# Computational Semantics for First-Order Logical Analysis of Cryptographic Protocols

Gergei Bana, Koji Hasebe, and Mitsuhiro Okada

[1] SQIG - Instituto de Telecomunicações and
Department of Mathematics, IST, Technical University of Lisbon, Portugal
[2] Graduate School of Systems and Information Engineering, University of Tsukuba, Japan
[3] Department of Philosophy, Keio University, Tokyo, Japan
gbana@math.ist.utl.pt hasebe@iit.tsukuba.ac.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 As opposed to earlier treatments, we introduce a new, fully probabilistic method to assign computational semantics to the syntax. 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. We present this via considering a simple example of such a formal model, the Basic Protocol Logic by K. Hasebe and M. Okada [20], but we think the technique is suitable for a wide range of formal methods for protocol correctness proofs. We first review our framework of proof-system, BPL, for proving correctness of authentication protocols, and provide computational semantics. Then we give a full proof of the soundness theorem. We also comment on the differences of our method and that of Computational PCL.

**Keywords.** cryptographic protocols, formal methods, first order logic, computational semantics

## 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 consider the relationship of the two models when formal security analysis is done with first order logic. Protocol correctness is analyzed by defining a syntax with adding some additional axioms (expressing security properties etc.) to the usual axioms and inference rules of first order logic and then proving some security property directly, instead of eliminating the possibility of successful (formal) adversaries. 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 [13] gave a computational semantics to the syntax of their Protocol Composition Logic of [16, 12] (cf. also [1] for *a protocol composition logic project overview*). In their treatment, every action by the honest participants is recorded on each (non-deterministic) execution trace, 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 formula is true on a particular trace (more exactly this is only true for a formula not containing their predicate $Indist$), and they say the formula is true in the model if it is true on an overwhelming number of traces. This method however, since it focuses on coincidences on individual traces, discards a large amount of information carried by the probabilistic structure of the protocol execution, and defines satisfaction and validity of formulas ignoring that information. As the comparisons are done on each trace separately it is not possible to track independence, correlations. But there are more serious problems too, as we will discuss later - such as, some of the syntactic axioms are defined through the semantics. We do not claim that it is impossible to fix these issues in their framework, but we suggest a different viewpoint in which these issues can be easily eliminated.

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

We introduce our method on a rather simple syntax, namely, a somewhat modified version of Basic Protocol Logic (or BPL, for short) by K. Hasebe and M. Okada [20] and leave extensions to more complex situations such as the Protocol Composition Logic to future work. The reason for this is 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, important details of the formalization is not yet publicly 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 (PCL) of [16, 12, 11] for proving some aimed properties in the sense of [25, 22] within the framework of first order logic; all notions and assumptions are strictly formulated by the first order logical language explicitly. Contrary to PCL, in BPL there are no explicit encrypt, decrypt, match actions, only nonce generation, send and receive. The version which we utilized as a simple sample of formal rule-based model in this paper does not accomodate some correctness proofs such as secrecy-properties although one could extend BPL to support them. Besides the usual axioms and derivation rules of first order logic, further axioms set the behavior of equality and subterm relations of terms created via pairing and encryption, two nonce-veryfication axioms incorporating the notion that only the person with the correct decryption key can read what is inside an encryption, and an axiom about the order of events in traces. Although this system is very simple, given a protocol (as we will indicate in the case of agreement in NSL), the nonce-veryfication axioms forcing certain messages to be included in others, and then the term axioms restricting what a certain pair of terms in a subterm relation can possibly be, the required property can be verified.

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.

Although BPL is sound with respect to formal semantics as shown in [20] with the traditional (Tarskian) model-theoretic formal semantics based on the free term algebra domain, in order to

ensure soundness for computational semantics, some modifications of the original syntax of BPL were necessary:

1. Instead of denoting encryptions as $\{m\}_A$, which was used in the original version of the purely symbolic model-based BPL inference system, we indicate the random seed of the encryption as $\{m\}^r_A$ (as Datta et al. do). As it turned out, a consistent computational interpretation is much harder, if not impossible without the random seed in the syntax.

2. The original subterm and equiterm axioms were not all computationally sound so we just take a certain subset, the elements of which we know that they are computationally sound. We are not taking all the sound term axioms, as it is not known how to give a complete characterization of them. [4]

The original BPL also proved completeness for formal semantics with the original set of axioms, however, we do not consider completeness in this paper. It is an open question whether anything about completeness can be said in the computational case.

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.

**Related Work.** Formal methods emerged from the seminal work of Dolev and Yao [15], whereas computational cryptography grew out of the work of Goldwasser and Micali [17]. 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" [23, 5], generalizing for weaker, more realistic encryptions schemes [5], considering purely probabilistic encryptions [19, 5], including limited models for active adversaries [21], addressing the issue of forbidding key-cycles [4], considering algebraic operations and static equivalence [8, 2]. Other approaches including active adversaries are considered by Backes et al. and Canetti in their *reactive simulatability* [6] and *universal composability* [10] frameworks, respectively. Non trace properties were investigated elsewhere too, however, not in the context of first order logic. A brief account of this present work was given in [7].

**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.

[4] The uses of subterm and equiterm relations, such as $s \sqsubseteq t$ and $s = t$, are essential for correctness proofs of protocols in general, including BPL and PCL. Any symbolic term model, hence, should reflect the symbolic term structures, and such a term model maybe called a "standard" model with respect to subterm and equiterm relations. BPL's symbolic semantics takes such standard term model which also satisfy certain properties for nonce-verifications, which are listed as non-logical axioms in our BPL syntax. Our result 2 above shows that only the truth of a certain useful subset of the subterm theory axioms is preserved under computational interpretation. As we will show, the nonce-verification axioms turn out to be sound.

## 2 Basic Protocol Logic

In this section, we present the syntax of Basic Protocol Logic modified to be suitable for computational interpretation. For the original BPL, please consult [20].

### 2.1 Language

**Sorts and terms.** Our language is order-sorted, with sorts `coin` , `name`, `nonce` and `message` such that terms of sorts `name` and `nonce` are terms of sort `message`. Let $\mathcal{C}_{\texttt{name}}$ be a finite set of constants of sort `name` (which represent principal names), and $\mathcal{C}_{\texttt{nonce}}$ a finite set of constants of sort `nonce`. Let $\mathcal{C}_{\texttt{coin}}$ be a finite set of constants of sort `coin`. The sort `coin` represent the random input of encryptions. We require countably infinite 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 `name`, $N, N', \ldots, N_1, N_2, \ldots$ ($n, n', \ldots, n_1, n_2, \ldots$, resp.) denote constants (variables, resp.) of sort `nonce`, $r, r', \ldots, r_1, r_2, \ldots$ ($s, s', \ldots, s_1, s_2, \ldots$, resp.) denote constants of sort `coin` (variables of sort `coin`, resp.). The symbols $m, m', \ldots, m_1, m_2, \ldots$ are used to denote variables of sort `message` and $M, M', \ldots, M_1, M_2, \ldots$ to denote constants of sort `message` (that is, either `name` or `nonce`). Let $P, P', \ldots, P_1, P_2, \ldots$ denote any term of sort `name`, let $\rho, \rho', \ldots, \rho_1, \rho_2, \ldots$ denote anything of sort `coin`, and let $\nu, \nu', \ldots$ denote any term of sort `nonce`. Compound terms of sort `message` are built from constants and variables, and are defined by the grammar:

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

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

**Formulas.** We introduce a number of predicate symbols: $P \ generates \ \nu$, $P \ receives \ t$, $P \ sends \ t$, $t = t'$, $t \sqsubset t'$, $t \sqsubset_P t'$, $t \sqsubset_{\neg P} t'$ and $|\overline{t_1 \sqsubset t_2 \sqsubset t_3}|$ which represent "$P$ generates a fresh value $\nu$ as a nonce", "$P$ receives a message of the form $t$", "$P$ sends a message of the form $t$", "$t$ is identical with $t'$", "$t$ is a subterm of $t'$", "$t$ is a subterm of $t'$ such that $t$ can be received from $t'$ decrypting only with the private key of $P$", "$t$ is a subterm of $t'$ such that $t$ can be received from $t'$ without decrypting with the private key of $P$", "$t_1 \sqsubset t_2 \sqsubset t_3$ and the only way $t_1$ occurs in $t_3$ is within $t_2$", respectively. The first three are called *action predicates*, and the meta expression $acts$ is used to denote one of the action predicates: $generates$, $receives$ and $sends$. We also introduce the *trace predicate* $P_1 \ acts_1 \ t_1; P_2 \ acts_2 \ t_2; \cdots ; P_k \ acts_k \ t_k$. A trace predicate is used to represent a sequence of the principals' actions such as "$P$ sends a message $m$, and after that, $Q$ receives a message $m'$". Atomic formulas are either of the form $P_1 \ acts_1 \ t_1; P_2 \ acts_2 \ t_2; \cdots ; P_k \ acts_k \ t_k$, or $t = t'$, or $t \sqsubset t'$, or $t \sqsubset_P t'$, or $t \sqsubset_{\neg P} t'$ or $|\overline{t_1 \sqsubset t_2 \sqsubset t_3}|$. The first one we also call *trace formula*. We also use $\alpha_1; \cdots ; \alpha_k$ (or $\boldsymbol{\alpha}$ in short) to denote $P_1 \ acts_1 \ t_1; \cdots ; P_k \ acts_k \ t_k$ (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 $j : \{1, ..., m\} \rightarrow \{1, ..., n\}$ such that $\alpha_i \equiv \beta_{j(i)}$. Formulas are defined by

$$\varphi \ ::= \ \boldsymbol{\alpha} \Big| t_1 = t_2 \Big| t_1 \sqsubset t_2 \Big| t_1 \sqsubset_P t_2 \Big| t_1 \sqsubset_{\neg P} t_2 \Big| |\overline{t_1 \sqsubset t_2 \sqsubset t_3}| \Big| \neg\varphi \Big| \varphi \wedge \varphi \Big| \varphi \vee \varphi \Big| \varphi \rightarrow \varphi \Big| \forall m \varphi \Big| \exists m \varphi$$

where $m$ is any variable. Those variables in a formula that are bound by the binding operators $\exists$ and $\forall$ will be referred to as bound variables, those that are not will be referred to as free variables.

