

A new method of verification of security protocols

Andrew M. Mironov

Moscow State University
Faculty of Mechanics and Mathematics

amironov66@gmail.com

Abstract. In the paper we introduce a process model of security protocols, where processes are graphs with edges labelled by actions, and present a new method of specification and verification of security protocols based on this model.

Keywords: security protocols, process model, graph representation, verification

1 Introduction

1.1 Security protocols and their properties

A **security protocol (SP)** is a distributed algorithm that determines an order of message passing between several agents. Examples of such agents are computer systems, bank cards, people, etc. Messages transmitted by SPs can be encrypted. We assume that encryption transformations used in SPs are perfect, i.e. satisfy some axioms expressing, for example, an impossibility of extraction of open texts from ciphertexts without knowing the corresponding cryptographic keys.

In this paper we present a new model of SPs based on Milner's Calculus of Communicating Systems [1] and theory of processes with message passing [2]. This model is a graph analog of a Calculus of Cryptographic Protocols (**spi-calculus**, [3]). It can serve as a theoretical foundation for a new method (presented in the paper) of **verification** of SPs, where verification means a constructing of mathematical proofs that SPs meet the desired properties. Examples of such properties are **integrity** and **secrecy**. These properties are defined formally, as some conditions expressed in terms of an observational equivalence.

1.2 Verification of security protocols

There are examples of SPs ([4]–[8]) which were used in safety-critical systems, however it turned out that the SPs contain vulnerabilities of the following forms:

- agents involved in these SPs can receive distorted messages (or lose them) as a result of interception, deletion or distortion of transmitted messages by an adversary, that violates the integrity property,

- an adversary can find out a confidential information contained in intercepted messages as a result of erroneous or fraudulent actions of SP agents.

These examples justify that for SPs used in safety-critical systems it is not enough informal analysis of required properties, it is necessary

- to build a **mathematical model** of an analyzed SP,
- to describe security properties of the analyzed SP as mathematical objects (e.g. graphs, or logical formulas), called a **formal specification**, and
- to construct a mathematical proof that the analyzed SP meets (or does not meet) the formal specification, this proof is called a **formal verification**.

In the process model described in the paper SPs and their formal specifications are represented by processes with message passing. Many important properties of SPs (in particular, integrity and secrecy) can be expressed as observational equivalence of such processes.

One of the most significant advantages of the proposed process model of SPs is a low complexity of proofs of correctness of SPs. In particular, there is no need to build a set of all reachable states of analyzed SPs, if the set of all these states and transmitted messages is unbounded.

Among other models of SPs most popular are logical models ([9]–[13]). These models provide possibility to reduce the problem of verification of SPs to the problem of proofs of theorems that analyzed SPs meet their specifications. Algebraic and logical approaches to verification are considered also in [14]–[16].

2 Description of a process model of security protocols

In the process model described below SPs and formal specifications of their properties are represented by graphs, whose edges are labeled by **actions**. Actions are expressions consisting of terms and formulas.

2.1 Variables, constants, terms

We assume that there are given a set \mathcal{X} of **variables**, a subset $\mathcal{K} \subseteq \mathcal{X}$ of **keys**, and a set \mathcal{C} of **constants**. A set \mathcal{E} of **terms** is defined inductively:

- $\forall x \in \mathcal{X}, \forall c \in \mathcal{C}$ x and c are terms,
- for each list e_1, \dots, e_n of terms the record $e_1 \dots e_n$ is a term, (if the above list is empty, then the corresponding term is denoted by ε),
- $\forall k \in \mathcal{K}, \forall e \in \mathcal{E}$ the record $k(e)$ is a term (called an **encrypted message (EM)**, this term represents a result of an encryption of e on the key k).

Terms are designed for a representation of messages transmitted between participants of communications, a term of the form $e_1 \dots e_n$ represents a composite message consisting of messages corresponding to the components e_1, \dots, e_n . $\forall e \in \mathcal{E}$ the set of variables occurred in e is denoted by X_e . If terms e, e' have the form e_1, \dots, e_n and $e'_1, \dots, e'_{n'}$, respectively, then the record ee' denotes the term $e_1, \dots, e_n e'_1, \dots, e'_{n'}$, and $\forall e \in \mathcal{E}$ $\varepsilon e = e\varepsilon = e$.

