The ISC Int'l Journal of Information Security

January 2009, Volume 1, Number 1 (pp. 55–67)

http://www.isecure-journal.org

Process Algebraic Modeling of Authentication Protocols for Analysis of Parallel Multi-Session Executions

Rasool Ramezanian [a,*]

[a] *Department of Mathematical Science, Sharif University of Technology, Tehran, Iran*

ARTICLE INFO

Article history:
Received: 1 July 2008
Revised: 14 November 2008
Accepted: 20 January 2009
Published Online: 28 January 2009

Keywords:
Authentication, Process Algebra, Parallel Sessions, Security

Abstract

Many security protocols have the aim of authenticating one agent acting as initiator to another agent acting as responder and vice versa. Sometimes, the authentication fails because of executing several parallel sessions of a protocol, and because an agent may play both the initiator and responder role in parallel sessions. We take advantage of the notion of transition systems to specify authentication for parallel multiple sessions execution. To model the authentication, two main notions called 1. *agent's scope* and 2. *agent's recognizability* are introduced, which consider the difference of ability of agents due to their different roles in the protocol and different access to keys and secrets. To formalize above notions, a process algebra provided by some primitives for manipulating cryptographic messages is used. We formalize some security protocols and examine our definition of authentication for them. We just discuss the symmetric key case.

1 Introduction

Security protocols are hard to design, even under the assumption of perfect cryptography. They are notoriously error prone, and many of the security protocols appearing in the literature have been shown to be flawed. Security protocols have various aims, such as *distributing a key*, *non-repudiation*, *secrecy*, *fairness*, and achieving *authentication*: where each agent becomes assured of the other's identity. To specify these aims, many formal methods have been proposed: Logic approaches as BAN, and its extensions [1, 2, 3, 4, 5, 6]; process algebraic approaches as the application of CSP to security protocols pioneered by Roscoe and Lowe [7, 8, 9]; and strand space [10, 11] approach. In this paper, *authentication* is modeled using the syntax of the process algebra [12]. We focus on attaining *authentication* in the *parallel multiple sessions* of the execution of protocols. Some systems permit several parallel execution of a protocol, (as an example, a server which several clients can log on); in this situation, the attacker may use several parallel runs of the protocol to fool one of the participants. For example, in the *mirror* attack, the trick is to let a participant answer his own question.

We model protocols by transition systems, and specify *authentication* as a property for transition systems, where *parallel multiple sessions* of protocols are considered. To do this, two notions are introduced which consider the capability of agents:

(1) *agent's scope*, different agents may have different *scopes* to sessions of a protocol; it is possible that an agent assumes two sessions equal and another agent assumes them different, (Definition 10),
(2) *agent's recognizability*, due to their different ac-

* Corresponding author.
Email address: ramezanian@cs.sharif.edu (R. Ramezanian).
ISSN: 2008-2045

cess to keys and secrets, different agents may have different power of recognizability of types of messages; it is possible that an agent can distinguish two different types of messages and another agent cannot (Definition 14).

In Section 2, various definitions of *authentication*, appearing in the literature, and what we mean by the term *authentication* are discussed. In Section 3, we model protocols and specify authentication using transition systems. In Section 4, the SPA, *secure process algebra*, is considered to specify security protocols, and to formalize two key notions: *agent's scope* and *agent's recognizability*. The SPA is not novel; its terms can be translated into the μCRL language [13]. It is not aimed to introduce a new process algebra, and the SPA is just considered for convenience of taking advantage of its terms to formalize the notions agent's recognizability and agent's scope. In Section 5, using the SPA, some security protocols are formalized, and finally, in the last Section 6, some theorems are indicated to show that it is decidable whether a protocol is secure or not.

2 Authentication

Many security protocols discussed in the literature have the aim of attaining authentication, i.e., one participant should become sure of the identity of an other. We consider two kinds of protocols, 1. protocols that aim to authenticate a responder R to an initiator I, and 2. protocols that aim to authenticate an initiator I to a responder R. It is differentiated between the two terms *'agent'* and *'participant'*. Participants are roles intended by a protocol designer such as *initiator*, *server* and *responder*. Agents are principals who execute the protocol. An agent may play more than one role; for example, she may play both the initiator and responder roles in parallel runs of the protocol.

There are various meanings of the term authentication, and there exists a hierarchy of authentication; *aliveness*, *weak agreement*, *non-injective agreement* and *injective agreement* spotlighted by Gavin Lowe in [14]. *Aliveness* is considered to be the weakest reasonable definition of authentication: *whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol.* From strengthening the *aliveness* to insisting that B agreed she was running the protocol with A, arises *weak agreement*. *Non-injective agreement* adds the condition that the two agents A, B agree as to which roles each was taking, and that they agree upon some of the data items used in the exchange. *Injective agreement* guarantees that there is a one to one relationship between the two agents' runs.

Some attacks can occur due to parallel runs of a protocol as *mirror attack* and *multiplicity attack* [15]. As it is mentioned, we study authentication for this particular condition. To do this, a definition of authentication for this case is considered. The definition of authentication which covers *weak agreement* and also guarantees that there is a one to one relationship between the two agents' runs is as follows (see also [16]):

(1) A protocol authenticates a *responder* to an *initiator*, whenever an agent A starts j runs of the protocol as an initiator and l runs as a responder *all in parallel*; and completes $k \leq j$ runs of the protocol acting as initiator apparently with responder B, then B has recently been running k runs acting as responder in parallel, apparently with A.
(2) A protocol authenticates an *initiator* to a *responder*, whenever an agent B starts j runs of the protocol as a responder and l runs as an initiator, *all in parallel*; and completes $k \leq j$ runs of the protocol acting as responder, apparently with initiator A, then A has recently been running k runs acting as initiator in parallel, apparently with B.

It is assumed that the meaning of two terms '*starting a run*' and '*completing a run*' are different. In the next section in Definition 11, the following assertions through transition systems are formalized.

(1) An agent *starts* i runs of a protocol in parallel as an initiator (or as a responder),
(2) An agent *completes* i runs of a protocol in parallel as an initiator (or as a responder) apparently with another agent.

To clarify the definition, the following examples are stated. We use the standard notations in the literature: it is referred to the shared key between A and B by k_{AB}, and a nonce is denoted by N or n.

Example 1 Assume the following faulty authentication protocol:

$$I \to R: \ \{N_I\}_{k_{RI}}$$
$$R \to I: \ N_I$$

This protocol aims to authenticate the responder R to the initiator I. In order to verify the identity of R, I sends a challenge N_I (a nonce) to R encrypted with the symmetric key k_{RI}, which is only known by I and R. If two agents A and B, with shared key k_{AB}, participate in the protocol and two sessions of the protocol are executed *in parallel*, and A plays both the initiator and the responder role, the following well-known *reflection attack* becomes possible.

$$A(I) \to E(R): \ \{N_A\}_{K_{AB}}$$

$E(I) \rightarrow A(R): \{N_A\}_{K_{AB}}$
$A(R) \rightarrow E(I): N_A$
$E(R) \rightarrow A(I): N_A$

$E(I)$ denotes the enemy (intruder) impersonating I, and $A(R)$ means agent A acts as a responder. In this attack, A completes one run as initiator and one run as responder, whereas, B does not participate in the protocol. The protocol fails the item (i) of the definition. Agent A completes one run acting as initiator apparently with responder B, and one run acting as responder with B, all in parallel. Whereas, B does not participate in the protocol.

Example 2 Consider the *challenge-response* protocol. The protocol aims both to authenticate initiator I to responder R, and to authenticate responder R to initiator I.

$I \rightarrow R: n_I$
$R \rightarrow I: \{n_I\}_{k_{RI}}.n_R$
$I \rightarrow R: \{n_R\}_{k_{RI}}$

Assume two agents A and B, with shared key k_{AB}, participate in the protocol, and two sessions of the protocol are executed *in parallel*, and B plays the responder roles. The following well known *reflection attack* becomes possible.

$\star$ $E(I) \rightarrow B(R): N_A$
$\star$ $B(R) \rightarrow E(I): \{N_A\}_{k_{AB}}.N_B$
$E(I) \rightarrow B(R): N_B$
$B(R) \rightarrow E(I): \{N_B\}_{k_{AB}}.N'_B$
$\star$ $E(I) \rightarrow B(R): \{N_B\}_{k_{AB}}$

The protocol fails the item (ii) of the definition. Agent B starts two runs as a responder, and completes one run ($\star$ lines) of the protocol acting as responder, apparently with initiator A (because of the shared key k_{AB}), whereas, A does not participate in the protocol. Note that B starts two runs as a responder in the protocol, while she completes just one run.

