Springer.
16. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security (Special Issue of Selected Papers from CSFW-16)*, 13:423–482, 2005.
17. A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In L. Caires, G. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings of the The 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3580 of *Lecture Notes in Computer Science*, pages 16–29, Lisbon, Portugal, July 11–15 2005. Springer.
18. D. Dolev and A. C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983. Preliminary version presented at FOCS'81.
19. N.A. Durgin, J.C. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11:677–721, 2003.
20. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and Systems Sciences*, 28(2):270–299, April 1984. Preliminary version presented at STOC'82.
21. J. D. Guttman and F. J. Thayer Fábrega. Authentication tests. In *IEEE Symposium on Security and Privacy*, pages 96–109, 2002.
22. J. D. Guttman, F. J. Thayer, and L. D. Zuck. The faithfulness of abstract protocol analysis: Message authentication. In P. Samarati, editor, *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, pages 186–195, Philadelphia, PA, USA, November 05–08 2001. ACM Press.
23. K. Hasebe and M. Okada. Completeness and counter-example generations of a basic protocol logic. In *6th International Workshop on Rule-Based Programming (RULE'05)*, volume 147(1), pages 73–92. Elsevier Science, 2005.
24. P. Laud. Encryption cycles and two views of cryptography. In *Proceedings of the 7th Nordic Workshop on Secure IT Systems (NORDSEC)*, number 31 in Karlstad University Studies, pages 85–100, Karlstad, Sweden, November 7–8 2002.
25. P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy (S&P)*, pages 71–85, Oakland, CA, USA, May 9–12 2004. IEEE Computer Society Press.
26. G. Lowe. A hierarchy of authentication specifications. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.
27. D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–130, 2004. Preliminary version presented at WITS'02.
28. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.
29. T. Woo and S. Lam. Verifying authentication protocols: Methodology and example. In *Proceedings of the International Conference on Network Protocols*, 1993.