Completeness and Counter-Example Generations of a Basic Protocol Logic (Extended Abstract)

Koji Hasebe [1,2] and Mitsuhiro Okada [1,3]

Department of Philosophy
Keio University
2-15-45, Mita, Minato-ku, Tokyo 108-8345, Japan

Abstract

We give an axiomatic system in first-order predicate logic with equality for proving security protocols correct. Our axioms and inference rules derive the basic inference rules, which are explicitly or implicitly used in the literature of protocol logics, hence we call our axiomatic system *Basic Protocol Logic* (or *BPL*, for short). We give a formal semantics for *BPL*, and show the completeness theorem such that for any given query (which represents a correctness property) the query is provable iff it is true for any model. Moreover, as a corollary of our completeness proof, the decidability of provability in *BPL* holds for any given query. In our formal semantics we consider a "trace" any kind of sequence of primitive actions, counter-models (which are generated from an unprovable query) cannot be immediately regarded as realizable traces (i.e., attacked processes on the protocol in question). However, with the aid of Comon-Treinen's algorithm for the intruder deduction problem, we can determine whether there exists a realizable trace among formal counter-models, if any, generated by the proof-search method (used in our completeness proof). We also demonstrate that our method is useful for both proof construction and flaw analysis by using a simple example.

Key words: security protocol analysis, first-order predicate logic, agreement properties, proof-search method.

[1] This work was partly supported by Grants-in-Aid for Scientific Research of MEXT, Center of Excellence of MEXT on Humanity Sciences (Keio University) and Oogata-kenkyu-jyosei grant (Keio University). The first author was also supported by Fellowship for Japan Young Scientists from Japan Society for the Promotion of Science.

[2] Email: hasebe@abelard.flet.keio.ac.jp

[3] Email: mitsu@abelard.flet.keio.ac.jp

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

1 Introduction

For the formal analysis of security protocols, there are two typical approaches among others: one emphasizing a syntactic method such as BAN-logic [2] and protocol logics of [9,7,16] (cf. also [1] for *a protocol composition logic project overview*), and the other emphasizing a semantic method such as the strand space method [18] and MSR [4]. The former approach aims at proving a property which guarantees a protocol correct in terms of a certain logical inference system, while the latter approach aims at detecting flaws in a protocol (i.e., concrete attacks on a protocol) in terms of a kind of trace-model. In this paper we take the former approach. That is, our main purpose in this paper is to give a simple formulation of a core part of the protocol logics of [9,7,16] for proving protocols correct. We also aim at connecting this formal approach to a method for the flaw detection. Especially, we concentrate on several types of agreement properties in the sense of [20,14], and investigate how much one can formulate a basic part of the protocol logics, which is enough to prove our aimed properties, within the first-order predicate logic. (Thus, in this paper we do not go into the secrecy property about nonces or session keys. Such a property is also an important matter for security analysis because agreement properties of some protocols depend on their secrecy properties.) Moreover, we also give a complete formal semantics of *BPL*, and present how to apply our framework for both proving correctness and detecting flaws of security protocols.

For that purpose, we first give an axiomatic system in first-order predicate logic for proving the agreement properties. In this system, we formalize some properties about nonces and cryptographic assumptions as non-logical axioms in first-order predicate logic with equality, and give a special form of formulas, called *query form*, which represents an agreement property restricted to the number of data items (i.e., nonces) in the protocol in question. Then the basic inference rules, which are explicitly or implicitly used for proving agreement properties in the protocol logics of [9,7,16], are derived rules of our system. Hence our formulation is called *Basic Protocol Logic* (or *BPL*, for short).

Next, we give a formal semantics for *BPL* and show the completeness theorem such that *for any given query, the query is provable in BPL iff it is true for any model*. This theorem is proved by adjusting the usual proof-search method for the first-order predicate logic into our framework (cf. [17]). As a direct corollary of the completeness theorem, our proof-search method of the completeness proof provides a counter-example generation if the given query is unprovable. Moreover, as a corollary of our completeness proof, the decidability of provability in *BPL* also holds for any query. (In this paper, we only sketch out the proofs of the completeness theorem and its corollaries. The detailed proofs will appear in the full version of this paper.) In our formal semantics we consider a "trace" any kind of sequence of primitive actions, thus a counter-example (generated from an unprovable query) cannot be immedi-

ately regarded as a realizable trace (i.e., an attacked process on the protocol in question). However, with the aid of Comon-Treinen's algorithm for the intruder deduction problem [6], we can determine whether there exists a realizable trace among formal counter-examples. Therefore, by the combination of our completeness proof and the Comon-Treinen's algorithm, for any given query we can generate attacked processes on the protocol, if any, whenever we set any upper-bound on the number of data items.

Since our proof construction procedure is directly a counter-example generation, this work would make a contribution to bridge the gap between the two different directions of security protocol analysis, namely proving correctness and finding attacks. Finally, we also demonstrate by a simple example that our method is useful for both proof constructions and flaw detections.

Organization of this paper. In Section 2 we introduce an axiomatic system, called *Basic Protocol Logic* in first-order predicate logic, and formalize our aimed correctness properties as a special form of formulas, called *query form*. In Section 3 we give a trace-based formal semantics and show the soundness theorem for the query form. In Section 4 we show the completeness theorem and the decidability of provability for the query form, and explain how to construct proofs/attacked processes for a given query by means of the proof-search method and Comon-Treinen's algorithm for the intruder deduction problem. Finally, in Section 5 we present the conclusions and outline some further directions of this research.

2 Basic Protocol Logic

We first fix our language in first-order predicate logic with equality, and give an axiomatic system, called *Basic Protocol Logic*. In this system, our aimed correctness properties are described as a special form of formulas, called *query form*.

2.1 Language

Sorts and terms. Our language is order-sorted, which consists of sorts `name`, `nonce` and `message`. $A, B, \ldots, A_1, A_2, \ldots$ ($P, Q, \ldots, P_1, P_2, \ldots$, resp.) are constants (variables, resp.) of sort `name` (which represent principal name), and $N, N', \ldots, N_1, N_2, \ldots$ ($n, n', \ldots, n_1, n_2, \ldots$, resp.) are constants (variables, resp.) of sort `nonce`. All terms of sorts `name` and `nonce` are terms of sort `message`. The symbols $m, m', \ldots, m_1, m_2, \ldots$ are used to denote variables of sort `message`. Compound terms of sort `message` are made by the functions $\langle m_1, \ldots, m_n \rangle$, $\{m\}_P$ and $\{m\}_{P^{-1}}$, which represent n-tuple concatenation of messages, encryption of public key and secret key for P, respectively. We also use the meta-symbols $s, s', \ldots, t, t', \ldots$ to denote any terms of sort `message`.[4]

[4] In order to make our discussion simpler, in this paper we do not consider symmetric cryptography nor protocols for sharing session keys, however our formalization can be easily

Formulas. We introduce five binary predicate symbols: P *generates* n, P *receives* m, P *sends* m, $m = m'$ and $m \sqsubseteq m'$, which represent "P generates a fresh value n as a nonce", "P receives a message m", "P sends a message m", and usual equality and subterm relation (i.e., "m is identical with m'" and "m is a subterm of m'"), 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*.