Example 3 Assume the *Wide Mouthed Frog* protocol.

$I \rightarrow S: I, \{t_I, R, k_{RI}\}_{k_{IS}}$
$S \rightarrow R: \{t_S, I, k_{RI}\}_{k_{RS}}$

Initiator, I, sends its name and a generated key to R via a trusted server, in this way, initiator is authenticated to responder [15]. Assume three agents A, B and D (trusted party), with shared keys k_{AD}, k_{BD} participate in the protocol. The following *multiplicity attack* is possible, whereby an enemy makes B to believe initiator, A, has established two sessions with her, when A only wanted a single session.

$A(I) \rightarrow D(S): A, \{t_A, B, k\}_{k_{AD}}$
$D(S) \rightarrow B(R): \{t_D, A, k\}_{k_{BD}}$
$E(S) \rightarrow B(R): \{t_D, A, k\}_{k_{BD}}$

The protocol fails the item (ii) of the definition. Agent B completes two runs of the protocol acting as a responder, apparently with initiator A, whereas A completes only one run.

3 Transition Systems

The notion of transition system can be considered as a fundamental notion for the description of process behavior [17]. In this section, some abstract formal definitions via transition systems are stated, and then the notion of authentication using these definitions is specified.

3.1 Preliminaries

Definition 1 *A transition system T is a quintuple $(S, A, \rightarrow, \downarrow, s_0)$ where*

- *S is a set of states,*
- *A is a set of actions containing an internal action τ,*
- *$\rightarrow \subseteq S \times A \times S$ is a set of transitions,*
- *$\downarrow \subseteq S$ is a set of successfully terminating states,*
- *$s_0 \in S$ is the initial state.*

The set $\twoheadrightarrow \subseteq S \times A^* \times S$ shows *generalized transitions* of T. A state $s \in S$ is called *reachable* state of T if there is $\sigma \in A^*$ such that $s_0 \overset{\sigma}{\twoheadrightarrow} s$. The set of all reachable states of a transition system T is denoted by $reach(T)$. We let $act(T) = \{a \in A \mid \exists s, s' \in reach(T)\ (s, a, s') \in \rightarrow\}$. In the sequel, it is assumed that every transition system T is connected, i.e., $reach(T) = S$, and $act(T) = A$. If S and A are finite, T is called a finite transition system.

For every $s \in S$, we let $Trace = \{\sigma \in A^* \mid \exists s' \in S\ s_0 \overset{\sigma}{\twoheadrightarrow} s'\}$. If there exists $n \in \mathbb{N}$ such that $\forall \sigma \in Trace\ (|\sigma| \leq n)$, where $|\sigma|$ is the length of the sequence σ, then T is called a finite-depth transition system. If for every $s \in S$, $\{(s, a, s') \in \rightarrow \mid a \in A, s' \in S\}$ is finite, then T is called a finite-branching transition system. The notation τ refers to the *silent action*.

Notation 1 *The notation $s \xrightarrow{a} s'$ denotes $(s, a, s') \in \rightarrow$.*

Proposition 1 *If a transition system T is both finite-depth and finite-branching then it is finite.*

Proof. It is straightforward. ∎

Definition 2 *Let $T = (S, A, \rightarrow, \downarrow, s_0)$ be a transition system. Then T is deterministic if the following condition holds. Whenever $s_0 \overset{\sigma}{\twoheadrightarrow} s$ and $s_0 \overset{\sigma}{\twoheadrightarrow} s'$, then $s = s'$.*

Definition 3 *Let A be a set of actions. A communication function on A is a partial function $\gamma : A \times A \rightarrow A$ such that for any $a, b \in A$: $\gamma(\tau, a)$ is not defined, and if $\gamma(a, b)$ is defined, then $\gamma(b, a)$ is defined and*

$\gamma(a,b)=\gamma(b,a)$. *The image of* γ *is shown by* C_γ, *and we define* $H_\gamma = A - C_\gamma$. *It is assumed that if* $\gamma(a,b)$ *is defined then both* $a, b \in H_\gamma$.

See the opening part of section 4, where the above definition is used.

Definition 4 (Parallel composition). *Let* $T = (S, A, \rightarrow, \downarrow, s_0)$ *and* $T' = (S', A', \rightarrow', \downarrow', s'_0)$ *be two transition systems, and* γ *a communication function on a set of actions that includes* $A \cup A'$. *The parallel composition of* T *and* T' *under* γ, *written* $T \parallel T'$, *is the transition system* $(S'', A'', \rightarrow'', \downarrow'', s''_0)$ *where*

- $S'' = S \times S'$,
- $A'' = A \cup A' \cup \{\gamma(a,a') \mid a \in A, a' \in A'\}$
- $\rightarrow''$ *is the smallest subset of* $S'' \times A'' \times S''$ *such that:*
  - *if* $s_1 \xrightarrow{a} s_2$ *and* $s' \in S'$, *then* $(s_1, s') \xrightarrow{a} ''(s_2, s')$,
  - *if* $s'_1 \xrightarrow{b} s'_2$ *and* $s \in S$, *then* $(s, s'_1) \xrightarrow{b} ''(s, s'_2)$,
  - *if* $s_1 \xrightarrow{a} s_2$, $s'_1 \xrightarrow{b} s'_2$, *and* $\gamma(a,b)$ *is defined, then* $(s_1, s'_1) \xrightarrow{\gamma(a,b)} ''(s_2, s'_2)$,
- $\downarrow''=\downarrow \times \downarrow'$,
- $s''_0 = (s_0, s'_0)$.

Definition 5 (Encapsulation). *Let* $T = (S, A, \rightarrow, \downarrow, s_0)$ *be a transition system. Let* H *be a set of actions. The encapsulation of* T *with respect to* H, *written as* $\delta_H(T)$ *is the transition system* $(S', A', \rightarrow', \downarrow', s'_0)$, *where:*

- $S' = S$, $A' = A$, $\downarrow'=\downarrow$, $s'_0 = s_0$ *and*
- $\rightarrow'=\rightarrow \cap (S \times (A - H) \times S)$.

If we assume two processes T_1 and T_2, and execute them in parallel, then for $H = A - C_\gamma$, the encapsulation of the process $T_1 \parallel T_2$ causes the processes to communicate. That is, the difference between $T_1 \parallel T_2$ and $\delta_H(T_1 \parallel T_2)$ is that in the second process, there are only communication actions.

Definition 6 (Abstraction). *Let* $T = (S, A, \rightarrow, \downarrow, s_0)$ *be a transition system. Let* I *be a set of actions. The abstraction of* T *with respect to* I, *written* $\tau_I(T)$ *is the transition system* $(S', A', \rightarrow', \downarrow', s'_0)$ *where*

- $S' = S$, $A' = A$, $\downarrow'=\downarrow$, $s'_0 = s_0$ *and*
- $\rightarrow'$ *is the smallest subset of* $S \times A' \times S$ *such that:*
  - *if* $s_1 \xrightarrow{a} s_2$ *and* $a \in I$, *then* $s_1 \xrightarrow{\tau} 's_2$,
  - *if* $s_1 \xrightarrow{a} s_2$ *and* $a \notin I$, *then* $s_1 \xrightarrow{a} 's_2$.

Proposition 2 *If* T *and* T' *are two transition systems, and* T *is finite-depth, then* $\delta_{H_\gamma}(T \parallel T')$ *is finite-depth.*

Proof. It is straightforward. ∎

Definition 7 *Assume two transition systems* $T = (S, A, \rightarrow, \downarrow, s_0)$ *and* $T' = (S', A', \rightarrow', \downarrow', s'_0)$, *and for* $s \in S$, *let* $l(s) = \{(s,a,t) \in \rightarrow \mid a \in A, t \in S\}$, *and for every* $s' \in S'$, $l'(s') = \{(s',b,t') \in \rightarrow' \mid b \in A', t' \in S'\}$. *The transition systems* T, T' *are called* communication finite-branching *with respect to communication function* γ, *if for any* $(s,s') \in S \times S'$ *the set* $\{((s \xrightarrow{a} t), (s' \xrightarrow{b} 't)) \in l(s) \times l(s') \mid \gamma(a,b)\ is\ defined\}$ *is finite.*

Proposition 3 *If two transition systems* $T = (S, A, \rightarrow, \downarrow, s_0)$ *and* $T' = (S', A', \rightarrow', \downarrow', s'_0)$ *are* communication finite-branching *with respect to a communication function* γ, *then* $\delta_{H_\gamma}(T \parallel T')$ *is finite-branching.*

Proof. It is straightforward. ∎

In the following, a behavioral semantics is defined. It used in order to formalize the notion of authentication.