2.2 Formulas

Elementary formulas (EFs) are records of the form $e = e'$ and $e \in E$ (where $e, e' \in \mathcal{E}$, and E is a subset of \mathcal{E}). A **formula** is a conjunction of EFs. The symbols \top and \perp denote true and false formulas respectively (for example, $\top = (c_1 = c_1)$, $\perp = (c_1 = c_2)$, where c_1 and c_2 are different constants). A set of formulas is denoted by \mathcal{B} . $\forall b \in \mathcal{B}$ X_b is a set of all variables occurring in b .

$\forall b_1, b_2 \in \mathcal{B}$ $b_1 \leq b_2$ means that b_2 is a logical consequence of b_1 (where the concept of a logical consequence is defined by a standard way).

If $b_1 \leq b_2$ and $b_2 \leq b_1$, then b_1 and b_2 are assumed to be equal.

$\forall k, k' \in \mathcal{K}$, $\forall e, e' \in \mathcal{E}$ the formulas $k(e) = k'(e')$ and $(k = k') \wedge (e = e')$ are assumed to be equal. The records $e_1 =_b e_2$ and $b \in_b E$ means that $b \leq (e_1 = e_2)$ and $b \leq (e \in E)$ respectively.

2.3 Closed sets of terms

Let $E \subseteq \mathcal{E}$ and $b \in \mathcal{B}$. The set E is said to be **b -closed** if

- $(\forall i = 1, \dots, n \ e_i \in E) \Leftrightarrow e_1 \dots e_n \in E$,
- $\forall k \in E \cap \mathcal{K} \ (e \in E \Leftrightarrow k(e) \in E)$,
- $\forall e, e' \in \mathcal{E} \ (e =_b e') \Rightarrow (e \in E \Leftrightarrow e' \in E)$.

Closed sets of terms are used for representation of sets of messages which can be known to an adversary. The above conditions correspond to operations which an adversary A can perform with his available messages:

- if A has e_1, \dots, e_n , then it can compose the message $e_1 \dots e_n$,
- if A has $e_1 \dots e_n$, then it may get its components e_1, \dots, e_n ,
- if A has k and e , where k is a key, then it can create a EM $k(e)$,
- if A has an EM $k(e)$ and a key k , then it can decrypt $k(e)$, i.e. get e .

Theorem 1. $\forall E \subseteq \mathcal{E}$, $\forall b \in \mathcal{B}$ there is a least (w.r.t. an inclusion of sets) b -closed set $E^b \subseteq \mathcal{E}$, such that $E \subseteq E^b$. ■

Let $D_1, D_2 \subseteq \mathcal{E}$, and $b_1, b_2 \in \mathcal{B}$. A binary relation $\mu \subseteq D_1^{b_1} \times D_2^{b_2}$ is said to be a **similarity** between (D_1, b_1) and (D_2, b_2) , if $\forall (e_1, e_2) \in \mu$

- $\forall e'_1, e'_2 \in \mathcal{E} \ (e'_1, e_2) \in \mu \Leftrightarrow (e_1 =_{b_1} e'_1), \ (e_1, e'_2) \in \mu \Leftrightarrow (e_2 =_{b_2} e'_2)$,
- the conditions $\exists e_i^1, \dots, e_i^n \in D_i^{b_i} : (e_i =_{b_i} e_i^1 \dots e_i^n) \ (i = 1, 2)$ are equivalent, and if these conditions hold, then $\forall i = 1, \dots, n \ (e_1^i, e_2^i) \in \mu$,
- the conditions $\exists k_i, e'_i \in D_i^{b_i} : (e_i =_{b_i} k_i(e'_i)) \ (i = 1, 2)$ are equivalent, and if these conditions hold, then $(k_1, k_2) \in \mu$ and $(e'_1, e'_2) \in \mu$.

A set of all similarities between (D_1, b_1) and (D_2, b_2) is denoted by the record $Sim((D_1, b_1), (D_2, b_2))$.