Atomic formulas are the following expressions: $P_1\ acts_1\ m_1; P_2\ acts_2\ m_2; \cdots; P_k\ acts_k\ m_k$ (where $k \geq 1$ and P_i (m_i, resp.) may be the same as P_j (m_j, resp.) for any $i \neq j$), $m = m'$ and $m \sqsubseteq m'$. The first one is called *trace formula*. This type of atomic formulas is used to represent a sequence of principal's actions: for example, the intuitive meaning of the atomic formula $P\ sends\ m; Q\ receives\ m'$ is "P sends a message m before Q receives a message m'". We also use the following symbols as meta expressions. $\alpha^P, \beta^P, \ldots, \alpha_1^P, \alpha_2^P, \ldots$ (or simply, $\alpha, \beta, \ldots, \alpha_1, \alpha_2, \ldots$) is used to denote a trace formula of the form $P\ acts\ m$, and $\alpha_1^{P_1}; \cdots; \alpha_k^{P_k}$ (or $\vec{\alpha}$, for short) is used to denote $P_1\ acts_1\ m_1; \cdots; P_k\ acts_k\ m_k$ (where k indicates the *length* of $\vec{\alpha}$). Especially, when every P_i is identical with P for $1 \leq i \leq k$ (i.e., a sequence of actions performed by a single principal P), we also use $\vec{\alpha}^P$ to denote such a trace formula. For $\vec{\alpha}$ ($\equiv \alpha_1; \cdots; \alpha_m$) and $\vec{\beta}$ ($\equiv \beta_1; \cdots; \beta_n$), we say $\vec{\beta}$ *includes* $\vec{\alpha}$ (denoted by $\vec{\alpha} \subseteq \vec{\beta}$), if $\vec{\alpha}$ and $\vec{\beta}$ satisfies the following condition: α_i appears in $\vec{\beta}$ for all i ($1 \leq i \leq m$), and for any $\alpha_i \equiv \beta_j$ and $\alpha_k \equiv \beta_l$ if $i < k$ then $j < l$ for any $1 \leq i, k \leq m$. (Roughly speaking, $\vec{\alpha} \subseteq \vec{\beta}$ means that all the action predicates in $\vec{\alpha}$ appear in $\vec{\beta}$ with preserving the order of $\vec{\alpha}$.)

The formulas (denoted by $\varphi, \psi, \ldots$) are made by the following grammar.

$$\varphi \ ::= \ \vec{\alpha} \mid m = m' \mid m \sqsubseteq m' \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \forall x\varphi \mid \exists x\varphi$$

We use the meta expression $\varphi[\vec{m}]$ to indicate the list of terms $\vec{m}$ occurring in φ. Substitutions are represented in terms of this notation.

Finally, we introduce the notion of (*strict*) *order-preserving merge of trace formulas* $\vec{\alpha}$ and $\vec{\beta}$, which is defined as follows.

Definition 2.1 (Order-preserving merge) An order-preserving merge of $\vec{\alpha}$ ($\equiv \alpha_1; \cdots; \alpha_l$) and $\vec{\beta}$ ($\equiv \beta_1; \cdots; \beta_m$) is a trace formula $\vec{\delta}$ ($\equiv \delta_1; \cdots; \delta_n$) which is made by the following rules. (As a special case, if $\vec{\alpha}$ and $\vec{\beta}$ is empty then we respectively consider $l = 0$ and $n = 0$.)

(i) $\delta_1 \equiv \alpha_1$ or β_1.

(ii) For each i ($1 \leq i \leq n$), if $\alpha_1; \cdots; \alpha_j \subseteq \delta_1; \cdots; \delta_i$ and $\beta_1; \cdots; \beta_k \subseteq \delta_1; \cdots; \delta_i$ (for some $0 \leq j \leq l$ and $0 \leq k \leq m$), then $\delta_{i+1} \equiv \alpha_{j+1}$ or β_{k+1}.

(iii) If $\vec{\delta} \equiv \delta_1; \cdots; \delta_i; \delta_{i+1}; \cdots; \delta_n$ is an order-preserving merge of $\vec{\alpha}$ and $\vec{\beta}$ with $\delta_i \equiv \delta_{i+1}$, then $\vec{\delta}' \equiv \delta_1; \cdots; \delta_{i-1}; \delta_{i+1}; \cdots; \delta_n$ is also an order-preserving

extended so as to include such notions.

merge of $\vec{\alpha}$ and $\vec{\beta}$.

We also introduce another type of order-preserving merge which is made only by the rules (i) and (ii), and call it *strict order-preserving merge of* $\vec{\alpha}$ and $\vec{\beta}$. For example, both $\alpha_1;\alpha_2;\alpha_2;\alpha_3$ and $\alpha_2;\alpha_1;\alpha_3;\alpha_2$ and $\alpha_1;\alpha_2;\alpha_3$ are order-preserving merges of $\alpha_1;\alpha_2$ and $\alpha_2;\alpha_3$, while the last one is not a strict order-preserving merge.

Description of roles

A *protocol* is a set of *roles*, and each role for a principal (say, P) is described as a trace formula of the form $\vec{\alpha}^P \equiv P\ acts_1\ m_1;\cdots;P\ acts_k\ m_k$.

As an example, here we consider the Needham-Schroeder public key protocol [15], whose informal description is as follows.

1. $P \to Q$: $\{n_1, P\}_Q$
2. $Q \to P$: $\{n_1, n_2\}_P$
3. $P \to Q$: $\{n_2\}_Q$

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

Example 2.2 (Roles of the Needham-Schroeder protocol)

$$Init_{NS}[P,Q,n_1,n_2] \equiv$$
$$P\ generates\ n_1; P\ sends\ \{n_1,P\}_Q; P\ receives\ \{n_1,n_2\}_P; P\ sends\ \{n_2\}_Q$$

$$Resp_{NS}[P,Q,n_1,n_2] \equiv$$
$$Q\ receives\ \{n_1,P\}_Q; Q\ generates\ n_2; Q\ sends\ \{n_1,n_2\}_P; Q\ receives\ \{n_2\}_Q$$

A *run* of a role $\vec{\alpha}^P$ is a formula obtained by substituting P, $\vec{Q}$ and $\vec{n}$ with some terms of the same sorts. For example, $Init_{NS}[A/P,B/Q,N_1/n_1,N_2/n_2]$ and $Resp_{NS}[A/P,B/Q,N_1/n_1,N_2/n_2]$ are runs of $Init_{NS}$ and $Resp_{NS}$, respectively, and we call such a set of instances A,B,N_1,N_2 *data items*. A strict order-preserving merge of runs of a role $\vec{\alpha}^P$ is called *multiple runs* of $\vec{\alpha}^P$.

2.2 Basic Protocol Logic

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

(I) Axioms of universal sentences over terms. We presume the following axioms for $=$ and $\sqsubseteq$. When a finite set of literals $\{t_1 = t_1', \ldots, t_n = t_n', s_1 \sqsubseteq s_1', \ldots s_j \sqsubseteq s_j', u_1 \neq u_1', \ldots, u_k \neq u_k', v_1 \not\sqsubseteq v_1', \ldots, v_l \not\sqsubseteq v_l'\}$ is unsatisfiable in the free term algebra of our language (where $=$ and $\sqsubseteq$ are the identity terms and subterm relation in the free term algebra), then $\forall\vec{m}\neg(t_1 = t_1' \wedge \cdots \wedge t_n = t_n' \wedge s_1 \sqsubseteq s_1' \wedge \cdots \wedge s_j \sqsubseteq s_j' \wedge u_1 \neq u_1' \wedge \cdots \wedge u_k \neq u_k' \wedge v_1 \not\sqsubseteq v_1' \wedge \cdots \wedge v_l \not\sqsubseteq v_l')$

is an axiom. Note that the satisfaction problem is decidable in the free term algebra (cf. [19]), hence the set of axioms of type 1 is recursive.