We use the meta expression $\varphi[\boldsymbol{m}]$ to indicate the list of all free variables $\boldsymbol{m}$ occurring in $\varphi$. We will also use $\varphi[\boldsymbol{M}, \boldsymbol{m}]$ to specify some constants $\boldsymbol{M}$ that occur in $\varphi$ where $\boldsymbol{m}$ again is all free variables in $\varphi$.

Finally, *order-preserving merge* of trace formulas $\boldsymbol{\alpha}$ ($\equiv \alpha_1; \cdots ; \alpha_l$) 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 $j^\alpha$ and $j^\beta$ cover $\{1, ..., n\}$. $\delta$ is called a *strict order-preserving merge* if, furthermore, the ranges of $j^\alpha$ and $j^\beta$ are disjoint.

**Roles.** Roles of principals are described by trace formulas of the form $\boldsymbol{\alpha}^A \equiv A\ acts_1\ t_1; \cdots ; A\ acts_k\ t_k$. Honest principals (those who generate keys, encryptions, nonces honestly, and don't share information with the adversary) are denoted by constants. Nonces and coins that these participants generate are also denoted by constants. Protocols are a set of roles together with a list of values that the principals have to agree on.

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

We consider the Needham-Schroeder-Lowe public key protocol [24], whose informal description is as follows.

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

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.

$Init^A_{NSL}[Q_2, N_1, n_2, r_1, s_2, r_3] \equiv$
$A\ generates\ N_1; A\ sends\ \{N_1, A\}^{r_1}_{Q_2}; A\ receives\ \{N_1, n_2, Q_2\}^{s_2}_A; A\ sends\ \{n_2\}^{r_3}_{Q_2}$

$Resp^B_{NSL}[Q_1, n_1, N_2, s_1, r_2, s_3] \equiv$
$B\ receives\ \{n_1, Q_1\}^{s_1}_B; B\ generates\ N_2; B\ sends\ \{n_1, N_2, B\}^{r_2}_{Q_1}; B\ receives\ \{N_2\}^{s_3}_B$

They further have to agree that $Q_1 = A$, $Q_2 = B$, $n_1 = N_1$, $n_2 = N_2$.

*Remark 1.* Notice that, for example, in the responder's role, we wrote $B\ receives\ \{n_1, Q_1\}^{s_1}_B$ instead of $B\ receives\ \{m, Q_1\}^{s_1}_B$, although $n_1$ may come from the adversary. This is, because we will assume in the semantics, that because of tagging, it is recognizable whether a string is a nonce or not. But, the distribution of $n_1$, if coming from the adversary, may not follow the correct distribution of nonces.

### 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).

*Remark 2.* For the axioms below to be more understandable, we make a few remarks about the semantics that we will define later. For each $\eta$, names will be interpreted as some constant bit string names of the participating principals, such that the principals corresponding to constants will generate nonces, keys and encrypt honestly. Other possible principles may be malicious, creating encryptions and nonces dishonestly. Nonces will have to have a certain fixed length, the interpretation of nonce constants will have to have the correct distribution and be independent of what happened earlier. The interpretation of coins will have to have the correct form for the random feed into the encryption, and further, constants of sort `coin` will have to have the correct

distribution, and when used for encryption, such a constant coin will have to have a distribution that is independent of the distribution of encrypted plaintext as well as independent of everything that happened earlier. The public keys of constants will also have to have the correct distributions and be generated at the very beginning. The interpretation of encryptions and pairing are defined the intuitive way.

**(I) Term axioms.** Consider the set $\bar{\mathcal{C}}$ of all variables and constants of each of sort `name`, sort `nonce` and sort `coin`. Let $\bar{\mathcal{A}}$ be the free algebra constructed from $\bar{\mathcal{C}}$ via $\langle\cdot,\cdot\rangle$ and $\{\cdot\}_{\cdot}$ (with the appropriate sorts in the indexes of the encryption terms) not including constants and variables of sort `coin`. The elements of $\bar{\mathcal{A}}$ are of sort `message`. Let $\sqsubseteq^{\bar{\mathcal{A}}}$ denote the natural subterm relation in $\bar{\mathcal{A}}$. Let $t \sqsubseteq^{\bar{\mathcal{A}}}_{P} t'$ mean that $t \sqsubseteq^{\bar{\mathcal{A}}} t'$ such that $t$ can be received from $t'$ by decrypting encryptions by the key of $P$ only. Let $t \sqsubseteq^{\bar{\mathcal{A}}}_{\neg P} t'$ mean that $t \sqsubseteq^{\bar{\mathcal{A}}} t'$ such that $t$ can be received from $t'$ by decrypting encryptions that are not done with the key of $P$.

Let $|t_1 \sqsubseteq^{\bar{\mathcal{A}}} t_2 \sqsubseteq^{\bar{\mathcal{A}}} t_3|$ mean that $t_1 \sqsubseteq^{\bar{\mathcal{A}}} t_2 \sqsubseteq^{\bar{\mathcal{A}}} t_3$ and the only way $t_1$ occurs in $t_3$ is within $t_2$. That is: $|t_1 \sqsubseteq^{\bar{\mathcal{A}}} t_2 \sqsubseteq^{\bar{\mathcal{A}}} t_3| := t_1 \sqsubseteq^{\bar{\mathcal{A}}} t_2 \sqsubseteq^{\bar{\mathcal{A}}} t_3 \wedge \forall t(t_1 \sqsubseteq^{\bar{\mathcal{A}}} t \sqsubseteq^{\bar{\mathcal{A}}} t_3 \rightarrow t_2 \sqsubseteq^{\bar{\mathcal{A}}} t \vee (t \sqsubseteq^{\bar{\mathcal{A}}} t_2 \wedge \forall t_4(t_2 \sqsubseteq^{\bar{\mathcal{A}}} t_4 \rightarrow \langle t, t_4\rangle \not\sqsubseteq^{\bar{\mathcal{A}}} t_3 \wedge \langle t_4, t\rangle \not\sqsubseteq^{\bar{\mathcal{A}}} t_3)))$.

We postulate the following term axioms. Here, and also later by using $\sqsubseteq_{(\neg)P}$ we mean two sentences, one with and one without $\neg$. Let $\boldsymbol{m}$ be all variables occurring in the corresponding terms. We require these for all $A, B \in \mathcal{C}_{\texttt{name}}$:

(a) If $t = t'$ is true in $\bar{\mathcal{A}}$, then $\forall \boldsymbol{m}(t = t')$ is axiom. If $t \sqsubseteq^{\bar{\mathcal{A}}} t'$ is true in $\bar{\mathcal{A}}$, then $\forall \boldsymbol{m}(t \sqsubseteq t')$ is axiom. If $t \sqsubseteq^{\bar{\mathcal{A}}}_{P} t'$ is true in $\bar{\mathcal{A}}$, then $\forall \boldsymbol{m}(t \sqsubseteq^{\bar{\mathcal{A}}}_{P} t')$ is axiom. If $\forall \boldsymbol{m} Q(Q \neq P_1 \wedge ... \wedge Q \neq P_k \rightarrow t \sqsubseteq^{\bar{\mathcal{A}}}_{\neg Q} t')$ is true in $\bar{\mathcal{A}}$ for all (possibly equal) constant or variable substitutions to $\boldsymbol{m}$ and $Q$, then it is an axiom.
(b) $\forall \boldsymbol{m}(t_1 = t_2 \rightarrow t_2 = t_1)$, $\forall \boldsymbol{m}(t_1 = t_2 \wedge t_2 = t_3 \rightarrow t_1 = t_3)$, $\forall \boldsymbol{m}(t_1 \sqsubseteq t_2 \wedge t_2 \sqsubseteq t_3 \rightarrow t_1 \sqsubseteq t_3)$,
(c) $\forall \boldsymbol{m} P(t_1 \sqsubseteq_{(\neg)P} t_2 \rightarrow t_1 \sqsubseteq t_2)$, $\forall \boldsymbol{m} P(t_1 = t_2 \rightarrow t_1 \sqsubseteq_{(\neg)P} t_2)$, $\forall \boldsymbol{m} P(t_1 \sqsubseteq_{(\neg)P} t_2 \wedge t_2 \sqsubseteq_{(\neg)P} t_3 \rightarrow t_1 \sqsubseteq_{(\neg)P} t_3)$
(d) $\forall \boldsymbol{m} Q s s'(\{t_1\}^s_Q = \{t_2\}^{s'}_Q \rightarrow t_1 = t_2)$,
(e) $\forall \boldsymbol{m}(\langle t_1, t_2\rangle = \langle t_3, t_4\rangle \rightarrow t_1 = t_3 \wedge t_2 = t_4)$
(f) $\forall \boldsymbol{m} Q s(\{t\}^s_Q \neq \langle t_1, t_2\rangle)$, $\forall \boldsymbol{m} Q s n(\{t\}^s_Q \neq n)$, $\forall \boldsymbol{m} Q Q' s(\{t\}^s_Q \neq Q')$
(g) $\forall \boldsymbol{m} n(\langle t_1, t_2\rangle \neq n)$, $\forall \boldsymbol{m} Q(\langle t_1, t_2\rangle \neq Q)$
(h) $\forall \boldsymbol{m}(t \sqsubseteq \langle t_1, t_2\rangle \rightarrow t \sqsubseteq t_1 \vee t \sqsubseteq t_2 \vee t = \langle t_1, t_2\rangle)$, $\forall \boldsymbol{m} Q s(t_1 \sqsubseteq \{t_2\}^s_Q \rightarrow t_1 = \{t_2\}^s_Q \vee \exists m Q' s'(\{t_2\}^s_Q = \{m\}^{s'}_{Q'} \wedge t_1 \sqsubseteq m))$
(i) $\forall \boldsymbol{m} P(t \sqsubseteq_P \langle t_1, t_2\rangle \rightarrow t \sqsubseteq_P t_1 \vee t \sqsubseteq_P t_2 \vee t = \langle t_1, t_2\rangle)$, $\forall \boldsymbol{m} Q s P(t_1 \sqsubseteq_P \{t_2\}^s_Q \rightarrow t_1 = \{t_2\}^s_Q \vee \exists m s'(\{t_2\}^s_Q = \{m\}^{s'}_P \wedge t_1 \sqsubseteq_P m))$, $\forall \boldsymbol{m} s P(t_1 \sqsubseteq_P \{t_2\}^s_P \rightarrow t_1 = \{t_2\}^s_P \vee t_1 \sqsubseteq_P t_2))$
(j) $\forall \boldsymbol{m} P(t \sqsubseteq_{\neg P} \langle t_1, t_2\rangle \rightarrow t \sqsubseteq_{\neg P} t_1 \vee t \sqsubseteq_{\neg P} t_2 \vee t = \langle t_1, t_2\rangle)$, $\forall \boldsymbol{m} Q s P(t_1 \sqsubseteq_{\neg P} \{t_2\}^s_Q \rightarrow t_1 = \{t_2\}^s_Q \vee \exists m Q' s'(Q' \neq P \wedge \{t_2\}^s_Q = \{m\}^{s'}_{Q'} \wedge t_1 \sqsubseteq m))$
(k) $\forall m n(m \sqsubseteq n \rightarrow m = n)$, $\forall m Q(m \sqsubseteq Q \rightarrow m = Q)$
(l) $\forall \boldsymbol{m} |t_1 \sqsubseteq t_2 \sqsubseteq t_3|$ is an axiom if $|t_1 \sqsubseteq^{\bar{\mathcal{A}}} t_2 \sqsubseteq^{\bar{\mathcal{A}}} t_3|[\boldsymbol{M}/\boldsymbol{m}]$ holds for all $\boldsymbol{M}$ vector of constans of appropriate sort.

**(II) Rules for trace formulas.** We postulate that $\boldsymbol{\beta} \rightarrow \boldsymbol{\alpha}$ for $\boldsymbol{\alpha} \subseteq \boldsymbol{\beta}$ and $\boldsymbol{\gamma_1} \vee \cdots \vee \boldsymbol{\gamma_n} \leftrightarrow \boldsymbol{\alpha} \wedge \boldsymbol{\beta}$, where $\boldsymbol{\gamma_i}$'s are the list of order-preserving merges of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. 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:**

$\forall Q_1 Q_2 n m(n \sqsubseteq m \rightarrow \neg(Q_2\ sends/receives/generates\ m; Q_1\ generates\ n))$.

**(2) Nonce verification 1:** For each $A$, $B$ constants of sort `name`, $r$ constant of sort `coin`, we postulate

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

**(3) Nonce verification 2:** For each $A$, $B$ $C$ of sort `name` (where $A$ and $C$ may coincide), $r_1$, $r_2$ constants of sort `coin`, we postulate we postulate

$\forall n_1 m_5 m_6 m_8 (A \textit{ generates } n_1; C \textit{ receives } m_5 \ \wedge \ n_1 \sqsubseteq_{\neg B} m_5$
$\quad \wedge \forall m_7 (A \textit{ sends } m_7 \wedge n_1 \sqsubseteq m_7 \rightarrow |\overline{n_1 \sqsubseteq \{m_6\}^{r_1}_B \sqsubseteq m_7}|)$
$\quad \wedge \forall m_4 (B \textit{ sends } m_4 \wedge n_1 \sqsubseteq m_4 \rightarrow |\overline{n_1 \sqsubseteq \{m_8\}^{r_2}_C \sqsubseteq m_4}|)$
$\quad \wedge \forall m_{10} (\neg (C \textit{ sends } m_{10} \wedge n_1 \sqsubseteq m_{10}) \vee A = C)$
$\quad \rightarrow \{m_8\}^{r_2}_C \sqsubseteq_C m_5$

There are other possible axiomatizations, but the authors of [20] found this particularly useful (more exactly a somewhat less general version). The meaning of the Ordering axiom is clear. Nonce verfication 1 and 2 are based on the idea of the authentication-tests [18]. Nonce verification 1 means that if $A$ generated a nonce $n_1$ that $Q$ received in $m_5$, and $A$ only sent $n_1$ encrypted with the public key of $B$ always in a given form $\{m_6\}^r_B$, and $Q$ received this nonce in some other form, then the encrypted nonce had to go through $B$, and before that, it had to be actually sent out by $A$. 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. Nonce verification 2 means that with the premises of Nonce verification 1, and if $B$ sends $n_1$ only inside $\{m_8\}^{r_2}_C$, and $C$ never sends $n_1$ unless $C = A$, then $C$ had to receive $\{m_8\}^{r_2}_C$ so that it is accessible to him.

### 2.3 Query form and correctness properties

We introduce a general form of formulas, called *query form*, to represent 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 1. (Query form)** *Query form is a formula of the following form:* $\exists \boldsymbol{m} Honest(\boldsymbol{\alpha}^A) \wedge \boldsymbol{\beta}^B \wedge Only(\boldsymbol{\beta}^B) \rightarrow \boldsymbol{\gamma}$

We present the precise definition of $Only(\boldsymbol{\alpha}^B)$ and of $Honest(\boldsymbol{\alpha}^A)$ in the Appendix. $Only(\boldsymbol{\alpha}^B)$ means that $B$ performs only the actions of $\boldsymbol{\alpha}^B$, and nothing else, whereas $Honest(\boldsymbol{\alpha}^A)$ 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 $\boldsymbol{\alpha}^A$". For example, from responder's (namely, $B$'s) view, the non-injective agreement of the protocol $\Pi = \{\boldsymbol{\alpha}^A[B/Q_2, \boldsymbol{N_1}, \boldsymbol{n_2}, \boldsymbol{r_1}, \boldsymbol{s_2}], \boldsymbol{\beta}^B[A/Q_1, \boldsymbol{n_1}, \boldsymbol{N_2}, \boldsymbol{s_1}, \boldsymbol{r_2}]\}$ can be described as the following formula: $\exists \boldsymbol{n_1 n_2 s_1 s_2} Honest(\boldsymbol{\alpha}^A)[Q_2, \boldsymbol{N_1}, \boldsymbol{n_2}, \boldsymbol{r_1}, \boldsymbol{s_2}] \wedge (\boldsymbol{\beta}^B \wedge Only(\boldsymbol{\beta}^B)[A/Q_1, \boldsymbol{n_1}, \boldsymbol{N_2}, \boldsymbol{s_1}, \boldsymbol{r_2}]) \rightarrow \boldsymbol{\alpha}^A[B/Q_2, \boldsymbol{N_1}, \boldsymbol{N_2}/\boldsymbol{n_2}, \boldsymbol{r_1}, \boldsymbol{r_2}/\boldsymbol{s_2}] \wedge \boldsymbol{n_1} = \boldsymbol{N_1}$