Definition 8 *A* rooted branching similarity $\ll$ *between two transition systems* T_1, T_2, *is a subset of* $S_1 \times S_2$, *with the following conditions:*

- *Transfer conditions:*

(1) if $s \ll t$ *and* $s \xrightarrow{a} s'$, *then either*
  - $a = \tau$ *and* $s' \ll t$,
  - *or,*
  - *there exist* t', t'' *and a sequence of (zero or more)* τ*-transitions* $t \xrightarrow{\tau} \cdots \xrightarrow{\tau} t'$ *such that* $s \ll t'$ *and* $t' \xrightarrow{a} t''$ *with* $s' \ll t''$.

(2) whenever $s \ll t$ *and* $s \downarrow$, *then there exists* t' *and a sequence of (zero or more)* τ*-transitions* $t \xrightarrow{\tau} \cdots \xrightarrow{\tau} t'$ *such that* $t' \downarrow$, *and* $s \ll t'$,

(3) $(s_0)_1 \ll (s_0)_2$,

- *Rooted conditions:*

(4) the pair $((s_0)_1, (s_0)_2)$ *satisfies the* rooted conditions, *that is,*
  - *whenever* $(s_0)_1 \xrightarrow{a} s$, *then there exists* t *such that* $(s_0)_2 \xrightarrow{a} t$ *and* $s \ll t$,
  - *whenever* $(s_0)_1 \downarrow$ *then* $(s_0)_2 \downarrow$.

We say the transition system T_2 simulates T_1, $T_1 \lll T_2$, *if there exists a* rooted branching similarity $\ll$ *between* T_1 *and* T_2.

Proposition 4 *If two transition systems* T_1 *and* T_2 *are finite, then* $T_1 \lll T_2$ *is decidable.*

Proof. It is straightforward. ∎

3.2 Authentication via Transition Systems

In this part, the notion of authentication is specified using transition systems. A finite set $P = \{I, R, p_1, p_2, \cdots, p_n\} \cup \{e\}$ of agents is assumed consisting of initiator (I), responder (R), honest third parties $(p_1, p_2, \cdots, p_n)$, and enemy (e).

Definition 9 *A* P*-transition system* T *is a quintuple* $(S, \mathbf{A}_P, \rightarrow, \downarrow, s_0)$ *where*

- S *is a set of states,*
- $\mathbf{A}_P$ *is a set of actions with an ownership function* $O : \mathbf{A}_P \to \mathcal{P}(P)$*, such that* $O(a)$ *is the set of participants who can perform* a*,*
- $\to \subseteq S \times \mathbf{A}_P \times S$ *is a set of transitions,*
- $\downarrow \subseteq S$ *is a set of successfully terminating states,*
- $s_0 \in S$ *is the initial state.*

For each participant $p \in P$*, the set* $O^{-1}(p) = \{a \in \mathbf{A}_P \mid p \in O(a)\}$ *is the set of actions performed by participant* p *in the* P*-transition system* T*.*

Definition 10 (Agent's Scope). *The* scope *of a participant* $p \in P$ *(agent's scope) to a* P*-transition system* T *is the* P*-transition system* $Sc(p)T = \tau_{\mathbf{A}-O^{-1}(p)}T$*. That is, all actions which do not belong to* $O^{-1}(p)$ *are hidden.*

The scope of a participant p to a P-transition system T is just those actions that p performs, and occurrence of other's action is unknown and hidden for the participant p. So if for two P-transition systems T and T', $Sc(p)T'$ *simulates* $Sc(p)T$, $Sc(p)T' \lll Sc(p)T$, then p is persuaded that T' simulates T. The notion of agent's scope plays an essential part in defining authentication in the next definition.

Definition 11 (Authentication).

(1) A P*-protocol* PR *consists of two things:*
  (a) P*-transition systems* $T_I, T_R, T_{p_1}, T_{p_2}, \cdots, T_{p_n}$*; where* I *is initiator,* R *is responder and* $p_1, p_2, \cdots, p_n$ *are honest third parties in the protocol,*
  (b) A partial communication function $\gamma : \mathbf{A}_P \times \mathbf{A}_P \to \mathbf{A}_P$ *satisfying Definition 3.*

(2) Assume E *is a transition system that for any action* a *occurred in it* $e \in O(a)$*, the enemy model. Assume also* A *and* B *are two agents who use the protocol. The transition system:*

$$\mathbf{T} = \prod^{i_{A(I)}} T_{A(I)} \parallel \prod^{i_{A(R)}} T_{A(R)} \parallel \prod^{i_{B(R)}} T_{B(R)} \parallel \prod^{i_{B(I)}} T_{B(I)} \parallel \prod^{i_{p_1}} T_{p_1} \parallel \prod^{i_{p_2}} T_{p_2} \parallel \cdots \parallel \prod^{i_{p_n}} T_{p_n} \parallel E,$$

is called a parallel-instance *of* PR *in the presence of the enemy* E*, where* $\prod^i T$ *is an abbreviation for* $T \parallel T \parallel \cdots \parallel T$*,* i *times, and* $A(I)$ *means agent* A *is impersonating initiator* I*. Let:*

$$length(\mathbf{T}) = max\{i_{A(I)}, i_{A(R)}, i_{B(R)}, i_{B(I)}, i_{p_1}, \ldots, i_{p_n}\}.$$

Then:

(1) Agent A *starts* $i_{A(I)}$ *runs as an initiator and* $i_{A(R)}$ *runs as a responder in* $\mathbf{T}$*.*

(2) Agent B *starts* $i_{B(I)}$ *runs as an initiator and* $i_{B(R)}$ *runs as a responder in* $\mathbf{T}$*.*

(3) Agent A *completes* $k \leq i_{A(I)}$ *runs as an initiator, apparently with* B*, in* $\mathbf{T}$*, whenever* $Sc(A(I))\delta_{H_\gamma}(\prod^k T_{A(I)} \parallel \prod^k T_{B(R)} \prod^k T_{p_1} \parallel \prod^k T_{p_2} \parallel \cdots \parallel \prod^k T_{p_n}) \lll Sc(A(I))\delta_{H_\gamma}(\mathbf{T})$ *where* δ*, encapsulation, forces the processes to communicate. In the above formula, the left part of the simulation,* $Sc(A(I))\delta_{H_\gamma}(\prod^k T_{A(I)} \parallel \prod^k T_{B(R)} \prod^k T_{p_1} \parallel \prod^k T_{p_2} \parallel \cdots \parallel \prod^k T_{p_n})$*, shows the scope of the agent* A*, when she participates* k *times in the protocol as initiator, all in parallel, and the protocol is completed correctly as the designer desired, where the agent* B *completes* k *runs as a responder with* A*. The right part shows the scope of the agent* A *of* $\mathbf{T}$ *, where she participates as an initiator in the presence of the enemy* E*. Now, if the left part simulates the right part, the agent* A *is persuaded that at least she could complete* k *runs as an initiator, apparently with* B *in* $\mathbf{T}$*.*

(4) Agent B *completes* $k \leq i_{B(R)}$ *runs as a responder, apparently with* A*, in* $\mathbf{T}$*, whenever* $Sc(B(R))\delta_{H_\gamma}(\prod^k T_{A(I)} \parallel \prod^k T_{B(R)} \prod^k T_{p_1} \parallel \prod^k T_{p_2} \parallel \cdots \parallel \prod^k T_{p_n}) \lll Sc(B(R))\delta_{H_\gamma}(\mathbf{T})$*.*

The protocol PR fails to authenticate initiator A to responder B; if there exists a parallel-instance $\mathbf{T}$, $j, l, k \in \mathbb{N}, k \leq j$, such that A starts j runs as an initiator and l runs as a responder, in $\mathbf{T}$, and completes k runs as an initiator apparently with B, in $\mathbf{T}$, whereas, B does not complete k runs as a responder apparently with A, in $\mathbf{T}$. Then the *parallel-instance* $\mathbf{T}$ is called an *initiator-fail*. The enemy E persuades A to believe that she has completed k runs with B.