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

(1) $\vec{\beta} \to \vec{\alpha}$ (for $\vec{\alpha} \subseteq \vec{\beta}$)

(2) $\vec{\gamma_1} \vee \cdots \vee \vec{\gamma_n} \leftrightarrow \vec{\alpha} \wedge \vec{\beta}$

(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 1:

$$\forall PQnm(P\ generates\ n\ \wedge\ Q\ sends/receives\ m\ \wedge\ n \sqsubseteq m \to \neg(Q\ sends/receives\ m; P\ generates\ n))$$

(2) Ordering 2:

$$\forall PQmm'(P\ sends/receives\ m\ \wedge\ \{m'\}_{Q^{-1}} \sqsubseteq m \to \exists m''(Q\ sends\ m''; P\ sends/receives\ m\ \wedge\ \{m'\}_{Q^{-1}} \sqsubseteq m''))$$

(3) Nonce Verification 1:

$$\begin{aligned}\forall PQn_1m_2m_5m_6(&P\ generates\ n_1\ \wedge\ P\ sends\ m_2\ \wedge\ n_1 \sqsubseteq m_2\ \wedge\ P\ receives\ m_5\\ &\wedge\ n_1 \sqsubseteq m_6\ \wedge\ \{m_6\}_{Q^{-1}} \sqsubseteq m_5\\ &\wedge\ \forall m_7(P\ sends\ m_7 \wedge n_1 \sqsubseteq m_7 \to m_7 = m_2)\\ \to\ &\exists m_3m_4(P\ sends\ m_2; Q\ receives\ m_3; Q\ sends\ m_4; P\ receives\ m_5\\ &\wedge\ n_1 \sqsubseteq m_3 \wedge \{m_6\}_{Q^{-1}} \sqsubseteq m_4))\end{aligned}$$

(4) Nonce verification 2:

$$\begin{aligned}\forall PQn_1m_2m_5m_6(&P\ generates\ n_1\ \wedge\ P\ sends\ m_2\ \wedge\ n_1 \sqsubseteq m_2\ \wedge\ P\ receives\ m_5\\ &\wedge\ \{m_6\}_Q \not\sqsubseteq m_5\ \wedge\ n_1 \sqsubseteq m_5\ \wedge\ \{m_6\}_Q \sqsubseteq m_2\\ &\wedge\ \forall m_7(P\ sends\ m_7 \wedge n_1 \sqsubseteq m_7 \to m_7 = m_2)\\ &\wedge\ \forall m_8(n_1 \sqsubseteq m_8 \wedge m_8 \sqsubseteq m_2 \to m_8 \sqsubseteq \{m_6\}_Q)\\ \to\ &\exists m_3m_4(P\ sends\ m_2; Q\ receives\ m_3; Q\ sends\ m_4; P\ receives\ m_5\\ &\wedge\ \{m_6\}_Q \sqsubseteq m_3 \wedge n_1 \sqsubseteq m_4))\end{aligned}$$

(5) Nonce verification 3:

$$\begin{aligned}\forall PQn_1m_4m_5m_6m_9(&P\ generates\ n_1 \wedge P\ sends\ m_2 \wedge n_1 \sqsubseteq m_2 \wedge P\ receives\ m_5\\ &\wedge\ \{m_6\}_Q \not\sqsubseteq m_5\ \wedge\ n_1 \sqsubseteq m_5\ \wedge\ \{m_6\}_Q \sqsubseteq m_2\\ &\wedge\ \forall m_7(P\ sends\ m_7\ \wedge\ n_1 \sqsubseteq m_7 \to m_7 = m_2)\\ &\wedge\ \forall m_8(n_1 \sqsubseteq m_8\ \wedge\ m_8 \sqsubseteq m_2 \to m_8 \sqsubseteq \{m_6\}_Q)\\ &\wedge\ Q\ sends\ \{m_4\}_P\ \wedge\ n_1 \sqsubseteq m_4\\ &\wedge\ \forall m_{10}(Q\ sends\ m_{10}\ \wedge\ n_1 \sqsubseteq m_{10} \to m_{10} = \{m_9\}_P)\\ \to\ &\{m_9\}_P = m_5)\end{aligned}$$

Here, the expression "*sends/receives*" denotes *sends* or *receives*, and these are corresponding in each axiom. Ordering 1 and 2 represent ordering of

actions related to nonces and encrypted messages, respectively. Nonce Verification 1−3 largely depend on the idea of *authentication tests-based strand space method* introduced by [10]: Nonce Verification 1 is a formalization of *Incoming tests* and Nonce Verification 2 and 3 are formalizations of *Outgoing tests*.

2.3 Query form and correctness properties

Our aimed correctness properties are described in a special form of formulas, called *query form*. The query form includes a formalization of *principal's honesty* (denoted by $Honest(\vec{\alpha}^P)$), which is defined as follows.

Definition 2.3 (Principal's honesty)
$Honest(\vec{\alpha}^P[P,\vec{Q},\vec{n}])$
$\stackrel{\text{def}}{\equiv} \exists\vec{Q}\vec{n}\bigvee_{i\in\{j \mid m_j\in Sends(\vec{\alpha}^P[P,\vec{Q},\vec{n}])\}\cup\{k\}} \vec{\alpha}^P(i)[P,\vec{Q},\vec{n}]\wedge Only(\vec{\alpha}^P(i)[P,\vec{Q},\vec{n}])$

Here, $\vec{\alpha}^P[P,\vec{Q},\vec{n}]$ is a role of the form $P\,acts_1\,m_1;P\,acts_2\,m_2;\cdots;P\,acts_k\,m_k$ where each $acts_i$ $(1\leq i\leq k)$ is one of *sends*, *receives* and *generates*, and $\vec{\alpha}^P(i)[P,\vec{Q},\vec{n}]$ denotes an initial segment of $\vec{\alpha}^P[P,\vec{Q},\vec{n}]$ ending with $P\,acts_i\,m_i$ (for $0\leq i\leq k$), i.e., $\vec{\alpha}^P(i)[P,\vec{Q},\vec{n}]\equiv P\,acts_1\,m_1;\cdots P\,acts_i\,m_i$. (As a special case, $\vec{\alpha}^P(0)$ denotes $\top$.) $Only(\vec{\alpha}^P(i))$ denotes the following formula, whose intuitive meaning is "P performs only $\vec{\alpha}^P(i)[P,\vec{Q},\vec{n}]$".

$$\begin{aligned}Only(\vec{\alpha}^P(i))\equiv\ &\forall n_1(P\ generates\ n_1\rightarrow n_1\in Generates(\vec{\alpha}^P(i)))\\ &\wedge\forall m_2(P\ sends\ m_2\rightarrow m_2\in Sends(\vec{\alpha}^P(i)))\\ &\wedge\forall m_3(P\ receives\ m_3\rightarrow m_3\in Receives(\vec{\alpha}^P(i))))\end{aligned}$$

Here, $Sends(\vec{\alpha}^P(i))$ denotes the set $\{m_j \mid P\ sends\ m_j\subseteq\vec{\alpha}^P(i)\}$. ($Receives(\vec{\alpha}^P(i))$ and $Generates(\vec{\alpha}^P(i))$ are similar.) Set theoretical notation, such as $m\in Sends(\vec{\alpha}^P(i))$ (as well as $m\in Receivess(\vec{\alpha}^P(i))$ and $m\in Generates(\vec{\alpha}^P(i))$) is an abbreviation of the disjunctive form: for example, if $Sends(\vec{\alpha}^P(i))=\{m'_1,\ldots,m'_j\}$, then $m\in Sends(\vec{\alpha}^P(i))$ denotes the formula $(m=m'_1)\vee(m=m'_2)\vee\cdots\vee(m=m'_j)$. (As a special case, if $Sends(\vec{\alpha}^P(i))$ is empty then $m\in Sends(\vec{\alpha}^P(i))$ denotes $\bot$.)

Intuitively, each disjunct $\vec{\alpha}^P(i)\wedge Only(\vec{\alpha}^P(i))$ in $Honest(\vec{\alpha}^P)$ represents a historical record of P's actions at each step of his/her run: the sequence of actions $\vec{\alpha}^P(i)\equiv P\,acts_1\,m_1;\cdots;P\,acts_i\,m_i$ represents the P's performance at this step, and $Only(\vec{\alpha}^P(i))$ represents that P performs only $\vec{\alpha}^P(i)$. Especially, as a special case, $\vec{\alpha}^P(0)\wedge Only(\vec{\alpha}^P(0))$ represents that P performs no action. Thus, $Honest(\vec{\alpha}^P[P,\vec{Q},\vec{n}])$ represents "P performs only a (possibly multiple) run of an initial segment of $\vec{\alpha}^P$ which ends with a sending action or the last action of $\vec{\alpha}^P$, and uses the same data items $\vec{Q}$ and $\vec{n}$, for each run".