*Example 2.* **(Agreement property in the responder's view of the NSL protocol)**
The initiator's honesty of the NSL protocol is $Honest(Init^A_{NSL})[Q_1, N_1, n_2, r_1, s_2, r_3] \ \equiv$

$$\left( \begin{pmatrix} \forall n_3 \neg (A \textit{ generates } n_3) \\ \wedge \forall m_4 \neg (A \textit{ sends } m_4) \\ \wedge \forall m_5 \neg (A \textit{ receives } m_5) \end{pmatrix} \vee \begin{pmatrix} A \textit{ generates } N_1; A \textit{ sends } \{N_1, A\}^{r_1}_{Q_1} \\ \quad \wedge \forall n_3 (A \textit{ generates } n_3 \rightarrow n_3 = N_1) \\ \quad \wedge \forall m_4 (A \textit{ sends } m_4 \rightarrow m_4 = \{N_1, A\}^{r_1}_{Q_1}) \\ \quad \wedge \forall m_5 \neg (A \textit{ receives } m_5) \end{pmatrix} \right)$$

$$\vee \left( \begin{array}{l} A \textit{ generates } N_1; A \textit{ sends } \{N_1, A\}^{r_1}_{Q_1}; A \textit{ receives } \{N_1, n_2, Q_1\}^{s_2}_{A}; A \textit{ sends } \{n_2\}^{r_3}_{Q_1} \\ \quad \wedge \forall n_3 (A \textit{ generates } n_3 \rightarrow n_3 = N_1) \\ \quad \wedge \forall m_4 (A \textit{ sends } m_4 \rightarrow m_4 = \{N_1, A\}^{r_1}_{Q_1} \vee m_4 = \{n_2\}^{r_2}_{Q_1}) \\ \quad \wedge \forall m_5 (A \textit{ receives } m_5 \rightarrow m_5 = \{N_1, n_2, Q_1\}^{s_2}_{A}) \end{array} \right)$$

We refer to Remark 1 for an explanation why nonce variables are used for even those nonces that are sent by the adversary.

The main steps of proving agreement from the responder's view are the following:

$$\exists n_1 s_1 s_3 Resp^B_{NSL} \wedge Only(Resp^B_{NSL})[A/Q_1, n_1, N_2, s_1, r_2, s_3]$$

implies by the 1st nonce verification axiom that

$$\exists m_3 m_4 (B \textit{ sends } \{n_1, N_2, B\}^{r_2}_{A}; A \textit{ receives } m_3; A \textit{ sends } m_4; B \textit{ receives } \{N_2\}^{s_3}_{B} \wedge \\ \wedge \{n_1, N_2, B\}^{r_2}_{A} \sqsubseteq_A m_3 \wedge N_2 \sqsubseteq m_4)).$$

Then from this together with $\exists Q_1 n_2 s_2 Honest(Init^A_{NSL})[Q_1, N_1, n_2, r_1, s_2, r_3]$ it follows that

$$\{n_1, N_2, B\}^{r_2}_{A} \sqsubseteq_A \{N_1, n_2, Q_1\}^{s_2}_{A}.$$

From this, using the term axioms (f), (i) and (k), we get that $\{N_1, n_2, Q_1\}^{s_2}_{A} = \{n_1, N_2, B\}^{r_2}_{A}$, and then from (d) and (e) that $n_1 = N_1$, $n_2 = N_2$, $Q_1 = B$. Then with a similar argument $\{N_1, A\}^{r_1}_{B} = \{n_1, A\}^{s_1}_{B}$ and $\{n_2\}^{r_3}_{B} = \{N_2\}^{s_3}_{B}$ are also proven, from which finally the completed initiator's role, $Init^A_{NSL}[B, N_1, N_2, r_1, r_2, r_3]$ follows. That is, the initiator also finished the protocol and the values agree.

# 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* $\eta \in \mathsf{param} := \{1\}^* \equiv \mathbb{N}$ (which can be roughly understood as key-length).

**Definition 2 (Negligible Function).** *A function $f : \mathbb{N} \rightarrow \mathbb{R}$ 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* $[\cdot,\cdot] : \mathsf{strings} \times \mathsf{strings} \rightarrow \mathsf{strings}$. 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 plaintexts, ciphertexts, publickey and secretkey be nonempty subsets of strings. The set coins is some probability field of (possibly infinite) bit-strings that stands for coin-tossing, *i.e.* feeds randomness into the Turing-machines realizing the algorithms.

**Definition 3 (Encryption Scheme).** *A* computational asymmetric encryption scheme *is a triple of algorithms $\mathfrak{E} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where:*

- $\mathcal{K} : \mathsf{param} \times \mathsf{coins} \rightarrow \mathsf{publickey} \times \mathsf{secretkey}$ *is a key-generation algorithm with* $\mathsf{param} = \{1\}^*$,
- $\mathcal{E} : \mathsf{publickey} \times \mathsf{plaintexts} \times \mathsf{coins} \rightarrow \mathsf{ciphertexts} \cup \{\perp\}$ *is an encryption algorithm, and*
- $\mathcal{D} : \mathsf{secretkey} \times \mathsf{strings} \rightarrow \mathsf{plaintexts} \cup \{\perp\}$ *is such that for all* $(e,d) \in \mathsf{publickey} \times \mathsf{secretkey}$ *and* $c \in \mathsf{coins}$

$$\mathcal{D}(d, \mathcal{E}(e, m, c)) = m \textit{ for all } m \in \mathsf{plaintexts} \textit{ and } (e,d) \textit{ output of the key generation.}$$

*All these algorithms are computable in polynomial time with respect to the length of their input.*

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

**Definition 4 (Adaptive Chosen Ciphertext Security).** *A computational public-key encryption scheme* $\mathfrak{E} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *provides* indistinguishability under the adaptive chosen-ciphertext attack *if for all* $PPT$ *adversaries* $\mathsf{A}$ *and for all sufficiently large security parameter* $\eta$*:*

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

*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 the above definition, what the brackets of the probability contains, is a commonly used shorthand for the following game: First a public key-private key pair is generated on input $1^\eta$, as well as a random bit $b$ with probabilities $1/2$ – $1/2$. 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 $1/2$.

It was shown in [9] 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 Filtrations and Stopping Times

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.

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 runtime). So, here we assume that for each security parameter, there is a maximum number of execution steps $n^\eta$. 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 $\eta$ by $\Omega^\eta$, subsets of which are called events. Let $\mathcal{F}^\eta$ denote the set of all subsets of $\Omega^\eta$ (including the empty set). A subset containing only one element is called an *elementary event*. The set $\Omega^\eta$ is meant to include all randomness of an execution of the protocol. A *probability measure* $p^\eta$ 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 $\Omega^\eta$, 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 $\Omega^\eta$, and $\mathcal{F}_i^\eta \subseteq \mathcal{F}_{i+1}^\eta$. We also assume that $\mathcal{F}_{n^\eta}^\eta = \mathcal{F}^\eta$, that is, $\Omega$ does not include information irrelevant to the protocol execution. 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 $\Omega^\eta$ is a function on $\Omega^\eta$ taking some discrete value. Since $\mathcal{F}_i^\eta$ contains the events determined until step $i$, a random variable $g^\eta$ depends only on the randomness until $i$ exactly if $g^\eta$ 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 $g^\eta$ 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 $\omega$. This function should only depend on the past, so for each $i \in \{0, 1, ..., n^\eta\}$, we require that the set $[J^\eta = i] \in \mathcal{F}_i^\eta$. Such a function is called *stopping time*. The events that have occurred until the stopping time $J^\eta$ 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 $f^\eta$ depends only on the events until the stopping time $J^\eta$ iff for each $c$ in its range, $[f^\eta = c] \in \mathcal{F}_J^\eta$. Furthermore, a random variable $h^\eta$ on $\Omega^\eta$ is said to be independent of what happened until $J^\eta$ iff for any $S \in \mathcal{F}_J^\eta$ and a $c$ possible value of $h^\eta$, $\Pr([h^\eta = c] \cap S) = \Pr([h^\eta = c]) \Pr(S)$. Finally, it is easy to see that for each random variable $f^\eta$, there is a stopping time $J_f^\eta$ such that $f^\eta$ is measurable with respect to $\mathcal{F}_{J_f}^\eta$, and $J_f^\eta$ is minimal in the sense that $f^\eta$ is not measurable with respect to any other $\mathcal{F}_J^\eta$ if there is an $\omega$ such that $J^\eta(\omega) < J_f^\eta(\omega)$.

*Example 3.* Let coins be 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 $\Omega$, 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: $J(\ (a_1, a_2, a_3)\ ) = i$ if $a_i = 1$ and $a_k = 0$ for $k < i$, and $J(\ (a_1, a_2, a_3)\ ) = 3$ if $a_k = 0$ for all $k = 1, 2, 3$. The atoms of $\mathcal{F}_J$ are $(1, \cdot, \cdot)$, $(0, 1, \cdot)$, $\{(0, 0, 1)\}$ and $\{(0, 0, 0)\}$.