The protocol PR fails to authenticate responder B to initiator A; if there exists a parallel-instance $\mathbf{T}$, $j, l, k \in \mathbb{N}, k \leq j$, such that B starts j runs as a responder and l runs as an initiator, in $\mathbf{T}$, and completes k runs as a responder apparently with A, in $\mathbf{T}$, whereas, A does not complete k runs as an initiator apparently with B, in $\mathbf{T}$. Then the *parallel-instance* $\mathbf{T}$ is called a *responder-fail*. The enemy E persuades B to believe that she has completed k runs with A.

The protocol PR is called secure, if it has no *initiator-fail* and no *responder-fail*. It is t-secure, if it has no *initiator-fail* and no *responder-fail* with length less than t.

In the next section, it is explained how to express security protocol by transition systems. We use process algebra [12], where it is provided with some primitives for manipulating messages.

4 The Model

In this section, we describe a language which is applied for the specification of security protocols. For convenience, we refer to the model as SPA standing for *secure process algebra*. It is not claimed that the SPA is a new process algebra; it can be translated into the μCRL [13], a process algebra which can handle data. We take advantage of the terms of SPA to formalize the notion of agent's recognizability.

4.1 The Syntax

SPA syntax is based on the following elements.

- A *finite* set M of basic messages containing participants' ID, Nonce, cryptographic keys and primitive messages. The set $\mathcal{M}$ of all messages is defined as the least set such that $M \subseteq \mathcal{M}$, and for each $m, m', k \in \mathcal{M}$, we let (m, m') (concatenation) and $\{m\}_k$ (encryption) also belong to $\mathcal{M}$.
- Actions: $Id.snd(Id, d)$ (sending action), $Id.rcv(Id, d)$ (receiving action), $comm(Id, Id, d)$ (communication action), and τ (internal action), where $Id \in ID, d \in \mathcal{M}$.
- A set Var-m $= x, y, z, ..$ of variables for messages, a set of Var-i $= i, j, \cdots$ of variables for ID.
- A communication function γ,

$$\gamma(Id_1.snd(Id_2, d), Id_2.rcv(Id_1, d)) = comm(Id_1, Id_2, d).$$

- An ownership function $O : A \to \mathcal{P}(ID)$ as follows.

$$O(Id_1.snd(Id_2, d)) = \{Id_1\},$$
$$O(Id_1.rcv(Id_2, d)) = \{Id_1\},$$
$$O(comm(Id_1, Id_2, d)) = \{Id_1, Id_2\}$$
$$O(\tau) = ID.$$

In the sequel, we consider two projection maps $\Pi_1, \Pi_2 : \mathcal{M} \to \mathcal{M}$ as follows: for any pair $m = (m_1, m_2)$, $\Pi_1(m) = m_1$ and $\Pi_2(m) = m_2$. We also consider for each key k, a decryption map $.\{-\}_{k^{-1}} : \mathcal{M} \to \mathcal{M}$ which for any encrypted message $m = \{m_1\}_k$, $\{m\}_{k^{-1}} = m_1$.

The set of message-formulas is simply defined inductively as follows:

(1) all $m \in \mathcal{M}$ and all variables $x, y, \cdots$ for messages are message-formulas,
(2) If c, d are message formulas, and $k \in KEY$ then (c, d) and $\{c\}_k$ are message-formulas.

Action formulas, like $Id.snd(Id, C[x, y, .., z])$, $Id.rcv(Id, C[x, y, .., z])$, $comm(Id, Id, C[x, y, .., z])$, where $Id \in ID$ and $C[x, y, .., z]$ is a message formula, are called *open actions* due to variables $x, y, \cdots, z$.

Definition 12 Types *of messages are defined inductively as follows:*

(1) The sets ID (names of participants), NC (nonce), KEY (cryptographic keys), PM (primitive messages) and $\{m\}$, $m \in \mathcal{M}$, are (finite) types.
(2) The sets $\mathcal{M}$ and $E\mathcal{M} = \{\{d\}_k \mid d \in \mathcal{M}, k \in KEY\}$, the set of all encrypted messages, are infinite *types.*
(3) If W, V are types and $k \in KEY$, (W, V) and $\{W\}_k$ are types.
(4) If W, V are finite types then (W, V) and $\{W\}_k$ are finite types.
(5) All types are defined through the above items.

Note that finite types have many finite elements.

Definition 13 *Let S be a subset of $\mathcal{M}$. A message d is derivable form S, denoted by $S \vdash d$, if d can be deduced from S by the following inference rules.*

$$\frac{m \quad m'}{(m, m')} \qquad \frac{(m, m')}{m} \qquad \frac{(m, m')}{m'}.$$

The set $\mathcal{L}$ of SPA's terms (processes) contains the process terms in [12]:

a constant 0, denoting *inaction*,
for each action a, a unary operator $a.-$, denoting *action prefix*. If P is a term, $a.P$ executes action a and then proceeds as P,
binary operators $+$, $.$, $\|$ respectively denoting alternative, sequential, and parallel compositions,
unary operators δ_H and τ_I, for $I, H \subseteq A$.

In addition to these syntaxes, in order to model message handling, the SPA also has the following three extra new terms.

(1) $a(x).P$, where P is a term, a an action, and x is variable.
(2) $[x : W]P(x)$, where P is a term with free variable x, W is a type and x is a variable.
(3) $\langle S \vdash x\rangle P(x)$, where P is a term with free variable x, and S a subset of $\mathcal{M}$.

Variable x is free in term $a(x).P$, If x is free in term Q, then it is free in all terms obtained by the signature, i.e., x is free in $Q.P$, $P.Q$, $Q + P$, $Q \| P$, $\tau(Q)$ (abstraction), and $\delta(Q)$ (encapsulation). Terms $[x : W]P(x)$ and $\langle S \vdash x\rangle P(x)$ bound x, and x is not free in them. A *process* is a term with no free variable. Informally, the process $[x : W]P(x)$ behaves as the possible choice between $P(d)$ for any message $d \in W$,

and the process $\langle S \vdash x\rangle P(x)$ behaves as the possible choice between $P(d)$ for any message $S \vdash d$.

4.2 The Operational Semantics

In order to model message handling, we added some new terms as above. Besides the following familiar operational semantics of process algebra [12],

$$\frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'} \qquad \frac{Q \xrightarrow{a} Q'}{P + Q \xrightarrow{a} Q'}$$

$$\frac{P \downarrow}{P + Q \downarrow} \qquad \frac{Q \downarrow}{P + Q \downarrow}$$

$$\frac{P \downarrow \qquad Q \downarrow}{P.Q \downarrow} \qquad \frac{P \xrightarrow{a} P'}{P.Q \xrightarrow{a} P'.Q}$$

$$\frac{P \downarrow \qquad Q \xrightarrow{a} Q'}{P.Q \xrightarrow{a} Q'} \qquad \frac{P \downarrow}{\delta_H P \downarrow}$$

$$\frac{P \xrightarrow{a} P' \qquad a \notin H}{\delta_H P \xrightarrow{a} \delta_H P'} \qquad \frac{P \downarrow \qquad Q \downarrow}{P \parallel Q \downarrow}$$

$$\frac{P \xrightarrow{a} P'}{P \parallel Q \xrightarrow{a} P' \parallel Q} \qquad \frac{Q \xrightarrow{a} Q'}{P \parallel Q \xrightarrow{a} P \parallel Q'}$$

$$\frac{P \downarrow}{\tau_I P \downarrow} \qquad \frac{P \xrightarrow{a} P' \qquad a \notin I}{\tau_I P \xrightarrow{a} \tau_I P'}$$

$$\frac{P \xrightarrow{a} P' \qquad a \in I}{\tau_I P \xrightarrow{\tau} \tau_I P'} \qquad a.P \xrightarrow{a} P$$

$$\frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{b} Q' \qquad \gamma(a,b) = c}{P \parallel Q \xrightarrow{c} P' \parallel Q'} \qquad 1 \downarrow$$

The SPA's operational semantics contains two extra new rules shown below.

$$\mathrm{R'1}: \frac{P(d) \xrightarrow{a} Q \qquad d \in W}{[x:W]P(x) \xrightarrow{a} Q}$$

$$\mathrm{R'2}: \frac{P(d) \xrightarrow{a} Q \qquad S \vdash d}{\langle S \vdash d\rangle P(x) \xrightarrow{a} Q}$$

In this way, the meaning of the terms is clarified. The behavior of a term is formally described by means of the *transition system* induced by the inference rules of the operational semantics. One may observe that the similarity relation $\lll$ is congruence with respect to functions $.$, $+$, $\|$, τ_I, and δ_H.