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

Example 2.4 (Initiator's honesty of the NS protocol)
$Honest(Init_{NS}[A/P, Q, n_1, n_2]) \equiv$

$$\exists Q n_1 n_2 \left(\begin{pmatrix} \forall n_3(A \textit{ generates } n_3 \to \bot) \\ \wedge \forall m_4(A \textit{ sends } m_4 \to \bot) \\ \wedge \forall m_5(A \textit{ receives } m_5 \to \bot) \end{pmatrix} \right.$$
$$\vee \begin{pmatrix} A \textit{ generates } n_1; A \textit{ sends } \{n_1, A\}_Q \\ \wedge \forall n_3(A \textit{ generates } n_3 \to n_3 = n_1) \\ \wedge \forall m_4(A \textit{ sends } m_4 \to m_4 = \{n_1, A\}_Q) \\ \wedge \forall m_5(A \textit{ receives } m_5 \to \bot) \end{pmatrix}$$
$$\left. \vee \begin{pmatrix} A \textit{ generates } n_1; A \textit{ sends } \{n_1, A\}_Q; A \textit{ receives } \{n_1, n_2\}_A; A \textit{ sends } \{n_2\}_Q \\ \wedge \forall n_3(A \textit{ generates } n_3 \to n_3 = n_1) \\ \wedge \forall m_4(A \textit{ sends } m_4 \to m_4 = \{n_1, A\}_Q \vee m_4 = \{n_2\}_Q) \\ \wedge \forall m_5(A \textit{ receives } m_5 \to m_5 = \{n_1, n_2\}_A) \end{pmatrix} \right)$$

Note that our formalization of honesty is stronger than the usual sense. That is, our definition of honesty is restricted to a single set of data items used for the honest principal's runs, whereas the usual sense of honesty means that P may perform multiple runs whose data items may differ from each other. However, by regarding a strict order-preserving merge of a certain number of the same role as a single role, we can represent the honesty with respect to any finite number of the sets of data items which are used for all possible runs by the honest principal. As an example, we present the case of initiator's (say, A's) honesty of the Needham-Schroeder protocol, such that A may use a couple of sets of data items $[Q, n_1, n_2]$ and $[Q', n_1', n_2']$.

$$\exists Q Q' n_1 n_2 n_1' n_2' \left(\begin{pmatrix} \forall n_3(A \textit{ generates } n_3 \to \bot) \\ \wedge \forall m_4(A \textit{ sends } m_4 \to \bot) \\ \wedge \forall m_5(A \textit{ receives } m_5 \to \bot) \end{pmatrix} \right.$$
$$\vee \begin{pmatrix} A \textit{ generates } n_1; A \textit{ sends } \{n_1, A\}_Q \\ \wedge \forall n_3(A \textit{ generates } n_3 \to n_3 = n_1) \\ \wedge \forall m_4(A \textit{ sends } m_4 \to m_4 = \{n_1, A\}_Q) \\ \wedge \forall m_5(A \textit{ receives } m_5 \to \bot) \end{pmatrix}$$
$$\vee \begin{pmatrix} A \textit{ generates } n_1; A \textit{ sends } \{n_1, A\}_Q; A \textit{ receives } \{n_1, n_2\}_A; A \textit{ sends } \{n_2\}_Q \\ \wedge \forall n_3(A \textit{ generates } n_3 \to n_3 = n_1) \\ \wedge \forall m_4(A \textit{ sends } m_4 \to m_4 = \{n_1, A\}_Q \vee m_4 = \{n_2\}_Q) \\ \wedge \forall m_5(A \textit{ receives } m_5 \to m_5 = \{n_1, n_2\}_A) \end{pmatrix}$$
$$\left. \vee \begin{pmatrix} A \textit{ generates } n_1; A \textit{ sends } \{n_1, A\}_Q \wedge A \textit{ generates } n_1'; A \textit{ sends } \{n_1', A\}_Q' \\ \wedge \forall n_3(A \textit{ generates } n_3 \to n_3 = n_1 \vee n_3 = n_1') \\ \wedge \forall m_4(A \textit{ sends } m_4 \to m_4 = \{n_1, A\}_Q \vee m_4 = \{n_1', A\}_{Q'}) \\ \wedge \forall m_5(A \textit{ receives } m_5 \to \bot) \end{pmatrix} \right)$$

$$\vee \left(\begin{array}{l} A \textit{ generates } n_1; A \textit{ sends } \{n_1, A\}_Q; A \textit{ receives } \{n_1, n_2\}_A; A \textit{ sends } \{n_2\}_Q \\ \quad \wedge A \textit{ generates } n_1'; A \textit{ sends } \{n_1, A\}_{Q'} \\ \quad \wedge \forall n_3 (A \textit{ generates } n_3 \rightarrow n_3 = n_1 \vee n_3 = n_1') \\ \quad \wedge \forall m_4 (A \textit{ sends } m_4 \rightarrow m_4 = \{n_1, A\}_Q \vee m_4 = \{n_2\}_Q \vee m_4 = \{n_1', A\}_{Q'}) \\ \quad \wedge \forall m_5 (A \textit{ receives } m_5 \rightarrow m_5 = \{n_1, n_2\}_A) \end{array} \right)$$

$$\vee \left(\begin{array}{l} A \textit{ generates } n_1; A \textit{ sends } \{n_1, A\}_Q; A \textit{ receives } \{n_1, n_2\}_A; A \textit{ sends } \{n_2\}_Q \\ \quad \wedge A \textit{ generates } n_1'; A \textit{ sends } \{n_1', A\}_{Q'}; A \textit{ receives } \{n_1', n_2'\}_A; A \textit{ sends } \{n_2\}_{Q'} \\ \quad \wedge \forall n_3 (A \textit{ generates } n_3 \rightarrow n_3 = n_1 \vee n_3 = n_1') \\ \quad \wedge \forall m_4 (A \textit{ sends } m_4 \\ \qquad \rightarrow m_4 = \{n_1, A\}_Q \vee m_4 = \{n_2\}_Q \vee m_4 = \{n_1', A\}_{Q'} \vee m_4 = \{n_2'\}_{Q'}) \\ \quad \wedge \forall m_5 (A \textit{ receives } m_5 \rightarrow m_5 = \{n_1, n_2\}_A \vee m_5 = \{n_1', n_2'\}_A) \end{array} \right)$$

First-order formalization of correctness properties

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

Definition 2.5 (Query form) *Query form* is a formula of the following form.

$$Honest(\vec{\alpha}^P) \wedge \vec{\beta}^Q \wedge Only(\vec{\beta}^Q) \rightarrow \vec{\gamma}$$

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

$$\begin{array}{l} Honest(\vec{\alpha}^P[A/P, Q, \vec{n}]) \wedge \vec{\beta}^Q[A/P, B/Q, \vec{N}/\vec{n}] \\ \quad \wedge Only(\vec{\beta}^Q[A/P, B/Q, \vec{N}/\vec{n}]) \rightarrow \vec{\alpha}^P[A/P, B/Q, \vec{N}/\vec{n}] \end{array}$$

The matching conversations and the injective agreement can be obtained by replacing the right hand side of implication with the strict order-preserving merge of $\vec{\alpha}^P[A/P, B/Q, \vec{N}/\vec{n}]$ and $\vec{\beta}^Q[A/P, B/Q, \vec{N}/\vec{n}]$ defined by the protocol Π, and with $\vec{\alpha}^P[A/P, B/Q, \vec{N}/\vec{n}] \wedge Only(\vec{\alpha}^P[A/P, B/Q, \vec{N}/\vec{n}])$, respectively.