We will also have to assume that the stopping times are such that they are polynomially decidable, that is, in the execution of a PPT algorithm, the computation of value of a stopping time on an execution trace should not destroy the polynomial bound. This is not really a restriction as in case of security properties, stopping times are just decided simply by the position of the protocol execution, carrying out some decryptions, and matching values.

### 3.3 Stochastic Model for the Computational Execution of BPL

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 $n^\eta$. 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 $(e_A^\eta, d_A^\eta)$ is a pair of probability distributions on $\Omega^\eta$ 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 $\eta$. 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 $m^\eta$ for each security parameter $\eta$. We assume $m^\eta$ is at most polynomial in $\eta$, and $1/2^{m^\eta}$ is negligible. Let $\mathcal{N}$ be a set of elements of the form $\{N^\eta\}_{\eta\in\mathsf{param}}$ where $N^\eta : \Omega^\eta \to \{0,1\}^{m^\eta} \cup \{\bot\}$ (taking the value $\bot$ means $N^\eta$ has no bit-string value on that particular execution). This set describes the nonces that were generated during the execution of the protocol. The nonces generated by honest participants must have some fixed distribution (uniform for example) over set of bit strings with the given length and also have to be independent of what happened earlier when they are being generated, but we will require this later in the definition of interpretation of constants and at the definition of the satisfaction of the formula $A\ generates\ N$.

Let $\mathcal{R}$ be a set of elements of the form $R = \{R^\eta\}_{\eta\in\mathsf{param}}$ where $R^\eta : \Omega^\eta \to \mathsf{coins} \cup \{\bot\}$.

**Messages:** Let the set of messages be $\mathcal{M}$ elements of the form $M = \{M^\eta\}_{\eta\in\mathsf{param}}$, where $M^\eta : \Omega^\eta \to \{0,1\}^{n^\eta} \cup \{\bot\}$. 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 $\eta$. 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

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

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 $X$. If the value of any of the input distributions is $\bot$ then we take the output to be $\bot$ as well. 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 hold iff there is an element $X \in T(D_M)$ such that $X_1 \sqsubseteq^{T(D_M)} X$ and $X_2 =^{\mathrm{C}} X$. Let for $P \in D_P$, let the notions analogous to $\sqsubseteq_P$ ($\sqsubseteq_{\neg P}$ respectively) be denoted by $\sqsubseteq_P^{T(D_M)}$ and $\sqsubseteq_P^{\mathrm{C}}$ ($\sqsubseteq_{\neg P}^{T(D_M)}$ and $\sqsubseteq_{\neg P}^{\mathrm{C}}$ respectively).

For any set of subsets $\mathfrak{D}^\eta \in \mathcal{F}^\eta$, $\mathfrak{D} = \{\mathfrak{D}^\eta\}_{\eta\in\eta}$ with non-negligible $p^\eta(\mathfrak{D}^\eta)$, we say that for $P \in D_P$, $X_1, X_2 \in D_M$, $X_1 = X_2$ on $\mathfrak{D}$ if there are $M_1 = \{M_1^\eta\}_{\eta\in\mathsf{param}} \in X_1$ and $M_2 = \{M_2^\eta\}_{\eta\in\mathsf{param}} \in X_2$ with $M_1^\eta(\omega) = M_2^\eta(\omega)$ for all $\omega \in \mathfrak{D}^\eta$. We say that $X_1 \sqsubseteq^{\mathrm{C}} X_2$ (or, $X_1 \sqsubseteq^{\mathrm{C}}_{(\neg)P} X_2$) on $\mathfrak{D}$ iff there is an element $X \in T(D_M)$ such that $X_1 \sqsubseteq^{T(D_M)} X$ (or, $X_1 \sqsubseteq^{T(D_M)}_{(\neg)P} X$) and $X_2 =^{\mathrm{C}} X$ on $\mathfrak{D}$.

**Execution trace:** The execution trace is defined as $Tr = \{Tr^\eta\}_{\eta\in\mathsf{param}}$ where $Tr^\eta : \omega \mapsto Tr^\eta(\omega)$ with either

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

where for each $\eta$ security parameter, $\omega \in \Omega^\eta$, $n^\eta(\omega)$ is a natural number less than $n^\eta$, $P_i^\eta(\omega) \in D_P$, $acts_i^\eta(\omega)$ is one of $generates$, $sends$, $receives$ and $s_i^\eta(\omega) \in \{0,1\}^*$; or $Tr^\eta(\omega) = \bot$ with $n^\eta(\omega) = 0$ meaning that no generate, send or receive action happened. For each $\eta$, $\omega$, and $i \in \{1, ..., n^\eta\}$, let

$$Tr_i^\eta(\omega) = \begin{cases} P_i^\eta(\omega)\ acts_i^\eta(\omega)\ s_i^\eta(\omega) & \text{if } i \in \{1, ..., n^\eta(\omega)\} \\ \bot \quad \text{otherwise} \end{cases}$$

We also require that for each $i$ there is a stopping time $J_i$ with $J_j(\omega) < J_{j+1}(\omega)$ for all $\omega$ and $j$ such that $Tr_i^\eta$ is measurable with respect to $\mathcal{F}_{J_i}^\eta$ for all $i$. Moreover, we require that any of $Tr$ is PPT computable from the earlier ones.

### 3.4 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 $r$ 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 (this can be achieved by tagging; in any case, we will call this *tagging condition*), and that honestly generated nonces have some fixed distribution over a set of bit strings with fixed length such that their collision probability is negligible in $\eta$. The network is completely controlled by an 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. We also assume that at one time only one action happens.

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

$$\mathfrak{M} = (\mathit{\Pi}, \mathfrak{E}, [\cdot,\cdot], \mathcal{N}_0, \mathbf{Pr}, \mathcal{P}, Tr, \Phi_{\mathcal{C}}, \mathfrak{D}),$$

where $\mathfrak{D} = \{\mathfrak{D}^\eta\}_{\eta\in\eta}$, $\mathfrak{D}^\eta \in \mathcal{F}^\eta$ a sequence of subsets where we focus our attention with $p^\eta(\mathfrak{D}^\eta)$ non-negligible; $\mathcal{N}_0 = \{\mathcal{N}_0^\eta\}_\eta$ is the distribution of correctly generated nonces; $\Phi_{\mathcal{C}}$ is a one-to-one function on $\mathcal{C}_{\mathtt{name}} \cup \mathcal{C}_{\mathtt{nonce}} \cup \mathcal{C}_{\mathtt{coin}}$ such that **(i)** $\Phi_{\mathcal{C}}(A) \in D_P$ 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; **(ii)** $\Phi_\mathcal{C}(N) \in D_N$ for any $N \in \mathcal{C}_{\texttt{nonce}}$ and for different constants the interpretations are not equal on $\mathfrak{D}$, and we further require that over $\{0,1\}^{m^\eta}$, $\Phi_\mathcal{C}(N)^\eta$ has the distribution (up to negligible probability) fixed for nonces (i.e. $\mathcal{N}_0^\eta$, e.g. uniform), and $\Phi_\mathcal{C}(N)^\eta$ is *independent* of $\mathcal{F}^\eta_{J_{\Phi_\mathcal{C}(N)}-1}$ for all $\eta$ on the condition that $[\Phi_\mathcal{C}(N)^\eta \neq \bot]$. (More precisely, there is an $N' \in \Phi_\mathcal{C}(N)$ such that $N'^\eta$ is independent of $\mathcal{F}^\eta_{J_{N'}-1}$ on the condition that $[N'^\eta \neq \bot]$. For $J^\eta_f$ see the paragraph before Example 3); **(iii)** $\Phi_\mathcal{C}(r) \in \mathcal{R}$ for any $r \in \mathcal{C}_{\texttt{coin}}$, and for different constants the interpretations are not equal on $\mathfrak{D}$, and we further require that over coins, $\Phi(r)^\eta$ has the distribution fixed for coins (e.g. uniform), and $\Phi_\mathcal{C}(r)^\eta$ is *independent* of $\mathcal{F}^\eta_{J_{\Phi_\mathcal{C}(r)}-1}$ for all $\eta$ on the condition that $[\Phi_\mathcal{C}(r)^\eta \neq \bot]$.

An *extension* of $\Phi_\mathcal{C}$ to evaluation of free variables is a function $\Phi$ that is the same on constants as $\Phi_\mathcal{C}$, and for variables $Q, n, m, s$ of sort `name`, `nonce`, `message`, and `coin` respectively, $\Phi(Q) \in D_P$, $\Phi(n) \in D_N$, $\Phi(m) \in D_M$, and $\Phi(s) \in \mathcal{R}$ hold. Let $\bar{\Phi}$ be defined to be the same as $\Phi$ on constants and variables, and let's extend the definition of $\bar{\Phi}$ to any term such that it takes values in $T(D_M)$ by the rules **(i)** $\bar{\Phi}(\langle t_1, t_2\rangle) = \langle \bar{\Phi}(t_1), \bar{\Phi}(t_2)\rangle$ ; **(ii)** $\bar{\Phi}(\{t\}^r_P) = \{\bar{\Phi}(t)\}^{\Phi(r)}_{\Phi(P)}$. Finally, for any $t$ term, let $\Phi(t) \in D_M$ be defined by $\Phi(t) =^\mathrm{C} \bar{\Phi}(t)$.