Theorem 1 *The similarity relation $\lll$ is congruence in SPA.*

Proof. The proof is straightforward. Proving the congruence for the functions $a.$ (for $a \in A$), $+$, $.$, $\|$, τ_I, and δ_H is the same as it is for the rooted branching bisimilarity in the literature. For the SPA's new operators, assume $p(d) \lll p'(d)$ for all $d \in W$ (or $S \vdash d$), where W is a type (S is a subset of $\mathcal{M}$). Then one may easily verify that $[x:W]p(x) \lll [x:w]p'(x)$ ($\langle S \vdash d\rangle p(x) \lll \langle S \vdash d\rangle p'(x)$). ■

The following examples show how security protocols can be specified via SPA.

Example 4 We give a simple example of SPA protocol specification. Assume the protocol in the Example 1:

$$I \to R: \ \{N\}_k$$
$$R \to I: \ N$$

it can be specified as the following SPA process. $T = T_I \parallel T_R$,

$$T_I = [x:NC]I.snd(R,\{x\}_k).[y:\{x\}]I.rcv(R,y).0$$
$$T_R = [x:\{NC\}_k]R.rcv(I,x).R.snd(I,\{x\}_{k^{-1}}).0$$

According to the protocol, the initiator I chooses a nonce x, sends encrypt of x to R, and expects to receive a message y from R if y is equal to x. Also, the responder R expects to receive an encrypted nonce x from I, and sends the decryption of x to I. Note that, since R has the key k (k is shared between I and R according to the assumptions of the protocol), she can detect if a message is an encrypted nonce or not; thus, she just receives messages from the type $\{NC\}_k$.

Example 5 Assume

$$S \to I: \{(N_1, \{N_2\}_{k_{SR}})\}_{k_{SI}}$$

it can be specified as follows:

$$T = T_S \parallel T_I$$
$$T_S = [x:NC][y:NC]S.snd(I,\{(x,\{y\}_{k_{SR}})\}_{k_{SI}}).0$$
$$T_I = [x:\{(NC,E\mathcal{M})\}_{k_{SI}}]I.rcv(S,x).0$$

The server S pairs a nonce x and an encrypted nonce $\{y\}_{k_{SR}}$. Then, she sends the encrypted form of the pair with key k_{SI} to the participant I. The initiator I receives x which she expects x to be an encrypted (with key k_{SI}) pair of a nonce and an encrypted message. Since I does not have access to k_{SR}, she cannot distinguish between an encrypted message with k_{SR} and other messages in $E\mathcal{M}$. Thus, I admits reception

of any message of type $\{(NC, E\mathcal{M})\}_{k_{SI}}$.

Definition 14 (Agent's Recognizability power). *For a participant p, the collection of p-recognizable types is defined as follows:*

(1) ID, NC, KEY, PM, $E\mathcal{M}$ and $\{m\}$ $(m \in \mathcal{M})$ are p-recognizable.
(2) If W, V are p-recognizable then (W, V) is p-recognizable.
(3) If W, V are p-recognizable and p has access to key k, e.g., k is a shared key between p and another participant, then $\{W\}_k$ is p-recognizable.
(4) All p-recognizable types are defined through 1, 2 and 3.

Note that, we assume a participant is able to recognize that whether a message is an encrypted one or not, i.e., she is able to decide membership of $E\mathcal{M}$, though she is not able to recognize the key that the encrypted message has been encrypted with, except she knows the key. The notion of agent's recognizability plays a significant part in verifying the authentication for a protocol.

Example 6 Assume two types $\{(NC, E\mathcal{M})\}_{k_{SI}}$, $\{(NC, \{NC\}_{k_{SR}})\}_{k_{SI}}$. The first one is I-recognizable, i.e., if I receives a message, she is able to detect whether the message belongs to $\{(NC, E\mathcal{M})\}_{k_{SI}}$ or not. Whereas, the second one is not I-recognizable, since I does not have access to the key k_{SR}. The membership status of elements of $\{(NC, \{NC\}_{k_{SR}})\}_{k_{SI}}$ is not decidable for I. So, if I receives a message from $\{(NC, \{NC\}_{k_{SR}})\}_{k_{SI}}$, since she only has the key k_{SI}, all she can detect is that she has received an encrypted (with key k_{SI}) pair of a nonce and an encrypted message whose encryption key she is not sure about. Therefore, if I wants to admit reception of elements of the type $\{(NC, \{NC\}_{k_{SR}})\}_{k_{SI}}$, she has to admit reception of elements of the first type $\{(NC, E\mathcal{M})\}_{k_{SI}}$.

Definition 15 *A* sequential *SPA process is defined inductively as follows:*

(1) 0 is a sequential *term.*
(2) If a is an action (or an action with free variable), and P is a sequential *term, then $a.P$ is sequential.*
(3) If P is a sequential *term and x a free variable in it, then $[x : W]P$ is* sequential.
(4) All sequential *terms are defined through 1,2,3.*

A sequential *process is a* sequential *term without any free variable.*

Definition 16 *A SPA process T is called* legal, *if it satisfies the following properties.*

(1) If a receiving action $p_1.rcv(p_2, \text{“}Message\text{”})$ appears in T, then the “$Message$” must be just a variable. For example, $[x : NC]p_1.rcv(p_2, \{x\}_k)$, $[x : NC][y : ID]p_1.rcv(p_2, (x, y))$ and $p_1.rcv(p_2, N)$ where N is a nonce, are not legal. The legal forms of them respectively are $[x : \{NC\}_k]p_1.rcv(p_2, x)$, $[x : (NC, ID)]p_1.rcv(p_2, x)$ and $[x : \{N\}]p_1.rcv(p_2, x)$.
(2) If a receiving action $p_1.rcv(p_2, x)$ appears in T, then the variable x must belong to a p_1 recognizable type, i.e., if $[x : W]p_1.rcv(p_2, x)$ then W must be p_1 recognizable.
(3) T is a term with no free variable, i.e., a process.

Now, we are equipped to express the formalization of security protocols using SPA syntax.

Definition 17 *For a given security protocol with participants $\{I, R, p_1, p_2, \cdots, p_n\}$, a SPA specification of the protocol consists of* sequential legal *SPA processes T_ps, where $p \in \{I, R, p_1, p_2, \cdots, p_n\}$.*

In the next section, the security protocols argued in Examples 1, 2, 3 are formalized.

5 Modelling Some Security Protocols

In this section, we model some security protocols using SPA. We also identify the enemy model and specify it by SPA.

5.1 Protocol 1.

Assume the authentication protocol:

$$I \to R: \ \{N_I\}_{k_{RI}}$$
$$R \to I: \ N_I$$

introduced in Example 1. It is specified through the following SPA processes T_I, T_R:

$$T_I = [x : NC]I.snd(R, \{x\}_{k_{RI}}).[y : \{x\}]I.rcv(R, y).0$$
$$T_R = [x : \{NC\}_{k_{RI}}]R.rcv(I, x).\ R.snd(I, \{x\}_{k_{RI}^{-1}}).0$$

Initiator chooses an x from type NC, encrypts x, and sends the encrypted message to R, and only receives the same x whose encryption she sent to R; she does not receive any other message. Responder receives an encrypted nonce with the key k_{RI}, i.e., a message that belongs to the type $\{NC\}_{k_{RI}}$ (responder, according to the protocol expects an encrypted nonce with the shared key k_{RI}, and she just performs the receiving action for the message in the very same type), decrypts it and sends the result to I. Both T_I and T_R are *sequential* and *legal*.

5.2 Protocol 2.

Assume the *challenge-response* protocol argued in Example 2.

$I \to R : n_I$
$R \to I : \{n_I\}_{k_{RI}}.n_R$
$I \to R : \{n_R\}_{k_{RI}}$

It is specified through the following SPA processes T_I, T_R:

$$T_I = [x : NC]I.snd(R, x).[y : (\{\{x\}_{k_{RI}}\}, NC)]I.$$
$$rcv(R, y).\ I.snd(R, \{\Pi_2(y)\}_{k_{RI}}).0$$
$$T_R = [x : NC]R.rcv(I, x).\ [y : NC]R.$$
$$snd(I, (\{x\}_{k_{RI}}, y)).\ [z : \{\{y\}_{k_{RI}}\}]R.rcv(I, z).0$$

Initiator chooses x from type NC and sends it to R, then receives a message if it belongs to the type $(\{\{x\}_{k_{RI}}\}, NC)$, and finally sends the encryption of the first part of y with key k_{RI} to R. Responder receives a message x from type NC, encrypts it and concatenates it with a nonce y, sends the whole to I, and receives the encryption of the variable y with key k_{RI}. Both T_I and T_R are *sequential* and *legal*.