Actually, our formalization of the agreement properties is weaker than the usual sense, because our honesty assumption is stronger than the usual sense. However, as we have explained in the definition of honesty (Definition 2.3), our query form can be extended so that the honest principal may use a finite number of sets of data items used for his/her runs.

A comparison between *BPL* and other protocol logics

In closing this chapter, we would like to point out some differences between *BPL* and other protocol logics.

One of the main differences is the formalization of inferences on honesty. In the protocol logics of [9,7,16], the notion of honesty is formalized as an atomic formula and inferences on honesty are drawn by a special inference rule. On the other hand, in *BPL*, the notion of honesty is formalized as a non-atomic formula and all inferences on honesty are drawn by logical inference rules, and the inferences on honesty essentially used in [9,7,16] are derived rules in *BPL*. More precisely, to prove protocols correct, the protocol logics of [9,7,16] implicitly or explicitly use the three types of inferences: we pick up these inferences as *Honesty rules* presented in Appendix A.

As for the non-logical axioms, our proposed non-logical axioms (1)-(5) of (III) introduced in Section 2.2 do not essentially depend on our framework. Especially, as we shall show in Section 4, our completeness and decidability arguments are not affected by the choice of non-logical axioms. Our choice of non-logical axioms in this paper is one of the simplest formalism sufficient to prove our aimed correctness properties. Then the basic inference rules used in the protocol logics are essentially derived rules of *BPL*. Here we emphasize that *BPL* is formalized in first-order predicate logic without any temporal modal operators nor Floyd-Hoare style dynamic operator which are used in the protocol logics of [9,7]: this is the reason why we call our system "basic".

3 Formal Semantics and Soundness

In this section, we give a semantics of our system.

A *trace model* (or *model*) is $M = (D_P, D_N, \vec{\alpha}, \Phi)$ where D_P is a set, called a *name domain*, D_N is a set called a *nonce domain*, D_M is the free algebra domain (i.e., the set of first-order terms) determined by D_P and D_N along with the term construction rules using $\langle\ ,\ \rangle$, $K(\ ,\ )$ and $K^{-1}(\ ,\ )$, (where $\langle\ ,\ \rangle$ is the ordered pair function symbol, and $K(\ ,\ )$ and $K^{-1}(\ ,\ )$ are the key-encryption and decryption function symbols), and $\vec{\alpha}$ is a trace. (Note that we often use abbreviation, say $\{n_1, \{n_2\}_A, n_3\}_{B^{-1}}$ for $K^{-1}(B, \langle\langle n_1, K(A, n_2)\rangle, n_3\rangle)$.) In this paper we consider only three kinds of forms of actions: A *sends* m, A *receives* m and A *generates* n, where $A \in D_P$, $m \in D_M$. A trace $\vec{\alpha}$ is of the form of a *finite* sequence $\alpha_1; \cdots; \alpha_n$ of (primitive) actions α_i. $\Phi(A) \in D_P$ for a constant symbol A of the name sort, and $\Phi(N) \in D_N$ for a constant symbol N of the nonce sort. We extend Φ to evaluation of variables such that $\Phi(P) \in D_P$ and $\Phi(n) \in D_N$, as usual. $\Phi(\langle t_1, t_2\rangle) = \langle\Phi(t_1), \Phi(t_2)\rangle$, $\Phi(K(A,t)) = K(\Phi(A), \Phi(t))$, $\Phi(K^{-1}(A,t)) = K^{-1}(\Phi(A), \Phi(t))$. For any action predicate $\alpha(A, m)$, $\Phi(\alpha(A,m)) = \alpha(\Phi(A), \Phi(m))$. For a trace formula of the form $\alpha_1; \cdots; \alpha_n$, $\Phi(\alpha_1; \cdots; \alpha_n) = \Phi(\alpha_1); \cdots; \Phi(\alpha_n)$. For a model $M = (D_P, D_N, \vec{\alpha}, \Phi)$, $m_1 \sqsubseteq m_2$ is true in M iff $\Phi(m_1)$ is a subterm of $\Phi(m_2)$, and $m_1 = m_2$ is true in M iff $\Phi(m_1)$ and $\Phi(m_2)$ are identical terms (of the same sort). $\vec{\beta}$ is true in M iff $\Phi(\vec{\beta}) \subseteq \vec{\alpha}$. All first-order logical connectives are interpreted in the standard way.

Theorem 3.1 (Soundness) *For any closed formula of the query form*

$$Honest(\vec{\alpha}^P) \wedge \vec{\beta}^Q \wedge Only(\vec{\beta}^Q) \to \vec{\gamma},$$

if this formula is provable in BPL, then it is true for any model $M = (D_P, D_N, \vec{\alpha}, \Phi)$.

This theorem is proved by a standard induction on the length of the proof.

4 Completeness, Decidability and Their Application to Counter-Example Generations

In this section, we first show the completeness theorem for the query form by means of the proof-search method. Moreover, as a corollary of our completeness proof, we also show the decidability of provability for any given query. (Actually, in this preliminary report we only sketch out these proofs. The detailed proofs will appear in the full version of this paper.) Next, as an application of these results, we show how to find an attack on the protocol in question. As the main result, with the aid of Comon-Treinen's algorithm for the intruder deduction problem [6], for any given query we can determine whether there exists an attack on the protocol in question, whenever we set any upper-bound on the number of data items. At the end of this section, we also present a concrete example of our proof construction/counter-example generation.

4.1 Completeness and decidability for the query form

Our completeness is stated as follows.

Theorem 4.1 (Completeness) *If a closed formula of the query form*

$$Honest(\vec{\alpha}^P) \wedge \vec{\beta}^Q \wedge Only(\vec{\beta}^Q) \to \vec{\gamma}$$

is true for any model $M = (D_P, D_N, \vec{\delta}, \Phi)$*, then this formula is provable in BPL.*

To prove the completeness theorem, we use the proof-search method (which is essentially the same as Beth's tableau method). Especially, we follow the method and several terminologies (such as *stage*, *branch*, and *available terms*) described in Section 1.8 of Takeuti [17]. In order to fix our query form to the sequent calculus-style proof-search method, we first slightly modify our query form as the following sequent, called a *query sequent*.

$$Honest(\vec{\alpha}^P),\ \vec{\beta}^Q,\ Only(\vec{\beta}^Q),\ Axioms(1)\text{-}(5) \vdash \vec{\gamma}$$

Here, $Axioms(1)$-(5) denotes the set of non-logical axioms (1)-(5) of (III) introduced in Section 2.2. We put these formulas as assumptions of the query.

Now we review the proof-construction process and remark the point to be slightly modified in our setting.

Proof-construction process. For any query sequent S, we shall define the *proof-construction tree for* S (denoted by $\mathcal{T}(S)$). $\mathcal{T}(S)$ is a (possibly infinite) tree which is constructed in *rounds*: the proof-construction process begins with Round 0, where we write the query sequent S at the bottom of the tree, and go to Round 1. Then each Round i (for $i = 1, 2, \ldots$) consists of *stages* ($k = 0, 1, 2, \ldots, 14$) defined by cases:

Case I: Every topmost sequent, which is of the form $\Gamma \vdash \Delta$, satisfies one of the following conditions (C1)−(C3), then the proof-construction process terminates. We call such a sequent *closed sequent*.
(C1) Γ and Δ include a formula in common.
(C2) $t_1 = t'_1, \ldots, t_n = t'_n, s_1 \sqsubseteq s'_1, \ldots s_m \sqsubseteq s'_m \in \Gamma$ and $u_1 = u'_1, \ldots, u_k = u'_k, v_1 \sqsubseteq v'_1, \ldots, v_l \sqsubseteq v'_l \in \Delta$, and the literal $\{t_1 = t'_1, \ldots, t_n = t'_n, s_1 \sqsubseteq s'_1, \ldots s_m \sqsubseteq s'_m, u_1 \neq u'_1, \ldots, u_k \neq u'_k, v_1 \not\sqsubseteq v'_1, \ldots, v_l \not\sqsubseteq v'_l\}$ is not satisfiable in the free term algebra of our language. This condition is essentially the same as axiom (I) in Section 2.2.
(C3) Γ includes a trace formula $\vec{\alpha}$ and Δ includes a trace formula $\vec{\beta}$ with $\vec{\beta} \subseteq \vec{\alpha}$. This condition is essentially the same as axiom (1) of (II) in Section 2.2.