We say that an ensemble of random variables $M = \{M^\eta\}_{\eta \in \mathsf{param}}$ such that $M^\eta$ is defined on $\mathfrak{D}^\eta$ is a *realization* of the term $t$ through $\Phi$ on $\mathfrak{D}$, which we denote $M \lll_{\Phi,\mathfrak{D}} t$, if there is an $M_1 \in \Phi(t)$ with $M^\eta_1(\omega) = M^\eta(\omega) \neq \bot$ for all $\omega \in \mathfrak{D}^\eta$; and if also $t = \{t'\}^r_P$, $r$ being a constant, then we further require that there is an $M' \in \Phi(t')$ such that $M' \lll_{\Phi,\mathfrak{D}} t'$ and $\Phi(M')$ is measurable with respect to $\mathcal{F}^\eta_{J_r-1}$.

In the following, we give the interpretation of BPL. Note, that the interpretation of conjunction, disjunction, negation and conclusion are defined in the most standard manner. We first define when a formula $\varphi$ is *satisfied* by $\Phi$ (remember, $\mathfrak{D} = \{\mathfrak{D}^\eta\}_{\eta\in\eta}$ with $\mathfrak{D}^\eta \in \mathcal{F}^\eta$ from $\mathfrak{M}$):

- For any terms $t_1, t_2$, $\varphi \equiv t_1 = t_2$ is satisfied by $\Phi$, iff $\Phi(t_1) = \Phi(t_2)$ on $\mathfrak{D}$, $\varphi \equiv t_1 \sqsubseteq t_2$ is satisfied by $\Phi$ iff $\Phi(t_1) \sqsubseteq^\mathrm{C} \Phi(t_2)$ on $\mathfrak{D}$, $\varphi \equiv t_1 \sqsubseteq_{(\neg)P} t_2$ is satisfied by $\Phi$ on $\mathfrak{D}$ iff $\Phi(t_1) \sqsubseteq^\mathrm{C}_{(\neg)\Phi(P)} \Phi(t_2)$ on $\mathfrak{D}$.
- For any terms $t_1, t_2, t_3$, $|\overline{t_1 \sqsubseteq t_2 \sqsubseteq t_3}|$ is satisfied by $\Phi$, iff there are $X_1, X_2, X_3 \in T(D_M)$ such that $\Phi(t_1) =^\mathrm{C} X_1$, $\Phi(t_2) =^\mathrm{C} X_2$ and $\Phi(t_3) =^\mathrm{C} X_3$ on $\mathfrak{D}$, such that the bottom of the parsing tree of $X_i$ agrees with the parsing tree of $t_i$, and the interpretation of constants and variables in $t_i$ is given on $\mathfrak{D}$ by the sub-trees rooted in the corresponding positions in $X_i$, and further that also $|\overline{X_1 \sqsubseteq^{T(D_M)} X_2 \sqsubseteq^{T(D_M)} X_3}|$ holds where this is defined the same way on $T(D_M)$ as we defined it on $\bar{\mathcal{A}}$, that is, $X_1$ occurs in $X_3$ only within $X_2$.
- For any term $u$ and $acts = sends/receives$, $\varphi \equiv P\ acts\ u$ is satisfied by $\Phi$ iff there is a polynomially decidable stopping time $J^\eta$ such that apart from sets of negligible probability, $Tr^\eta_{J^\eta(\omega)}(\omega)$ is of the form $A^\eta\ acts\ M^\eta(\omega)$ for $\omega \in \mathfrak{D}^\eta$ where $M := \{M^\eta\}_{\eta\in\mathsf{param}} \lll_{\Phi,\mathfrak{D}} u$ and $A := \{A^\eta\}_{\eta\in\mathsf{param}} \lll_{\Phi,\mathfrak{D}} P$. We also require that the interpretation of every constant and variable in $u$ be measurable with respect to $\mathcal{F}^\eta_J$. We will denote this as $Tr_J \lll_{\Phi,\mathfrak{D}} P\ acts\ u$.
- If $acts = generates$ then the $u$ above is a nonce $\nu$, and so $M := \{M^\eta\}_{\eta\in\mathsf{param}} \lll_{\Phi,\mathfrak{D}} u$ means there is an $N \in \Phi(\nu)$ such that $M^\eta|_{\mathfrak{D}^\eta} \approx N^\eta|_{\mathfrak{D}^\eta}$ in this case, and we further require that over $\{0,1\}^{m^\eta}$, $N^\eta$ has the distribution fixed for nonces (i.e. $\mathcal{N}^\eta_0$), and $N^\eta$ is *independent* of $\mathcal{F}^\eta_{J-1}$ for all $\eta$ on the condition that $[N \neq \bot]$.
- $\varphi \equiv \beta_1, ..., \beta_n$ sequence of actions is satisfied by $\Phi$ if each of $\beta_k$ $(k = 1, ..., n)$ is satisfied by $\Phi$, and if $J_k$ is the stopping time belonging to $\beta_k$, then we require that $J_k < J_l$ on $\mathfrak{D}$ whenever $k < l$ (that is, for each $\eta \in$ param and $\omega \in \mathfrak{D}^\eta$, $J^\eta_k(\omega) < J^\eta_l(\omega)$.

– For any formulas $\varphi$, $\varphi_1$, $\varphi_2$, $\neg\varphi$ is satisfied by $\Phi$ iff $\varphi$ is not satisfied by $\Phi$; $\varphi_1 \vee \varphi_2$ is satisfied by $\Phi$ iff $\varphi_1$ is satisfied by $\Phi$ or $\varphi_2$ is satisfied by $\Phi$; $\varphi_1 \wedge \varphi_2$ is satisfied by $\Phi$ iff $\varphi_1$ is satisfied by $\Phi$ and $\varphi_2$ is also satisfied by $\Phi$. $\varphi_1 \rightarrow \varphi_2$ is satisfied by $\Phi$ iff $\neg\varphi_1 \vee \varphi_2$ is satisfied by $\Phi$.
– If $\varphi$ is a formula, $m$ a variable, then $\forall m\varphi$ (or $\exists m\varphi$, resp.) is satisfied by $\Phi$ iff $\varphi$ is satisfied by each (or some, resp.) $\Phi'$ extension of $\Phi_{\mathcal{C}}$ when $\Phi'$ only differs from $\Phi$ on $m$.

A formula $\varphi$ is *true* in the structure $\mathfrak{M}$, iff $\varphi$ is satisfied by every $\Phi$ 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.

**Truth of the Term axioms.**

(a) These axioms are true since if terms are equal in the free algebra $\bar{\mathcal{A}}$, then their interpretations are also equal, no matter how $\Phi$ is extended to variables. Further, if $t \sqsubseteq t'$ holds in the free algebra, then the way we receive $t'$ from $t$ by pairing and encryptions carries over to the computational world, no matter how $\Phi$ is evaluated on variables. Same is true for $\sqsubseteq_P$. As for $\sqsubseteq_{\neg Q}$, it is made sure that in the free algebra $t$ can be received from $t'$ without encrypting with the key of the substitution for $Q$ as long as it is not equal the $P$'s. Since the explicit inclusion in the free algebra carries over to the interpretation, the formula must be satisfied.
(b) These axioms hold as computational equality is also symmetric, reflexive and transitive. Further, subterm relation is also transitive for the interpretations.
(c) Almost the same as (b). Equality implies computation subterm relation by definition of computational subterm. Subterm reachable using only decryption with respect to the private key of a specific principal (or other than a specific principal) is also clearly a general subterm.
(d) If the interpretations of $\{t_1\}^s_Q$ and $\{t_2\}^{s'}_Q$ are computationally equal up to negligible probability, then the interpretations of $t_1$ and $t_2$ must also be equal up to negligible probability as the decryptions of both sides with the decryption key of $Q$ give the interpretations of the encrypted terms: $\Phi(t_1) = \mathcal{D}(d_{\Phi(Q)}, \Phi(\{t_1\}^s_Q))$ and $\Phi(t_2) = \mathcal{D}(d_{\Phi(Q)}, \Phi(\{t_2\}^{s'}_Q))$ and the right-hand sides are equal up to negligible probability.
(e) Soundness of this axiom follows as we supposed that computational pairing is one-to-one.
(f) These follow from the tagging condition as tagging ensures that encryption is never confused with pairs, nonces, names.
(g) Follows from tagging.
(h) Soundness of the first formula follows as if $t \sqsubseteq \langle t_1, t_2\rangle$ is satisfied, then either the interpretations of the two sides are equal (up to negligible probability) and hence $t = \langle t_1, t_2\rangle$ is satisfied, or (by definition of satisfaction of $t \sqsubseteq \langle t_1, t_2\rangle$) the interpretation of $\langle t_1, t_2\rangle$ can be received from the interpretation of $t$ via encryptions and pairing, of which the last has to be pairing because the tags have to match; then by soundness of (e), it follows that the paired items must in fact be interpretations of $t_1$ and $t_2$, which implies that either of the interpretations of $t_1$ or of $t_2$ was received from the interpretation of $t$ via pairing and encryptions, which means that either $t \sqsubseteq t_1$ or $t \sqsubseteq t_2$ is satisfied, and that proofs the soundness of this formula. As for the second formula, if $t_1 \sqsubseteq \{t_2\}^s_Q$ is satisfied, then either the interpretations of the two sides are equal, or the interpretation of $\{t_2\}^s_Q$ can be received from the interpretation of $t_1$ via encryptions and pairing, of which the last has to be encryption because the tags have to match, and so soundness follows.
(i) Proof for the first and second formulas are the same as in (h), For the third, if the premise holds, then the interpretation of $\{t_2\}^s_P$ can be received from the interpretation of $t_1$ with pairing with

others and encrypting only with the encryption key of $P$. Therefore there is some $t$ and $s'$ such that $t_1 \sqsubseteq_P t \wedge \{t\}_P^{s'} = \{t_2\}_P^s$ is satisfied. But then from (d) it follows that $t = t_2$ and so $t_1 \sqsubseteq_P t_2$.