5.3 Protocol 3.

Assume the *Wide Mouthed Frog* protocol.

$I \to S :\ I, \{t_I, R, k_{RI}\}_{k_{IS}}$
$S \to R :\ \{t_S, I, k_{RI}\}_{k_{RS}}$

Unfortunately, SPA is not rich to formalize *timestamps*. Certainly, it is easy to add a type message for timestamps to the syntax, but it is not enough and it is needed to assume a clock process which runs in parallel with protocols. This is because, when an agent wants to verify that a received timestamp is fresh, or wants to generate a fresh timestamp, she needs a clock to communicate with it. For simplicity, in this paper, we do not consider timestamps, and instead of the *Wide Mouthed Frog* protocol, the following faulty protocol is specified:

$$I \to S :\ I, \{R, k_{RI}\}_{k_{IS}}$$
$$S \to R :\ \{I, k_{RI}\}_{k_{RS}}$$
$$T_I = [x : KEY]I.snd(S, \{(R, x)\}_{k_{IS}}).0$$
$$T_S = [x : \{(ID, KEY)\}_{k_{IS}}]S.rcv(I, x).$$
$$S.snd(R, \{(I, \Pi_2(\{x\}_{k_{IS}^{-1}}))\}_{k_{RS}}).0$$
$$T_R = [x : \{(ID, KEY)\}_{k_{RS}}]R.rcv(S, x).0$$

The three specifications T_I, T_R and T_S are *sequential* and *legal*.

5.4 Enemy (intruder model)

Intuitively, an enemy can be thought as a process which runs in parallel to a protocol. If a protocol is used for security purposes, this postulates an enemy against which the protocol is secure. The standard enemy for formal analysis of security protocols was introduced by Dolev and Yao in 1983, and is commonly known as the Dolev-Yao intruder model. It is a very strong enemy, that all messages sent from any honest principal to any other must pass through the enemy. The enemy can read, alter, and redirect any and all messages. However, encryption is treated as a black box. The enemy can only decrypt a message if she has the right keys. For simplicity, we assume a week enemy that has no key, and can only concatenate messages and unfold concatenations. The enemy is modeled by the following SPA process.

$$E(S) = \sum_{p,q \in ID}[x : \mathcal{M}]e(p).rcv(q, x).E(S \cup \{x\}) +$$
$$\sum_{p,q \in ID}\langle S \vdash y\rangle e(p).snd(q, y).E(S),$$
$$initial : E(S_0),$$

where $S_0 \subseteq \mathcal{M}$, a finite subset of $\mathcal{M}$, is the initial knowledge of enemy, and $e(p)$ denotes the enemy impersonating a participant p.

Note that, $\sum$ is not a signature of the syntax of SPA; since ID is finite, for convenience, we used it for alternative composition. Enemy starts with an initial knowledge S_0 a subset of $\mathcal{M}$, it is assumed that this initial knowledge is finite. Impersonating some participants, she receives some new messages from other participants and increases her knowledge. She also may impersonate some participants and send some messages to other participants. Note that the enemy just can unfold pairs and concatenate messages, and she cannot encrypt or decrypt any message. Therefore, because it is assumed S_0 is finite, the enemy always has access to a finite number of elements of $E\mathcal{M}$, i.e., encrypted messages. Since S_0 is finite, $\{y \mid S_0 \vdash y\} \cap E\mathcal{M}$ is finite. Now, assume $\{y \mid S \vdash y\} \cap E\mathcal{M}$ is finite, as enemy receives new messages, she can increase her knowledge S, but since she cannot encrypt any message, she just gains access to a message m in $E\mathcal{M}$ by receiving a message m' that $S \cup \{m'\} \vdash m$. Surely, m' could only finitely contain many encrypted messages as its components, thus $\{y \mid S \cup \{m'\} \vdash y\} \cap E\mathcal{M}$ is also finite $(\star)$.

Proposition 5 *Assume a participant p, if W is p-recognizable type then enemy always has access to a finite number of elements of W, i.e., at any state for the knowledge S of the enemy $\{y \mid S \vdash y\} \cap W$ is finite.*

Proof. If W is a finite type, then trivially $\{y \mid S \vdash y\} \cap W$ is finite. If W is infinite, according to Definition 14, the construction of W must contain $E\mathcal{M}$, i.e., either $W = E\mathcal{M}$, or W is constructed by several times applying rule 2 of Definition 14, when the type $E\mathcal{M}$ is used in the construction. In other words, since the only basic infinite p-recognizable type is $E\mathcal{M}$, any

infinite type has $E\mathcal{M}$ in it structure. So, we prove the proposition by induction on the structure of W. If W is one of the finite p-recognizable types ID, NC, KEY, PM, and $\{m\}$ ($m \in \mathcal{M}$) then the answer is trivial. If W is $E\mathcal{M}$, the above discussion ($\star$) proves the proposition. Assume for p-recognizable types V_1 and V_2, $\{y \mid S \vdash y\} \cap V_i$ is finite. Then, surely $\{y \mid S \vdash y\} \cap (V_1, V_2)$ and $\{y \mid S \vdash y\} \cap \{V_1\}_k$ are finite. ■

Now, by identifying the enemy model, we are in a position to look over the definition of authentication (Definition 11), for security protocols.

Example 7 The protocol 1 is not secure. In fact, the parallel instances

$$\mathbf{T} = T_{A(I)} \parallel T_{A(R)} \parallel E$$

is an initiator-fail. Agent A completes one run as an initiator with B in T, i.e.,

$$Sc(A(I))\delta_{H_\gamma}(T_{A(I)} \parallel T_{B(R)}) \lll Sc(A(I))\delta_{H_\gamma}(\mathbf{T}),$$

whereas, B does not complete one run as a responder apparently with A, i.e.,

$$Sc(B(R))\delta_{H_\gamma}(T_{A(I)} \parallel T_{B(R)}) \not\lll Sc(B(R))\delta_{H_\gamma}(\mathbf{T}).$$

6 Verification

In this section, we address the issue of formal verification of authentication stated in Definition 11. For a given protocol $PR = (T_I, T_R, T_{p_1}, T_{p_2}, \cdots, T_{p_n})$, and enemy E introduced in Section 5.4, to verify that if PR is secure, we must check all possible *parallel-instances* $\mathbf{T}$ of PR, stated in Definition 11. It is showon that for every *deterministic finite-sending* protocol PR, and every natural number t, it is decidable whether PR is t-secure or not. If a protocol is t-secure, then we can be sure that if no agent runs the protocol more than t times all in parallel, then the enemy cannot make any flaw. To verify that if a protocol PR is t-secure, we must show for every *parallel-instances* $\mathbf{T}$, $length(\mathbf{T}) \leq t$, $\mathbf{T}$ neither is an *initiator-fail* nor a *responder-fail.*

Definition 18 *Assume a sequential SPA process T, the process T is called finite-sending, if any state of the transition system T (the transition system induced by the inference rules of the operational semantics) can just finitely make many sending transitions, i.e., if $s \in S_T$ is a state of T, then the set $J(s) = \{s \xrightarrow{a} s' \mid a = Id_1.snd(Id_2, m), s' \in S_T, m \in \mathcal{M}, Id_1, Id_2 \in ID\}$ is finite.*

Proposition 6 *A sequential SPA process T, is finite-sending, if for any sending action*

$$Id_1.snd(Id_2, C[x])$$

in T, where $C[x]$ is a message formula with free variable x, either x belongs to a finite type, i.e., T contains an operator $[x : W]$, W a finite type, taking place before $Id_1.snd(Id_2, C[x])$, or there exists a receiving action $Id_3.rcv(Id_4, x)$ taking place before $Id_1.snd(Id_2, C[x])$ such that $Id_1 = Id_3$.