2.4 Actions

An **action** is a record of one of the three kinds: an input, an output, an internal action. Inputs and outputs are associated with an **execution**, defined below.

- An **input** is an action of the form $e?e'$, where $e, e' \in \mathcal{E}$. An execution of this action consists of a receiving a message through a channel named e , and writing components of this message to variables occurring in e' .
- An **output** is an action of the form $e!e'$, where $e, e' \in \mathcal{E}$. An execution of this action consists of a sending a message e' through a channel named e .
- An **internal action** is an action of the form b , where $b \in \mathcal{B}$.

The set of all actions is denoted by \mathcal{A} , $\forall a \in \mathcal{A}$ a set of variables occurred in a is denoted by X_a .

2.5 Processes with a message passing

Processes with a message passing are intended for description of SPs and formal specifications of their properties.

A **process with a message passing** (called below briefly as a **process**) is a tuple $P = (S, s^0, R, b^0, D^0, H^0)$, where

- S is a set of **states**, $s^0 \in S$ is an **initial state**,
- $R \subseteq S \times \mathcal{A} \times S$ is a set of **transitions**, each transition $(s, a, s') \in R$ is denoted by the record $s \xrightarrow{a} s'$,
- $b^0 \in \mathcal{B}$ is an **initial condition**,
- $D^0 \subseteq \mathcal{E}$ is a set of **disclosed terms**, values of these terms are known to both the process P and the environment at the initial moment, and
- $H^0 \subseteq \mathcal{X}$ is a set of **hidden variables**.

A set of all processes is denoted by \mathcal{P} , $\forall P \in \mathcal{P}$ the records $S_P, s_P^0, R_P, b_P^0, D_P^0, H_P^0$ denote the corresponding components of P . A set of variables occurring in P is denoted by X_P . A process P such that $R_P = \emptyset$ is denoted by $\mathbf{0}$.

A transition $s \xrightarrow{a} s'$ is said to be an **input**, an **output**, or an **internal transition**, if a is an input, an output, or an internal action, respectively.

A process P can be represented as a graph (denoted by the same symbol P): its nodes are states from S_P , and edges are corresponded to transitions from R_P : each transition $s \xrightarrow{a} s'$ corresponds to an edge from s_1 to s_2 labelled by a . We assume that for each process P under consideration the graph P is acyclic.

2.6 An execution of a process

An execution of a process $P \in \mathcal{P}$ can be informally understood as a walk on the graph P starting from s_P^0 , with an execution of actions that are labels of traversed edges. At each step $i \geq 0$ of this walk there are defined

- a state $s_i \in S_P$ of the process P at the moment i ,
- a condition $b_i \in \mathcal{B}$ on variables of P at the moment i , and

- a set $D_i \subseteq \mathcal{E}$ of **disclosed messages** at the moment i , i.e. messages known to both the process P and the environment at the moment i .

An **execution** of a process $P \in \mathcal{P}$ is a sequence of the form

$$(s_P^0, b_P^0, D_P^0) = (s_0, b_0, D_0) \xrightarrow{a_1} (s_1, b_1, D_1) \xrightarrow{a_2} \dots \xrightarrow{a_n} (s_n, b_n, D_n)$$

where $\forall i = 1, \dots, n$ $(s_{i-1} \xrightarrow{a_i} s_i) \in R_P$, $(b_i, D_i) = (b_{i-1}, D_{i-1})a_i$, and

$$\forall b \in \mathcal{B}, D \subseteq \mathcal{E}, a \in \mathcal{A} \quad (b, D)a = \begin{cases} (b, D \cup \{e\}), & \text{if } a = d?e \text{ or } d!e, \text{ where } d \in D^b, \\ (b \wedge a, D), & \text{if } a \in \mathcal{B}, \\ \text{undefined,} & \text{otherwise.} \end{cases}$$

We assume that a value of each variable $x \in H_P^0$ is unique and unknown to an environment of P at the initial moment of any execution of P .