(j) Similar to (h) and (i).

(k) Also follows from tagging.

(l) This follows trivially from the interpretation and that we assumed that the interpretations of constants are distinguishable. So if $|\overline{t_1 \sqsubseteq^{\bar{A}} t_2 \sqsubseteq^{\bar{A}} t_3}|[\boldsymbol{M}/\boldsymbol{m}]$ holds for all $\boldsymbol{M}$ vector of constans of appropriate sort, then with $X_1 = \bar{\Phi}(t_1)$, $X_2 = \bar{\Phi}(t_2)$ and $X_3 = \bar{\Phi}(t_3)$, the relation $|\overline{X_1 \sqsubseteq^{T(D_M)} X_2 \sqsubseteq^{T(D_M)} X_3}|$ must also hold as the parsing tree of $\bar{\Phi}(t_i)$ is the same as that of $t_i$, and further, for differing constants in $t_i$ the interpretations are assumed to differ, and although the interpretations of differing variables may coincide, the relation in $T(D_M)$ must hold as we assumed that $|\overline{t_1 \sqsubseteq^{\bar{A}} t_2 \sqsubseteq^{\bar{A}} t_3}|[\boldsymbol{M}/\boldsymbol{m}]$ holds for all $\boldsymbol{M}$ vector of constans of appropriate sort (hence also replacement by identical constants).

**Truth of the Ordering axiom.** Suppose that there is an extension $\Phi$ and a domain $\mathfrak{D}$ such that the formula $Q_2\ sends\ m; Q_1\ generates\ n$ is satisfied on $\mathfrak{D}$ with non-negligible probability as well as the formula $n \sqsubseteq m$. Then, there are stopping times $J_1^\eta$, $J_2^\eta$ such that $Tr_{J_1} \lll_{\Phi,\mathfrak{D}} Q\ sends\ m$, and $Tr_{J_2} \lll_{\Phi,\mathfrak{D}} Q\ generates\ n$, and $J_1 < J_2$. Then, $Tr_{J_1} \lll_{\Phi,\mathfrak{D}} Q\ sends\ m$ implies that there is $M \lll_{\Phi,\mathfrak{D}} m$ such that $M^\eta$ 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, $N^\eta$ only depends on the randomness until $J_1$, so $N^\eta$ is measurable with respect to $\mathcal{F}^\eta_{J_1}$. But, $Tr_{J_2} \lll_{\Phi,\mathfrak{D}} Q\ generates\ n$ implies that $\Phi(n)$ has an element $N'^\eta$ measurable with respect to $\mathcal{F}^\eta_{J_2}$ and independent of $\mathcal{F}^\eta_{J_2-1}$ on $[N'^\eta \neq \bot]$, and hence independent of $\mathcal{F}^\eta_{J_1}$ and of $N^\eta$ on $[N'^\eta \neq \bot]$. So, $N$ and $N'$ only differ up to negligible probability, but $N^\eta$ and $N'^\eta$ are independent for all $\eta$, which is possible only if $\Pr[N'^\eta \neq \bot]$ is negligible. That means $\mathfrak{D}$ has negligible probability, a contradiction.

**Truth of the Nonce verification axioms.** In order to show that the axioms are satisfied, we use the assumption that regular participants (the ones represented by constants) encrypt with a CCA-2 secure encryption scheme. For the first nonce-verification axiom, suppose there is a $\Phi$ and non-negligible $\mathfrak{D}$ such that the premise of the axioms are satisfied by $\Phi$ on $\mathfrak{D}$, but the conclusion is not. Then, if the conclusion is not satisfied, that means that either $A$ never sends the nonce out (which clearly cannot happen with non-negligible probability as later $Q$ receives it and the probability of collision of nonces is negligible), or $\{m_6\}_B^r$ does not go through $B$ between $A$ sending it and $Q$ receiving it with non-negligible probability. The premise however says that $n_1$ shows up in $m_5$ later in another form, and it can be recovered from there up to negligible probability via a series of de-coupling and decryption such that the key of $B$ does not have to be used. We have to show that a PPT algorithm can be constructed that breaks CCA-2 security.

First observe, that the satisfaction of $\forall m_7(A\ sends\ m_7 \wedge n_1 \sqsubseteq m_7 \rightarrow |\overline{n_1 \sqsubseteq \{m_6\}_B^r \sqsubseteq m_7}|)$ means that $B$ indeed sent $n_1$ out only in the form of $\{m_6\}_B^r$. The reason is that if, according to the satisfaction of this formula, there are $X_1, X_2, X_3 \in T(D_M)$ such that $\Phi(n_1) =^{\mathrm{C}} X_1$, $\Phi(\{m_6\}_B^r) =^{\mathrm{C}} X_2$ and $\Phi(m_7) =^{\mathrm{C}} X_3$ on $\mathfrak{D}$ so that $X_1$ occurs in $X_3$, but only within $X_2$, then there cannot be any way to create $X_3$ at the point when $A$ sends it first out other than from $\Phi(\{m_6\}_B^r)$, because otherwise without the decryption of $\Phi(\{m_6\}_B^r)$ we could access $n_1$ contradicting the CCA-2 security of $\Phi(\{m_6\}_B^r)$. Why is this encryption CCA-2? Because at the first time when $n_1$ is sent out, $r$ had to be created by $A$, and hence never revealed to anyone. The fact that we assume in the interpretation of $\{m_6\}_B^r$ that $r$ has to be independent of what happened earlier and that $m_6$ has to be part of the earlier history, ensures that this term is not confused by any other encryption that was sent out by $A$.

The algorithm that breaks CCA-2 security is simply the protocol execution itself with the following modifications:

1. The encryption-key decryption-key pair of $B$ is generated by the challenger in the CCA-2 game. The encryption key is accessible to the algorithm, that is, the protocol execution uses it every time encryption with the public key of $B$ is necessary.
2. Since the algorithm cannot use the decryption-key of $B$ known only to the challenger, the decryption oracle (that the algorithm may access according to the definition of CCA-2 security) does all decryptions that occur in the protocol execution using the private key $d_B$, except for the decryption of the interpretation of $\{m_6\}^r_B$.
3. The algorithm generates two samples of $n_1$ when (indicated by the stopping time from the satisfaction of the formula; notice, that we need to know in polynomial time where it is) the protocol execution samples $n_1$.
4. From this on, run the protocol parallel with the two values of $n_1$. Hence, when (again, given by the corresponding stopping time) the protocol execution is to produce $m_6$, 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 $m_6$.
5. Submit to the encryption oracle of the CCA-2 game the pair of samples of $m_6$, and use the cipher that it outputs in both of the parallel processes whenever $\{m_6\}^r_B$ occurs again (that is, if the protocol execution is defined to use it).
6. If one of the parallel processes happens to crash as that process is running with the wrong encryption, then output the value of $b$ that corresponds the other process.
7. If the sample for $\{m_6\}^r_B$ goes through $B$ in either of the parallel processes, terminate, and output a random guess of $b$. If not, continue until the $Q$ receives the sample for $m_5$ (that is, until the value of the stopping time indicating the point of reception of $m_5$).
8. Recover the sample for $n_1$ via de-coupling and decryption. Since $m_5$ contains $m_1$ not encrypted by the key of $B$, it can be recovered. The bit string hence obtained is the one (out of the two generated for $n_1$) that was in the plaintext encrypted by the oracle, so the bit value $b$ of the game can be determined.
9. If any one of 3., 4., or 6. does not happen on an execution trace then proceed to the end and output a random guess of $b$.
10. If the premise of the axiom is satisfied, then, of course $n_1$ had to be sent out by $A$ in some message $m_2$ before $Q$ received it, otherwise whoever $Q$ received it from could only guess $n_1$, but guessing it correctly has only negligible probability as we assumed.
11. If the premise of the axiom is satisfied but rest of the conclusion is not satisfied, that is, $\{m_6\}^r_B$ does not go through $B$, but $n_1$ turns up unencrypted by the key of $B$, then, since this algorithm determines the value of $b$ in all these cases, the algorithm has probability non-negligibly different from $1/2$ of winning the CCA-2 game to break $\{m_6\}^r_B$ as $\mathfrak{D}$ is non-negligible.