Proof. Assume for an arbitrary state $s \in S_T$, the set $J(s) = \{s \xrightarrow{a} s' \mid a = Id_1.snd(Id_2, m), s' \in S_T, m \in \mathcal{M}, Id_1, Id_2 \in ID\}$. Since T is sequential, there exists an open action $Id.snd(Id', C[x_1, x_2, \cdots, x_k])$ in the term T, where $x_1, x_2, \cdots, x_k$ are message variables, such that $s \xrightarrow{a} s' \in J(s)$ then $a = Id.snd(Id', C[m_1, m_2, \cdots, m_k])$ for some possible messages $m_1, m_2, \cdots, m_k \in \mathcal{M}$. Now according to the assumption, for each variable x_i, either x_i belongs to a finite type or there exists a receiving action $Id.rcv(Id', x_i)$ happening before $Id.snd(Id', C[x_1, x_2, \cdots, x_k])$. If x_i is a variable of a receiving action $Id.rcv(Id', x_i)$, then since it happened before reaching the state s, the value of x_i is determined and fixed. If x_i is not a variable of a receiving action, then it ranges over a finite type. Therefore, the number of sending transitions that s makes is finite. ■

Example 8 The sequential process

$$T_1 = [x : NC][y : NC]S.snd(I, \{(x, \{y\}_{k_{SR}})\}_{k_{SI}}).0$$

is finite-sending. Both variables x, y appear in the sending action that belong to the finite type NC. The sequential process

$$T_2 = [\ x : \{(NC, E\mathcal{M})\}_{k_{SI}}]I.rcv(S, x).$$
$$I.snd(S, \Pi_1(\{x\}_{k_{SI}^{-1}})).0$$

is also finite-sending, because the variable x occurs in the receiving action $I.rcv(S, x)$. The sequential process $T_3 = [x : E\mathcal{M}]I.snd(S, x).0$ is not finite-sending.

Definition 19 *A protocol $PR = (T_I, T_R, T_{p_1}, \cdots, T_{p_n})$ is called* deterministic*, if all transition systems $T_I, T_R, T_{p_1}, \cdots, T_{p_n}$ are deterministic.*

Definition 20 *A protocol $PR = (T_I, T_R, T_{p_1}, \cdots, T_{p_n})$ is called finite-sending, if all transition systems $T_I, T_R, T_{p_1}, \cdots, T_{p_n}$ are finite-sending processes.*

Remark 1 One can easily verify that all security protocols specified in Section 5, are deterministic and finite-sending. In fact, many security protocols like the Needham-Schroeder protocol (with shared keys), the Kerberos protocol, the Andrew secure RPC protocol, the Yahalom Protocol etc. [1], can be specified by deterministic finite-sending processes.

Lemma 1 *If $PR = (T_I, T_R, T_{p_1}, \cdots, T_{p_n})$ is a deterministic finite-sending protocol, then every two processes T_i and T_j, $i, j \in \{I, R, p_1, p_2, \cdots, p_n\}$, are com-*

munication finite-branching.

Proof. Since both T_i and T_j are sequential finite-sending, each state of them can only make finitely many sending transitions. On the other hand, the communication function γ, just permits communication between sending actions and receiving actions. Since both T_i and T_j are deterministic, no state of them makes a definite receiving transition more than one time, i.e., any state s of T_i or T_j, by each action $Id_1.rcv(Id_2, d)$, $Id_1, Id_2 \in ID, d \in \mathcal{M}$, makes at most one transition. Therefore T_i and T_j are communication finite-branching. ■

Lemma 2 *If* $PR = (T_I, T_R, T_{p_1}, \cdots, T_{p_n})$ *is a deterministic finite-sending protocol, then for any* T_i, $i \in \{I, R, p_1, p_2, \cdots, p_n\}$, *two processes* T_i *and* E *are communication finite-branching.*

Proof. First, note that the process E is deterministic. Now, since T_i is finite-sending and E is deterministic, the number of pairs (a, b) that can communicate ($\gamma(a, b)$ is defined, a is a sending action of T_i, and b is a receiving action of E) is finite. Therefore, if it is proved that the number of pairs (b, a) that can communicate is finite, b a receiving action of T_i, and a a sending action of E, we are done. Since T_i is a *legal* SPA process, the receiving action b is in the form $p.rcv(q, x)$, for some $p, q \in ID$, where x ranges over a p-recognizable type W. Due to the Proposition 5, E can just finitely make many sending transitions containing messages in W. Therefore, the number of pairs (b, a) that can communicate is finite. ■

Theorem 2 *If* $PR = (T_I, T_R, T_{p_1}, \cdots, T_{p_n})$ *is a deterministic finite-sending protocol, then for any parallel-instance* $\mathbf{T}$ *of* PR *and for every* $k \in \mathbb{N}$, *both processes*

$$\delta_{H_\gamma}(\prod^k T_{A(I)} \parallel \prod^k T_{B(R)} \prod^k T_{p_1} \parallel \prod^k T_{p_2} \parallel \cdots \parallel \prod^k T_{p_n})$$

and $\delta_{H_\gamma}(\mathbf{T})$ *are finite transition systems.*

Proof. Because of the last two Lemmas 1, 2, both $\delta_{H_\gamma}(\mathbf{T})$ and

$$\delta_{H_\gamma}(\prod^k T_{A(I)} \parallel \prod^k T_{B(R)} \prod^k T_{p_1} \parallel \prod^k T_{p_2} \parallel \cdots \parallel \prod^k T_{p_n})$$

are finite-branching. On the other hand, since each T_i, $i \in \{I, R, p_1, p_2, \cdots, p_n\}$, is finite-depth, due to Proposition 2, both processes are also finite-depth. Thus, by Proposition 1, they are finite. ■

Assume two agents A, B execute a protocol PR. If we want to verify that a parallel-instance $\mathbf{T}$ of PR is an *initiator-fail*, we must check for all $k \leq i_{A(I)}$ whether A completes k runs as initiator in $\mathbf{T}$, i.e.,

$$Sc(A(I))\delta_{H_\gamma}(\prod^k T_{A(I)} \parallel \prod^k T_{B(R)} \prod^k T_{p_1} \parallel \prod^k T_{p_2} \parallel \cdots \parallel \prod^k T_{p_n}) \lll Sc(A(I))\delta_{H_\gamma}(\mathbf{T}).$$

then B completes k runs as responder in $\mathbf{T}$, i.e.,

$$Sc(B(R))\delta_{H_\gamma}(\prod^k T_{A(I)} \parallel \prod^k T_{B(R)} \prod^k T_{p_1} \parallel \prod^k T_{p_2} \parallel \cdots \parallel \prod^k T_{p_n}) \lll Sc(B(R))\delta_{H_\gamma}(\mathbf{T}).$$

Also, to verify that $\mathbf{T}$ is a *responder-fail*, we must check for all $k \leq i_{B(R)}$ whether B completes k runs as responder in $\mathbf{T}$, then A completes k runs as initiator in $\mathbf{T}$. According to the Proposition 4, for two transition systems T_1, T_2, $T_1 \lll T_2$ is decidable if both T_1 and T_2 are finite. Due to the last theorem,

$$Sc(A(I))\delta_{H_\gamma}(\prod^k T_{A(I)} \parallel \prod^k T_{B(R)} \prod^k T_{p_1} \parallel \prod^k T_{p_2} \parallel \cdots \parallel \prod^k T_{p_n}),$$

$$Sc(A(I))\delta_{H_\gamma}(\mathbf{T}),$$

$$Sc(B(R))\delta_{H_\gamma}(\prod^k T_{A(I)} \parallel \prod^k T_{B(R)} \prod^k T_{p_1} \parallel \prod^k T_{p_2} \parallel \cdots \parallel \prod^k T_{p_n}),$$

and

$Sc(B(R))\delta_{H_\gamma}(\mathbf{T})$ are finite; thus, the following corollary can be deduced.

Corollary 1 *If* $PR = (T_I, T_R, T_{p_1}, \cdots, T_{p_n})$ *is a deterministic finite-sending protocol, then for any parallel-instance* $\mathbf{T}$ *of* PR, *it is decidable whether it is an* initiator-fail *(a* responder-fail*) or not.*

The following theorem asserts that if a protocol PR is deterministic finite-sending then it is decidable whether it is t-secure or not.

Theorem 3 *If* PR $= (T_I, T_R, T_{p_1}, \cdots, T_{p_n})$ *is a deterministic finite-sending protocol, then it is decidable if it is* t*-secure or not.*

Proof. To verify that a protocol is t-secure, we must check for any parallel-instance $\mathbf{T}$ of PR with $length(\mathbf{T}) < t$, whether it is an *initiator-fail* (a *responder-fail*) or not, is decidable by the last corollary. ■

7 Concluding Remarks and Related Works

We considered SPA to specify security protocols, however, as it is said, SPA is not novel and it can be translated to the μCRL language [13], the reason to consider SPA is just for convenience to take advantage of its terms to formalize the notion of agent's scope and agent's recognizability.