A set of all executions of P can be represented by a labelled tree T_P , where

- a root t_P^0 of the tree T_P is labelled by the triple (s_P^0, b_P^0, D_P^0) , and
- if the set of edges of P outgoing from s_P^0 is $\{s_P^0 \xrightarrow{a_i} s_i \mid i = 1, \dots, m\}$, then for each $i \in \{1, \dots, m\}$, such that $\exists (b_i, D_i) = (b_P^0, D_P^0)a_i$,
 - T_P has an edge of the form $t_P^0 \xrightarrow{a_i} t_i$, and
 - a subtree growing from t_i is T_{P_i} , where $P_i = (S_P, s_i, R_P, b_i, D_i, H_P^0 \setminus D_i^{b_i})$.

The set of nodes of T_P is denoted by the same record T_P . For each node $t \in T_P$ the records s_t, b_t, D_t denote corresponding components of a label of t .

$\forall t, t' \in T_P$ the record $t \rightarrow t'$ means that either $t = t'$, or there is a path in T_P of the form $t = t_0 \xrightarrow{a_1} t_1 \xrightarrow{a_2} \dots \xrightarrow{a_m} t_m = t'$, where $a_1, \dots, a_m \in \mathcal{B}$.

2.7 Observational equivalence of processes

In this section we introduce a concept of observational equivalence of processes. This concept has the following sense: processes P_1 and P_2 are observationally equivalent iff for any external observer (which can observe a behavior of P_1 and P_2 by sending and receiving messages) these processes are indistinguishable.

An example of a pair of observationally equivalent processes is the pair

$$P_i = (\{s_i^0, s_i\}, s_i^0, \{s_i^0 \xrightarrow{c!k_i(e_i)} s_i\}, \top, \{c\}, \{k_i\}) \quad (i = 1, 2). \quad (1)$$

P_1 and P_2 send a unique message via channel c and then terminate. Any process observing an execution of P_1 and P_2 is unable to distinguish them.

Processes $P_1, P_2 \in \mathcal{P}$, are said to be **observationally equivalent** iff there is a binary relation $\mu \subseteq T_{P_1} \times T_{P_2}$ satisfying the following conditions:

1. $\forall (t_1, t_2) \in \mu \quad \exists \mu_{t_1, t_2} \in \text{Sim}((D_{t_1}, b_{t_1}), (D_{t_2}, b_{t_2}))$,
2. $(t_{P_1}^0, t_{P_2}^0) \in \mu, \forall (d_1, d_2) \in \mu_{t_{P_1}^0, t_{P_2}^0} \quad \exists d \in \mathcal{E} : d_i =_{b_{P_i}^0} d \quad (i = 1, 2)$,
3. $\forall (t_1, t_2) \in \mu$, for each edge $t_1 \xrightarrow{a_1} t'_1, \exists t'_2 \in T_{P_2} : (t'_1, t'_2) \in \mu, \mu_{t_1, t_2} \subseteq \mu_{t'_1, t'_2}$,
 - if $a_1 = d_1 \triangleright e_1$ ($\triangleright \in \{?, !\}$), then $\exists t, t' \in T_{P_2} : t_2 \rightarrow t, t' \rightarrow t'_2$, and $\exists d_2, e_2 : t \xrightarrow{d_2 \triangleright e_2} t', (d_1, d_2) \in \mu_{t'_1, t'_2}, (e_1, e_2) \in \mu_{t'_1, t'_2}$,
 - if $a_1 \in \mathcal{B}$, then $t_2 \rightarrow t'_2$,
4. a condition which is symmetric to condition 3: for each pair $(t_1, t_2) \in \mu$, and each edge $t_2 \xrightarrow{a_2} t'_2$ there is a node $t'_1 \in T_{P_1}$, such that $(t'_1, t'_2) \in \mu$, etc.

For example, processes P_i ($i = 1, 2$) from (1) are observationally equivalent, because in this case T_{P_i} has the form $(s_0^i, \top, \{c\}) \xrightarrow{c!k_i(e_i)} (s^i, \top, \{c, k_i(e_i)\})$, and the required μ is $\{(s_1^0, s_1), (s_2^0, s_2)\}$.

2.8 Operations on processes

In this section we define operations on processes which can be used for a construction of complex processes from simpler ones.