Case II: Not Case I. Then this stage is defined according as $k = 0, 1, 2, \ldots, 13$, 14, where the cases $k = 0, 1, 2, \ldots, 11$ and 14 are the same as [17], i.e., $k = 0$ and 1 concern the symbol $\neg$, $k = 2$ and 3 concern $\wedge$, $k = 4$ and 5 concern $\vee$, $k = 6$ and 7 concern $\rightarrow$, $k = 8$ and 9 concern $\forall$, $k = 10$ and 11 concern $\exists$. In addition to the above stages, we insert the following rules as $k = 12$ and 13.

- $k = 12$ (equality rule): Let $\Gamma[t] \vdash \Delta[t]$ be any topmost sequent of the tree which has been defined by stage $k - 1$. Then write down
  $\Gamma[t] \vdash \Delta[t], s = t$ and $s = t, \Gamma[t], \Gamma[s/t] \vdash \Delta[t], \Delta[s/t]$
  above $\Gamma[t] \vdash \Delta[t]$ for all terms s and t, which are made by all available terms of sorts `name` and `nonce` with function symbols.
  We introduce this stage instead of the infinite scheme $\forall x \forall y(x = y \wedge F(x) \rightarrow F(y))$ as a hypothesis in the query.
- $k = 13$ (rule for trace formulas): Let $\Gamma \vdash \Delta$ be any topmost sequent of the tree which has been defined by stage $k - 1$, and $\vec{\alpha}_1, \ldots, \vec{\alpha}_j$ be the trace formulas appearing in Γ. Then write down all sequents of the form
  $\vec{\gamma}_i, \Gamma' \vdash \Delta$
  above $\Gamma \vdash \Delta$, where $\vec{\gamma}_i$ is an order-preserving merge of $\vec{\alpha}_1, \ldots, \vec{\alpha}_j$, and Γ' is $\Gamma - \{\vec{\alpha}_1, \ldots, \vec{\alpha}_j\}$.
  This stage is the combination of the if-part of axiom (II) (2) in Section 2.2 (i.e., $\vec{\gamma}_1 \vee \cdots \vee \vec{\gamma}_n \rightarrow \vec{\alpha} \wedge \vec{\beta}$) and the $\vee$-left reduction. Note that we do not need to consider the only-if part of axiom (II) (2) (i.e., $\vec{\alpha} \wedge \vec{\beta} \rightarrow \vec{\gamma}_1 \vee \cdots \vee \vec{\gamma}_n$), because in our proof-construction process there is no sequent such that a formula of the form $\vec{\alpha} \wedge \vec{\beta}$ appears in the right hand side.

After applying the rule of Stage 14, if the topmost sequent is not closed, then

we go to Round $i+1$ and repeat the above procedure.

Proof sketch of the completeness theorem. From now we show our completeness proof, which is proved by using the following lemma.

Main Lemma. *If there exists a branch $S_0, \ldots, S_{3\times 14}$ in $\mathcal{T}(S_0)$, where $S_{3\times 14}$ is a sequent at the end of Round 3 and not closed, then there exists a counter-model $M = (D_P, D_N, \vec{\delta}, \Phi)$ for S_0.*

Here we sketch out how to construct such a counter-model M from $S_{3\times 14}$.

Assume that $\vec{\theta}$ is the trace formula appearing in the left hand side of $S_{3\times 14}$. (Note that at the end of each round, the left hand side of the sequent always includes only a single trace formula.) We fix D_P and D_N by the following steps: we first take the set of all literals appearing in $S_{3\times 14}$ and solve the satisfaction problem of these literals, then decompose each literal which consists of compound terms (e.g., $\{N_1, A\}_B = \{n_1, P\}_Q$ is decomposed as $N_1 = n_1$, $A = P$ and $B = Q$), then take D_P and D_N as representatives of these decomposed literals. We define the assignment Φ for terms by induction on the length of terms as follows. As the base case, each constant and variable of sort `name` or `nonce` is interpreted by its representative (i.e., $\Phi(N) = N^*$ $(\in D_N)$, $\Phi(n) = n^*$ $(\in D_N)$ where N^* and n^* are the representatives of equivalence classes of N and n, respectively, and the interpretation for terms of sort `name` is similar.) Each variable (say, m) of sort `message` which is neither of sort `name` nor `nonce` is interpreted by the representative of the equivalence class of m. The induction step for terms and the definition of evaluation for each formula are followed by the definition of Φ in Section 3. Finally, as $\vec{\delta}$, we take $\vec{\delta} = \Phi(\vec{\theta})$.

From now we show that M is a counter-model of S_0. The essential idea to prove this fact is to use the following facts: (1) Every non-logical Π^0_2 axiom is satisfiable in M; (2) Axioms about trace formulas are satisfiable in M; (3) Axioms about $=$ and $\sqsubseteq$ are satisfiable in M. As for the fact (1), for any branch in a proof-construction tree, if an eigenvariable (say, m) appears in the branch, such an eigenvariable always appears in a formula of the form $A\, acts\, m$. On the other hand, as the descendants of the honesty assumption, $\neg A\, acts\, m \vee m = t$ always appears in the branch, where t is a term appearing at this stage. Thus, if T is the set of terms in Round 3, for the eigenvariable m which appears above Round 3, an equation $m = t$ with some $t \in T$ always appears in the left side, then the search domain does not increase above Round 3. As for the facts (2) and (3), these are immediately derived from the correspondence of the logical axioms (I) (introduced in Section 2.2) and the termination condition (C2), and the correspondence of the logical axiom (II) (introduced in Section 2.2) and the termination condition (C3), respectively.

By this Main Lemma, proof of our completeness theorem goes as follows. For any given query form, if each branch of the reduction tree up to Round

3 terminates, then we can easily write a proof of this query. Then by the contraposition, for any unprovable query there exists a branch which includes a non-closed sequent (say, $S_{3\times 14}$) at the end of Round 3. By Main Lemma, we obtain a counter-model from the information of $S_{3\times 14}$. Then by the contraposition, the completeness theorem holds. □

This Main Lemma guarantees not only that our completeness holds, but also that we only need to make a proof-construction tree up to Round 3 to find a counter-model. Therefore, if we introduce a suitable enumeration of all instantiations for ∀-left, ∃-right and equality rules, the following decidability is immediately derived from Main Lemma.

Corollary 4.2 (Decidability) *For any given query form, the provability of the query in BPL is decidable.*

Moreover, the following theorem holds.

Theorem 4.3 (Finite number of minimal counter-models) *For any unprovable query form, the number of counter-models, which are obtained from the proof-construction tree of the query up to the end of Round 3, is finite.*

4.2 Construction of attacked processes

By the proof-search method presented in the previous subsection, for any given query we obtain a proof if the query holds. Otherwise we obtain counter-models for the query. However, the traces obtained from these counter-models cannot be immediately considered as attacked processes because in our semantics we consider as a "trace" any sequence of primitive actions. In order to find an attacked process, we introduce the notion of *realizable trace* by means of the Comon-Treinen's algorithm for the intruder deduction problem such that *for any given finite (or regular) set of messages T, and for any given message m, whether it is possible for the intruder to retrieve m from T or not.*

The definition of *realizable trace* is as follows.