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:

1. The encryption-key decryption-key pairs of $B$ and $C$ are generated by the challenger in the CCA-2 game. The encryption keys are accessible to the algorithm.
2. The decryption oracles (that the algorithm may access according to the modified definition of CCA-2 security) do all decryptions with the private keys $d_B$ and $d_C$.
3. The algorithm generates two samples of $n_1$ when the protocol execution samples $n_1$.

4. From this on, run the protocol parallel with the two values of $n_1$. Hence, when the protocol execution is to produce $\{m_6\}_B^{r_1}$, 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 $m_6$.
5. Submit to the first encryption oracle of the CCA-2 game the pairs of samples of $m_6$, and use the cipher that it outputs in both of the parallel processes whenever $\{m_6\}_B^{r_1}$ occurs again (that is, if the protocol execution is defined to use it).
6. Skip the step when $B$ decrypts $\{m_6\}_B^{r_1}$.
7. When $\{m_8\}_C^{r_2}$ 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.
8. If one of the parallel processes happens to crash as that process is running with the wrong encryption, then output the value of $b$ that corresponds the other process.
9. Continue until $C$ receives the sample for $m_5$.
10. Recover the sample for $n_1$ via de-coupling and decryption. 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.
11. If any one of 2., 3., 6., or 7. does not happen on an execution trace then proceed to the end and output a random guess of $b$. This is again PPT algorithm given that the protocol execution was PPT, so it breaks CCA-2 security.

**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 $\mathfrak{M} = (\Pi, \mathfrak{E}, [\cdot,\cdot], \mathcal{N}_0, \mathbf{Pr}, \mathcal{P}, Tr, \Phi_{\mathcal{C}}, \mathfrak{D})$, then, if a formula (the query form in particular) is provable in the syntax with first-order predicate logic and axioms (I), (II), (III), then it is true in $\mathfrak{M}$.*

*Proof.* We have showed that the term axioms and non-logical axioms of BPL are true in the model. It is routine to check that all the logical axioms and logical inference rules of first order logic are also true in the model, because we followed the usual first-order logical operations of composed formulas in the interpretation. Hence the theorem holds.

### 3.5 Our semantics and Computational PCL

We would like to point out some aspects where problems arise in case of the treatment of Datta et al. and how they are related to our treatment. We emphasize that we do not claim that these issues are impossible to be fixed in their framework, we only indicate what our answers are to them.

1. Their treatment is non-deterministic, that is, they rely on counting equiprobable traces. Unequal probabilities may be dealt with by counting a trace more then once (although a priori it is not quite clear whether this will lead to problems), but their method certainly only applies to executions when the number of possible computational traces for a given security parameter is finite. Since some formulations of probabilistic polynomial time processes are not limited to finitely many traces (only the *expected termination time* must be polynomial), it is better not to exclude infinite number of traces. Our method works for infinite number of traces and arbitrary probability distributions. Removing the bound $n^\eta$ from the length of executions is not a difficult 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. It is perhaps worthwhile to note that although manipulations with expected polynomial time algorithms may

lead out of their realm, the proof of the nonce-verification axioms only involve minor modifications (no compositions of two expected PT algorithms) of the expected polynomial-time protocol execution that should not lead out of the realm.

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. 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 $\sigma$ evaluation of variables by bit-strings, $\sigma(b)\neq\sigma(\mathrm{Send}(\hat{X},t))$. Let us now not be bothered by the problem that they define a syntactic axiom using the semantics. 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 $\sigma$ 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. 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. 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. A simple example of such a situation is when on two traces, which coincide up to step $t_0$, say, $\mathrm{Send}(\hat{X},t)$ is chosen on one trace for $t_1<t_0$, but on the other trace it is chosen somewhere else. Since the two traces coincide at step $t_1$, if this time is picked on one trace, it must be picked on the other trace too. 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*, according to which random variables have to be measurable, dependence on the future is taken care of by measurability.

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 (which though does not appear in their computational PCL papers) 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 $N$) in Section 4.7 of[14]. This problem persists even if such a coincidence cannot be efficiently computed. In our method, 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 this anomaly 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. As the idea of making use of the notions from the theory of stochastic processes in the definition of satisfaction of formulas does not require the special properties of BPL, we believe that it should not be difficult to adopt this method to a wide range of formal models such as PCL or strand space models.

## 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 FoSSaCS'06*, Lecture Notes in Computer Science. Springer-Verlag, March-April 2006.
3. M. Abadi and P. Rogaway. Reconciling two views of cryptography. *Journal of Cryptology*, 15(2):103–127, January 2002.
4. P. Adão, G. Bana, J. Herzog, and A. Scedrov. Soundness of formal encryption in the presence of key-cycles. In *Proceedings of ESORICS'05*, volume 3679 of *LNCS*, 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 CSFW'05*, pages 170–184, Aix-en-Provence, France, June 20–22 2005. IEEE Computer Society Press.
6. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proceedings of CCS'03*, pages 220–230. ACM Press, 2003.
7. G. Bana, K. Hasebe, and M. Okada. Computational semantics for basic protocol logic - a stochastic approach. In *Proceedings of 12th Asian Computing Science Conference*, volume 4846 of *LNCS*, pages 86–94, Doha, Qatar, November 2007. Springer.
8. M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Proceedings of ICALP'05*, volume 3580 of *Lecture Notes in Computer Science*, pages 652–663. Springer-Verlag, July 2005.
9. 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.
10. 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.
11. I. Cervesato, C. Meadows, and D. Pavlovic. An encapsulated authentication logic for reasoning about key distribution protocols. In *Proceedings of CSFW'05*, pages 48–61, 2005.
12. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13:423–482, 2005.
13. A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In *Proceedings of ICALP'05*, volume 3580 of *LNCS*, pages 16–29. Springer, 2005.
14. Anupam Datta, Ante Derek, John C. Mitchell, and Arnab Roy. Protocol composition logic (pcl). *Electron. Notes Theor. Comput. Sci.*, 172:311–358, 2007.

15. 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.
16. 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.
17. 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.
18. J. D. Guttman and F. J. Thayer Fábrega. Authentication tests. In *IEEE Symposium on Security and Privacy*, pages 96–109, 2002.
19. J. D. Guttman, F. J. Thayer, and L. D. Zuck. The faithfulness of abstract protocol analysis: Message authentication. In *Proceedings of CCS'01*, pages 186–195. ACM Press, November 05–08 2001.
20. K. Hasebe and M. Okada. Completeness and counter-example generations of a basic protocol logic. In *Proceedings of RULE'05*, volume 147(1), pages 73–92. Elsevier Science, 2005. Available also at http://dx.doi.org/10.1016/j.entcs.2005.06.038.
21. P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proceedings of S&P'04*, pages 71–85. IEEE Computer Society Press, May 9–12 2004.
22. G. Lowe. A hierarchy of authentication specifications. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.
23. D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–130, 2004.
24. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.
25. T. Woo and S. Lam. Verifying authentication protocols: Methodology and example. In *Proceedings of the International Conference on Network Protocols*, 1993.

## A Definition of $Only$ and $Honest$

Our aimed correctness properties are described in a special form of formulas, called *query form*. Let $\boldsymbol{\alpha}^A[\boldsymbol{Q}, \boldsymbol{N}, \boldsymbol{n}, \boldsymbol{r}, \boldsymbol{s}]$ 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 nonces in $\boldsymbol{N} = \{N_1, ..., N_g\}$ and $\boldsymbol{n} = \{n_1, ..., n_h\}$ from coins in $\boldsymbol{r} = \{r_1, ..., r_i\}$ and $\boldsymbol{s} = \{s_1, ..., s_j\}$ and from names $A$ and $\boldsymbol{Q} = \{Q_1, ..., Q_l\}$. Let $\boldsymbol{\alpha}^A_{\leq i}$ denote an initial segment of $\boldsymbol{\alpha}^A$ ending with $A\, acts_i\, t_i$ (for $1 \leq i \leq k$), i.e., $\boldsymbol{\alpha}^A_{\leq i} \equiv A\, acts_1\, t_1; \cdots A\, acts_i\, t_i$. Let $\boldsymbol{\alpha}^A_{\leq 0} \equiv A = A$.

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:

$$Honest(\boldsymbol{\alpha}^A) \stackrel{\text{def}}{\equiv} \bigvee_{i \in \{0\} \cup \{j \mid \ acts_j = sends\ \} \cup \{k\}} \boldsymbol{\alpha}^A_{\leq i} \wedge Only(\boldsymbol{\alpha}^A_{\leq i})$$

$Only(\boldsymbol{\alpha}^A)$ denotes the following formula, whose intuitive meaning is "$A$ performs only $\boldsymbol{\alpha}^A$".

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

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.