Prefix action $\forall a \in \mathcal{A}, \forall P \in \mathcal{P}$ $[a]P$ is a process defined as follows:

$$\begin{aligned} S_{[a]P} &\stackrel{\text{def}}{=} \{s\} \sqcup S_P, & s_{[a]P}^0 &\stackrel{\text{def}}{=} s, & R_{[a]P} &\stackrel{\text{def}}{=} \{s \xrightarrow{a} s_P^0\} \sqcup R_P, \\ b_{[a]P}^0 &\stackrel{\text{def}}{=} b_P^0, & D_{[a]P}^0 &\stackrel{\text{def}}{=} X_a \cup D_P^0, & H_{[a]P}^0 &\stackrel{\text{def}}{=} H_P^0. \end{aligned}$$

An execution of $[a]P$ can be informally understood as follows: at first the action a is executed, then $[a]P$ is executed just like P .

Choice $\forall P_1, P_2 \in \mathcal{P}$ $P_1 + P_2$ is a process defined as follows: all states of P_1 , that also belong to S_{P_2} , are replaced by fresh states, and

$$\begin{aligned} S_{P_1+P_2} &\stackrel{\text{def}}{=} \{s\} \sqcup S_{P_1} \sqcup S_{P_2}, & s_{P_1+P_2}^0 &\stackrel{\text{def}}{=} s, \\ R_{P_1+P_2} &\stackrel{\text{def}}{=} R_{P_1} \sqcup R_{P_2} \sqcup \{s \xrightarrow{a} s' \mid (s_{P_i}^0 \xrightarrow{a} s') \in R_{P_i}, i \in \{1, 2\}\}, \\ b_{P_1+P_2}^0 &\stackrel{\text{def}}{=} b_{P_1}^0 \wedge b_{P_2}^0, & D_{P_1+P_2}^0 &\stackrel{\text{def}}{=} D_{P_1}^0 \cup D_{P_2}^0, & H_{P_1+P_2}^0 &\stackrel{\text{def}}{=} H_{P_1}^0 \cup H_{P_2}^0. \end{aligned}$$

An execution of $P_1 + P_2$ can be understood as follows: at first it is selected (non-deterministically) a process $P_i \in \{P_1, P_2\}$ which can execute its first action, and then $P_1 + P_2$ is executed as the selected process.

Parallel composition $\forall P_1, P_2 \in \mathcal{P}$ (P_1, P_2) is a process defined as follows: all variables in $X_{P_1} \setminus D_{P_1}^0$, that also belong to $X_{P_2} \setminus D_{P_2}^0$, are replaced by fresh variables, and

- $S_{(P_1, P_2)} \stackrel{\text{def}}{=} S_{P_1} \times S_{P_2}$, $s_{(P_1, P_2)}^0 \stackrel{\text{def}}{=} (s_{P_1}^0, s_{P_2}^0)$,
- $R_{(P_1, P_2)}$ consists of the following transitions:
 - $(s_1, s_2) \xrightarrow{a} (s'_1, s_2)$, where $(s_1 \xrightarrow{a} s'_1) \in R_{P_1}$, $s_2 \in S_{P_2}$,
 - $(s_1, s_2) \xrightarrow{a} (s_1, s'_2)$, where $s_1 \in S_{P_1}$, $(s_2 \xrightarrow{a} s'_2) \in R_{P_2}$,
 - $(s_1, s_2) \xrightarrow{(d_1=d_2) \wedge (e_1=e_2)} (s'_1, s'_2)$, where $(s_i \xrightarrow{a_i} s'_i) \in R_{P_i}$ ($i = 1, 2$), $\{a_1, a_2\} = \{d_1!e_1, d_2?e_2\}$ (such transition is said to be **diagonal**),
- $b_{(P_1, P_2)}^0 \stackrel{\text{def}}{=} b_{P_1}^0 \wedge b_{P_2}^0$, $D_{(P_1, P_2)}^0 \stackrel{\text{def}}{=} D_{P_1}^0 \cup D_{P_2}^0$, $H_{(P_1, P_2)}^0 \stackrel{\text{def}}{=} H_{P_1}^0 \sqcup H_{P_2}^0$.