Definition 4.4 (Realizable trace) Let $\vec{\delta}$ be a sequence of actions, P_1 *receives* m_1, $\ldots$, P_k *receives* m_k be the list of all receiving actions in $\vec{\delta}$, and $\vec{\delta}(i)$ be the initial segment of $\vec{\delta}$, which ends with P_i *receives* m_i. $\vec{\delta}$ is realizable if it satisfies the following condition: for any P_i *receives* m_i $(1 \leq i \leq k)$, m_i is provable (retrievable) from $Sends(\vec{\delta}(i-1))$ in Comon-Treinen's system of [6].

Intuitively, a realizable trace is a sequence of actions, where each receiving message can be generated by the Dolev-Yao intruder [8]. Clearly, from a realizable trace, which is obtained from a counter-model of a query, we can easily construct a concrete attacked process on the protocol in question by inserting some suitable intruder's actions.

Since the procedure to check the realizable trace is decidable (cf. [6]), the following theorem is immediately derived.

Theorem 4.5 (Decidability of the attacked process detection) *For any given query, the problem whether there exists a counter-model* $M = (D_P, D_N, \vec{\delta}, \Phi)$ *such that* $\vec{\delta}$ *is an realizable trace is decidable.*

As we have explained in the definition of our query form, we can represent the correctness properties for any number of sets of data items (used by the honest principal). Thus, this decidability guarantees that we can determine whether there exists an attacked trace whenever we set any upper-bound on the number of data items used by the principals. This decidability corresponds to the result of former works (cf. [4]).

A suitable tactics to find an attack on the protocol in question is to increase the search domain by extending the query form. This procedure is generally infinite if we set no limit to the search domain. However, this procedure is optimal since the same undecidability result also holds in some other framework (cf. [4]).

4.3 An example of attacked process detection

In this subsection, we show a simple example of how to find an attacked process from counter-models obtained by a proof-search of unprovable query. Here we consider the matching conversations for the Needham-Schroeder public key protocol (from responder's view), whose query sequent is as follows.

$$\begin{array}{l} \mathit{Resp}_{NS}[A/P, B/Q, N_1/n_1, N_2/n_2],\ \mathit{Honest}(\mathit{Init}_{NS}[A, Q, n_1, n_2]),\ \mathit{Axioms}, \\ \quad \forall n_3(B \textit{ generates } n_3 \to n_3 = N_2),\ \forall m_4(B \textit{ sends } m_4 \to m_4 = \{N_1, N_2\}_A), \\ \quad \forall m_5(B \textit{ receives } m_5 \to m_5 = \{N_1, A\}_B \vee m_5 = \{N_2\}_B) \\ \vdash A \textit{ generates } N_1; A \textit{ sends } \{N_1, A\}_B; B \textit{ receives } \{N_1, A\}_B; B \textit{ generates } N_2; \\ \quad B \textit{ sends } \{N_1, N_2\}_A; A \textit{ sends } \{N_2\}_B; B \textit{ receives } \{N_2\}_B \end{array}$$

In the proof-construction tree of this query, there are 16 non-closed branches at the end of Round 3, which are of the following form. (Here we use schematic expressions: t_1 is N_1 or n_1, t_2 is N_2 or n_2.)

- in the left hand side, a trace formula appears which is made by order-preserving merges of the following trace formulas:
  - A *generates* t_1; A *sends* $\{t_1, A\}_Q$; A *receives* $\{t_1, t_2\}_A$; A *sends* $\{t_2\}_Q$
  - B *receives* $\{t_1, A\}_B$; B *generates* t_2; B *sends* $\{t_1, t_2\}_A$; B *receives* $\{t_2\}_B$
  - B *sends* $\{t_1, t_2\}_A$; A *receives* m_1; A *sends* m_2; B *receives* $\{t_2\}_B$
- $A \neq B$, $A \neq Q$, $B \neq Q$, $N_1 \neq N_2$, $n_1 \neq n_2$, $N_1 = n_1$, $N_2 = n_2$, $m_1 = \{t_1, t_2\}_A$ and $m_2 = \{t_2\}_Q$ are satisfied by all equations in this sequent.

Among the counter-models obtained by these non-closed sequents, we can find a model $M = (D_P, D_N, \vec{\alpha}, \Phi)$ such that $D_P = \{A, B, Q\}$, $D_N = \{N_1, N_2\}$, where $A \neq B$, $A \neq Q$, $B \neq Q$, $N_1 \neq N_2$, $\vec{\alpha} =$ A *generates* N_1; A *sends* $\{N_1, A\}_Q$; B *receives* $\{N_1, A\}_B$; B *generates* N_2; B *sends* $\{N_1, N_2\}_A$; A *receives* $\{N_1, N_2\}_A$; A *sends* $\{N_2\}_Q$; B *receives* $\{N_2\}_B$. Moreover, the trace $\vec{\alpha}$ is realizable: actually, we can construct an attacked process as follows.

A *generates* N_1; A *sends* $\{N_1, A\}_Q$; Q *receives* $\{N_1, A\}_Q$; Q *sends* $\{N_1, A\}_B$;
B *receives* $\{N_1, A\}_B$; B *generates* N_2; B *sends* $\{N_1, N_2\}_A$; A *receives* $\{N_1, N_2\}_A$
A *sends* $\{N_2\}_Q$; Q *receives* $\{N_2\}_Q$; Q *sends* $\{N_2\}_B$; B *receives* $\{N_2\}_B$.

This is the scenario of the attack on the Needham-Schroeder protocol detected by Lowe [13].

Remark. It is known that the modification of the second message (i.e., replacing $\{N_1, N_2\}_A$ with $\{N_1, N_2, B\}_A$) makes impossible the Lowe's attack. Actually, in our case, this modification changes the A's honesty as follows: $\forall m(A \textit{ receives } m \to m = \{n_1, n_2, Q\}_A)$. Then, after the appearance of B *sends* $\{N_1, N_2, B\}_A$; A *receives* m_1; A *sends* m_2; B *receives* $\{N_2\}_B \wedge \{N_1, N_2, B\}_A \sqsubseteq m_1 \wedge N_2 \sqsubseteq m_2$, the equation $B = Q$ should appear in the left hand side. (Otherwise such branches should be closed by the termination condition (C2).) Therefore, the trace formula A *generates* N_1; A *sends* $\{N_1, A\}_B$; A *receives* $\{N_1, N_2, B\}_A$; A *sends* $\{N_2\}_B$ should appear in the left hand side. Then, by the termination condition (C3), such branches should be closed. Therefore, all branches should be closed and then the agreement of this modified protocol is provable.

5 Conclusions and Future Work

We introduced an axiomatic system in first-order predicate logic for proving protocols correct, called *Basic Protocol Logic*, as a simple formulation of a basic part of the protocol logics [9,7,16]. We also gave a simple trace-based semantics which is complete for our query form in *BPL*. Moreover, as a corollary of our completeness proof, we obtained the decidability of provability in *BPL* with respect to our query. Then by combining our completeness proof and the Comon-Treinen's algorithm, for any given query we can generate concrete attacks on the protocol in question, if any, whenever we set any upper bound on the number of data items used in the protocol.

There are several directions in which this work can be developed. First, we are interested in the extension in order to prove secrecy property about session keys issued in a protocol. As we have mentioned in Section 1, by this extension we can treat correctness properties which are related to secrecy property, such as in the case of Kerveros Version 5 (cf. [3]). We are also interested in compositional approach to prove correctness of compound protocols (cf. [9,7,16]). In our previous work [11,12], we proposed some inference systems to prove correctness properties of a composed protocol by reusing proofs about its components. The main idea in [11,12] was to weaken the notion of honesty. However, our decidability result does not hold by introducing such a weak honesty. In this paper we omitted the compositionality, although we consider such a problem to be one of our good target to develop our framework.

Acknowledgments

We thank Andre Scedrov and Iliano Cervesato for their helpful comments and discussions at early stages of this research.