It is shown that it is decidable if a protocol is t-secure or not for any arbitrary $t \in \mathbb{N}$. We conjecture that

For every protocol there exists a number $M \in \mathbb{N}$, such that if for all $t < M$ the protocol is t-secure, then it is secure.

That is, if an enemy cannot obtain enough information from M parallel instances of the protocol, in order to attack it, then extra sessions do not increase the ability of the enemy. we guess that the number M is a function of the complexity of the structure of the message types appearing in the protocol.

There are lots of works in the literature which attempt to model authentication using process algebraic approaches such as [18, 19, 20, 21, 22, 23]. This work should be considered in line with works which express the correctness of protocols via the notion of *correspondence* [24, 25, 26, 27]. Woo and Lam originally proposed *correspondence assertions* for specifying authentication:

> ... when an authenticating principal finishes its part of the protocol, the authenticated principle must have been presented and participated in its part of the protocol [24].

Also, in [25, 26] a type system for correspondence assertions is presented for spi and pi calculus. The assertions are used to formalize that in all possible executions of parallel composition of two processes, P and Q, some point of execution in P must have been preceded by some point of execution in Q.

The notion of authentication, considered in this paper, is in some sense similar to the correspondence assertions, where we take advantage of the notion of rooted branching simulation for transition systems. To verify the notion of authentication, we do not apply usual model checking technics which check whether the possible states satisfy a certain property or not. The verification is simply finding a rooted branching simulation for transition systems which the existence of such a simulation is decidable. Actually, the notion of *correspondence* is indicated via the two following notions:

(1) agent's scope to the protocol (Definition 10), and
(2) agent's recognizability of types (Definition 14).

Different agents may have different scopes and different recognizability powers. These two notions may be compared with the works which aim to model the notion of authentication via epistemic logic [28, 29, 30]. In epistemic logic, different agents may have different beliefs due to their different information. Here, different agents may have different scopes of the protocol due to their various roles in the protocol, and they may have different recognizability powers due to their different access to keys and secrets. In epistemic logic, the knowledge of agent is modelled through the notion of possible worlds of Kripke models [31, 32]. A protocol is described by a sequence of epistemic actions which update the Kripke models of knowledge, and security properties (authentication, anonymity, and etc) are specified through the formulas of the epistemic logic. After updating the model using the epistemic actions, it is verified whether the formula specifying the security property is satisfied in the resulting updated models. Epistemic actions are actions where different agents may have different views on them. Therefore, the notion of agent's scope has shown itself in the notion of epistemic actions.

Acknowledgements

A grateful acknowledgement goes to the anonymous referees for their very helpful comments.

References

[1] Michael Burrows, Martín Abadi, and Roger Needham. A Logic of Authentication. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences (1934-1990)*, 426(1871):233–271, 1989.

[2] Martín Abadi and Mark R. Tuttle. A Semantics for a Logic of Authentication (Extended Abstract). In *Proceedings of the ACM Symposium of Principles of Distributed Computing*, pages 201–216. ACM Press, 1991.

[3] Paul van Oorschot. Extending Cryptographic Logics of Belief to Key Agreement Protocols. In *Proceedings of the 1st ACM conference on Computer and Communications Security (CCS '93)*, pages 232–243, New York, NY, USA, 1993. ACM.

[4] Paul F. Syverson and Paul C. van Oorschot. On Unifying Some Cryptographic Protocol Logics. In *Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy (CSFW94)*, pages 14–28, 1994.

[5] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about Belief in Cryptographic Protocols. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society, 1990.

[6] Yan Zhang and Vijay Varadharajan. A Logic for Modeling the Dynamics of Beliefs in Cryptographic Protocols. In *Proceedings of the 24th Australaian Conference on Computer Science*, pages 215–222. IEEE Computer Society Washington, DC, USA, 2001.

[7] Gavin Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR. In *Proceeidngs of Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055, pages 147–166. Springer, 1996.

[8] A. William Roscoe. Modelling and Verifying Key-Exchange Protocols using CSP and FDR. In *Proceedings of the 8th IEEE Computer Security Foundations Workshop*, pages 98–107, 1995.

[9] Peter Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe, and Bill Roscoe. *The Modelling and Analysis of Security Protocols: the CSP Approach.* Addison-Wesley, 2001.

[10] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand Spaces: Why is a Security Protocol Correct? In *Proceedingsof the IEEE Symposium on Security and Privacy*, pages 160–171, 1998.

[11] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand Spaces: Proving Security Protocols Correct. *Journal of Computer Security*, 7(2-3):191–230, 1999.

[12] Wan Fokkink. *Introduction to Process Algebra*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2000.

[13] Wan Fokkink. *Modelling Distributed Systems*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2007.

[14] Gavin Lowe. A Hierarchy of Authentication Specifications. In *Proceedings of 10th IEEE Computer Security Foundations Workshop*, volume 15. IEEE, 1997.

[15] Gavin Lowe. A Family of Attacks upon Authentication Protocols. Technical Report 1997/5, Department of Mathematics and Computer Science, University of Leicester, England, 1997.

[16] Rasool Ramezanian. Authentication in Parallel Multiple-Sessions Execution of Protocols. In *Proceedings of the 4th Iranian Society of Cryptology Conference*, 2007.

[17] C.A. Middelburg and Michel A. Reniers. Introduction to Process Theory, 2004. Technische Universiteit Eindhoven.

[18] Stefan Blom, Jan Friso Groote, Sjouke Mauw, and Alexander Serebrenik. Analysing the BKE-security Protocol with μCRL. *Electronic Notes in Theoretical Computer Science*, 139(1):49–90, 2005.

[19] Cas J. F. Cremers, Sjouke Mauw, and Erik P. de Vink. A Syntactic Criterion for Injectivity of Authentication Protocols. *Electronic Notes in Theoretical Computer Science*, 135(1):23–38, 2005. Proceedings of the Second Workshop on Automated Reasoning for Security Protocol analysis (ARSPA 2005).

[20] Cas J. F. Cremers, Sjouke Mauw, and Erik P. de Vink. Injective Synchronisation: An Extension of the Authentication Hierarchy. *Theoretical Computer Science*, 367(1-2):139–161, 2006. Automated Reasoning for Security Protocol Analysis, Automated Reasoning for Security Protocol Analysis.

[21] Mohammad Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Fair Exchange Protocols*. PhD thesis, Vrije Universitei, Amsterdam, 2008.

[22] Riccardo Focardi and Roberto Gorrieri. A Classification of Security Properties for Process Algebras. *Journal of Computer Security*, 3:5–33, 1994.

[23] Jun Pang. Analysis of a Security Protocol in μCRL. In *Proceedings of the 4th International Conference on Formal Engineering Methods (ICFEM): Formal Methods and Software Engineering*, pages 396–400. Springer, 2002.

[24] Thomas Y.C. Woo and Simon S. Lam. A Semantic Model for Authentication Protocols. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 178–194, 1993.

[25] Andrew D. Gordon and Alan Jeffrey. Authenticity by Typing for Security Protocols. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 145–159, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[26] Andrew D. Gordon and Alan Jeffrey. Typing Correspondence Assertions for Communication Protocols. *Theoretical Computer Science*, 300(1-3):379–409, 2003.

[27] Eduardo Bonelli, Adriana Compagnoni, and Elsa Gunter. Correspondence Assertions for Process Synchronization in Concurrent Communications. *Journal of Functional Programming*, 15(02):219–247, 2005.

[28] Annette M. Bleeker and Jan van Eijck. The Epistemics of Encryption. Technical report, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, 2000.

[29] Francien Dechesne and Yanjing Wang. Dynamic Epistemic Verification of Security Protocols: Framework and Case Study. In *Proceedings of the Workshop on Logic, Rationality and Interaction*, pages 129–144, 2007.

[30] Jan van Eijck and Simona Orzan. Epistemic Verification of Anonymity. *Electronic Notes in Theoretical Computer Science*, 168:159–174, 2007. Proceedings of the Second International Workshop on Views on Designing Complex Architectures (VODCA 2006).

[31] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2007.

[32] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge MA, August 1995.

Rasool Ramezanian was born in Mashhad in 1979. He got his BA and MS degrees in the Mathematics. In August 2008, he graduated from the PhD program of the Department of Mathematical Science, Sharif University of Technology, under supervision of Prof. M. Ardeshir. The title of his PhD thesis is "The Temporal Continuum and Dynamic Computation," which is categorized in constructive analysis. He currently works as an assistant professor in the same department.