An execution of (P_1, P_2) can be understood as undeterministic interleaving of executions of P_1 and P_2 : at each moment of an execution of (P_1, P_2)

- either one of P_1, P_2 executes an action, and another is in waiting,
- or one of P_1, P_2 sends a message, and another receives this message.

A process $(\dots(P_1, P_2), \dots, P_n)$ is denoted by (P_1, \dots, P_n) .

Replication $\forall P \in \mathcal{P}$ a **replication** of P is a process P^\wedge that can be understood as infinite parallel composition (P, P, \dots) , and is defined as follows.

$\forall i \geq 1$ let P_i be a process which is obtained from P by renaming of variables: $\forall x \in X_P \setminus D_P^0$ each occurrence of x in P is replaced by the variable x_i , such that all the variables x_i are fresh. Components of P^\wedge have the following form:

- $S_{P^\wedge} \stackrel{\text{def}}{=} \{(s_1, s_2, \dots) \mid \forall i \geq 1 \ s_i \in S_P\}$, $s_{P^\wedge}^0 \stackrel{\text{def}}{=} (s_P^0, s_P^0, \dots)$,
- $\forall (s_1, \dots) \in S_{P^\wedge}, \forall i \geq 1, \forall (s_i \xrightarrow{a} s) \in R_{P_i}$ R_{P^\wedge} contains the transitions
 - $(s_1, \dots) \xrightarrow{a} (s_1, \dots, s_{i-1}, s, s_{i+1}, \dots)$, and
 - $(s_1, \dots) \xrightarrow{(d_1=d_2) \wedge (e_1=e_2)} (s_1, \dots, s_{i-1}, s, s_{i+1}, \dots, s_{j-1}, s', s_{j+1}, \dots)$, where $(s_j \xrightarrow{a'} s') \in R_{P_j}$ for some $j \neq i$, and $\{a, a'\} = \{d_1!e_1, d_2?e_2\}$,
- $b_{P^\wedge}^0 \stackrel{\text{def}}{=} b_P^0$, $D_{P^\wedge}^0 \stackrel{\text{def}}{=} D_P^0$, $H_{P^\wedge}^0 \stackrel{\text{def}}{=} \bigsqcup_{i \geq 1} H_{P_i}^0$.

Hiding $\forall P \in \mathcal{P}, \forall X \subseteq \mathcal{X}$ $P_X \stackrel{\text{def}}{=} (S_P, s_P^0, R_P, b_P^0, D_P^0 \setminus X, H_P^0 \cup X)$.

If $X = \{x_1, \dots, x_n\}$, then P_X is denoted by P_{x_1, \dots, x_n} .

Theorem 2. Observational congruence preserves operations of prefix action, parallel composition, replication and hiding. ■

2.9 A sufficient condition of an observational equivalence

Let $P \in \mathcal{P}$. A **labeling of states** of P is a set $\{(b_s, D_s) \mid s \in S\}$, such that

- $S \subseteq S_P, \forall s \in S \ b_s \in \mathcal{B}$ and $D_s \subseteq \mathcal{E}, s_P^0 \in S, b_{s_P^0} = b_P^0, D_{s_P^0} = D_P^0$,
- for each transition $(s \xrightarrow{a} s') \in R_P$, if $s' \in S$ then $s \in S$, and in this case
 - if $a = d \triangleright e$, where $\triangleright \in \{?, !\}$, then $d \in D_s^{b_s}, b_s \leq b_{s'}, D_s \cup \{e\} \subseteq D_{s'}^{b_{s'}}$,
 - if $a \in \mathcal{B}$, then $b_s \wedge a \leq b_{s'}$ and $D_s \subseteq D_{s'}$.

$\forall s, s' \in S_P$ the record $s \rightarrow s'$ means that either $s = s'$, or there is a set of thansitions of the form $s = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_m} s_m = s'$, where $a_1, \dots, a_m \in \mathcal{B}$.

Theorem 3 (a sufficient condition of an observational equivalence).