References

[1] `www.stanford.edu/~danupam/logic-derivation.html`

[2] M. Burrows and M. Abadi and R. Needham. A Logic of Authentication. *Technical Report 39*, Digital System Research Center, 1989.

[3] F. Butler, I. Cervesato, A.D. Jaggard and A. Scedrov. A Formal Analysis of Some Properties of Kerberos 5 Using MSR. *University of Pennsylvania Department of Computer and Information Science Technical Report MS-CIS-04-04*, 59 pages, 2004.

[4] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell and A. Scedrov. Multiset Rewriting and the Complexity of Bounded Security protocols. *Journal of Computer Security*, vol.12, no.1, pp.677-722, 2004.

[5] J. Clark and J. Jacob. A Survey of Authentication Protocol Literature: Version 1.0 (web draft), 1997.

[6] Hubert Comon-Lundh and Ralf Treinen. Easy Intruder Deductions. In *Verification: Theory and Practice, essays dedicated to Zohar Manna*. LNCS series 2772, Springer-Verlag, 2003.

[7] A. Datta and A. Derek and J. C. Mitchell and D. Pavlovic. A Derivation System for Security Protocols and its Logical Formalization. *Journal of Computer Security (Special Issue of Selected Papers from CSFW-16)*, 52 pages, to appear, 2005.

[8] D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, vol.29, no.2, 1983.

[9] N. Durgin and J. C. Mitchell and D. Pavlovic. A Compositional Logic for Proving Security Properties of Protocols. *Journal of Computer Security*, vol.11, no.4, pp.677-721, 2003.

[10] J. D. Guttman and F. J. Thayer Fábrega. Authentication Tests. *IEEE Symposium on Security and Privacy*, pp.96-109, 2002.

[11] K. Hasebe and M. Okada. Inferences on Honesty in Compositional Logic for Protocol Analysis. *Proceedings of the International Symposium on Software Security 2003*, LNCS 3233, pp.65-86, 2004.

[12] K. Hasebe and M. Okada. Non-monotonic Properties for Proving Correctness in a Framework of Compositional Logic. *Proceedings of the workshop on Foundations of Computer Security '04*, pp.97-113, 2004.

[13] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-key Protocol Using FDR. *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, vol.1055, pp.147-166, 1996.

[14] G. Lowe. A Hierarchy of Authentication Specifications. *Proceedings of the 10th Computer Security Foundations Workshop*, pp. 31-43, 1997.

[15] R. Needham and M. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, vol.21, no.12, pp.993-999, 1978.

[16] C. Meadows and D. Pavlovic. Deriving, Attacking and Defending the GDOI Protocol. *Proceedings of 9th European Symposium on Research in Computer Security*. pp.53-72, 2004.

[17] G. Takeuti. *Proof Theory (2nd edition)*. North-Holland. 1987.

[18] F. J. Thayer Fábrega, J. C. Herzog and J. D. Guttman. Strand Spaces: Why is a Security Protocol Correct? *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pp.160-171, 1998.

[19] K. N. Venkataraman. Decidability of the Purely Existential Fragment of the Theory of Term Algebras. *Journal of the ACM*, vol.34 no.2, pp.492-510, 1987.

[20] T. Y. C. Woo and S. S. Lam. Verifying authentication protocols: Methodology and example. *Proceedings of the International Conference on Network Protocols*, 1993.

A Some Derived Rules on Honesty

In this appendix, we present some rules on honesty (called *honesty rules*), which are explicitly or implicitly used to prove security protocols correct in the protocol logics of [9,7,16]. Although we omit the formal proofs, we can easily prove that these rules are derived rules in *BPL*. We also show a simple example of correctness proof (i.e., the non-injective agreement property of the Needham-Schroeder-Lowe protocol from the responder's view) in the system where the notion of honesty (denoted by $Honest(\vec{\alpha}^P)$) is introduced as an atomic formula and the honesty rules as axioms.

Honesty rules

The honesty rules with respect to P's role $\vec{\alpha}^P$ ($\equiv P\, acts_1\, m_1; \cdots; P\, acts_k\, m_k$) are the following formulas, and we admit the formulas obtained by replacing each *sends* with *receives* or *generates*.

(Hon 1) $\forall m(P\, sends\, m \wedge Honest(\vec{\alpha}^P) \rightarrow m \in Sends(\vec{\alpha}^P))$

(Hon 2) $\forall m(P\, sends\, m \wedge Honest(\vec{\alpha}^P) \rightarrow P\, sends\, m_i \vee m \in Sends(\vec{\alpha}^P) - \{m_i\})$

(Hon 3) $P\, sends\, m_i \wedge Honest(\vec{\alpha}^P) \rightarrow P\, acts_{i-1}\, m_{i-1}; P\, sends\, m_i$

The following rules (1′) and (2′) are variants of (1) and (2), respectively.

(Hon 1′) $\forall m'(\exists m(P\ sends\ m \wedge m' \sqsubseteq m) \wedge Honest(\vec{\alpha}^P) \rightarrow m' \sqsubseteq m'' \in Sends(\vec{\alpha}^P))$

(Hon 2′) $\forall m'(\exists m(P\ sends\ m \wedge m' \sqsubseteq m) \wedge Honest(\vec{\alpha}^P) \rightarrow P\ sends\ m_i \vee m' \sqsubseteq m'' \in Sends(\vec{\alpha}^P) - \{m_i\})$

Here, the set theoretical notation such as $m \in Sends(\vec{\alpha}^P)$ is the same abbreviation used in Definition 2.3 in Section 2.3. The abbreviation $m' \sqsubseteq m'' \in Sends(\vec{\alpha}^P)$ denotes the formula $m' \sqsubseteq m''_1 \vee m' \sqsubseteq m''_2 \vee \cdots \vee m' \sqsubseteq m''_j$, where $Sends(\vec{\alpha}^P) = \{m''_1, \ldots, m''_j\}$.

Note that these rules (1), (2) and (3) correspond to the rules introduced in our previous works [11,12]: *Substitution*, *Matching* and *Deriving another action*, respectively. See [11,12] for the intuitive meanings of these rules.

Proof of the non-injective agreement of the NS protocol from responder's view

Init	$Resp[A, B, N_1, N_2] \vdash Resp[A, B, N_1, N_2]$	(1)
(1), NV2	$Resp[A, B, N_1, N_2]$ $\vdash \exists m_1 m_2 (A\ rec\ m_1; A\ sen\ m_2 \wedge \{N_1, N_2, B\}_A \sqsubseteq m_1 \wedge N_2 \sqsubseteq m_2)$	(2)
(2), Hon 1′, Eq	$Resp[A, B, N_1, N_2] \wedge Honest(Init[P, Q, n_1, n_2])$ $\vdash N_1 = n_1 \wedge N_2 = n_2 \wedge B = Q$	(3)
(2), Hon 2′	$Resp[A, B, N_1, N_2] \wedge Honest(Init[P, Q, n_1, n_2])$ $\vdash A\ sen\ \{N_2\}_B \vee N_2 \sqsubseteq \{n_1, A\}_Q$	(4)
(4), Eq	$Resp[A, B, N_1, N_2] \wedge Honest(Init[P, Q, n_1, n_2])$ $\vdash A\ sen\ \{N_2\}_B$	(5)
(5), Hon 3	$Resp[A, B, N_1, N_2] \wedge Honest(Init[P, Q, n_1, n_2])$ $\vdash A\ gen\ n_1; A\ sen\ \{n_1, A\}_Q; A\ rec\ \{n_1, n_2, Q\}_A; A\ sen\ \{n_2\}_Q$	(6)
(2), (6), Eq	$Resp[A, B, N_1, N_2] \wedge Honest(Init[P, Q, n_1, n_2])$ $\vdash A\ gen\ N_1; A\ sen\ \{N_1, A\}_B; A\ rec\ \{N_1, N_2, B\}_A; A\ sen\ \{N_2\}_B$	(6)