Let $P_1, P_2 \in \mathcal{P}$, where $S_{P_1} \cap S_{P_2} = \emptyset$. Then $P_1 \approx P_2$, if there are a binary relation $\mu \subseteq S_{P_1} \times S_{P_2}$ and labelings $\{(b_s, D_s) \mid s \in S_{P_1}\}, \{(b_s, D_s) \mid s \in S_{P_2}\}$ of states of P_1 and P_2 respectively, such that

1. each pair $(s_1, s_2) \in \mu$ is associated with $\mu_{s_1 s_2} \in \text{Sim}((D_{s_1}, b_{s_1}), (D_{s_2}, b_{s_2}))$,
2. $(s_{P_1}^0, s_{P_2}^0) \in \mu$, and each element of the set $\mu^0 \stackrel{\text{def}}{=} \mu_{s_{P_1}^0 s_{P_2}^0}$ has the form (x, x) , where $x \in D_{P_1}^0 \cap D_{P_2}^0$,
3. for each pair $(s_1, s_2) \in \mu$, and each transition $(s_1 \xrightarrow{a_1} s'_1) \in R_{P_1}$ there is a state $s'_2 \in S_{P_2}$, such that $(s'_1, s'_2) \in \mu, \mu_{s_1 s_2} \subseteq \mu_{s'_1 s'_2}$, and
 - if a_1 is input or output, then $a_1 = x \triangleright e_1$, where $\triangleright \in \{?, !\}, (x, x) \in \mu^0$, $\exists s, s' \in S_{P_2}: s_2 \rightarrow s, s' \rightarrow s'_2, \exists e_2: s \xrightarrow{x \triangleright e_2} s', (e_1, e_2) \in \mu_{s'_1 s'_2}$,
 - if $a_1 \in \mathcal{B}$, then $s_2 \rightarrow s'_2$,

4. a condition which is symmetric to condition 3: for each pair $(s_1, s_2) \in \mu$ and each transition $s_2 \xrightarrow{a_2} s'_2$, $\exists s'_1 \in S_{P_1} : (s'_1, s'_2) \in \mu$, etc. ■

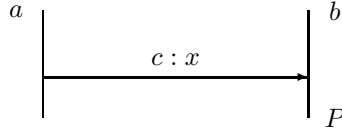
Theorem 4. Let P be a process, $\{(D_s, b_s) \mid s \in S\}$ be a labelling of P , and R_P has an edge $s \xrightarrow{a} s'$ such that $s, s' \in S$, and a has the form $d?k(e)$, where $D_s^{b_s}$ does not contain k and any term of the form $k(e')$. Then $P \approx P'$, where P' is obtained from P by removing the above edge and all unreachable (from s_P^0) states which appear after removing the edge. ■

3 Security protocols

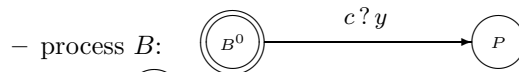
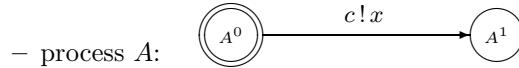
A **security protocol (SP)** is a process $P \in \mathcal{P}$ of the form $(P_1, \dots, P_n)_X$, where P_1, \dots, P_n are processes corresponding to **agents** involved in the SP, and $X \subseteq \mathcal{X}$ is a **shared secret** of the agents. In this section we present an application of the proposed approach to description, specification of properties and verification of several examples of SPs, all of them are analogs of examples from [3].

3.1 A message passing through a hidden channel

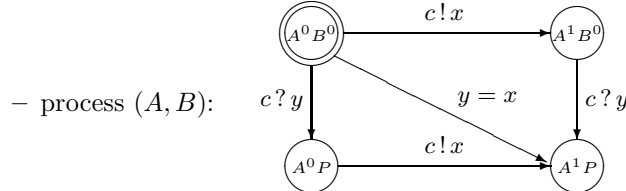
First example is a simplest SP for a message passing through a hidden channel. This SP consists of a sending of a message x from an agent a to an agent b through a channel named c (where only a and b know the name c of this channel), b receives the message and stores it in variable y , then b behaves like a process P . This SP is represented by the diagram



A behavior of a and b is represented by processes A and B respectively, $A \stackrel{\text{def}}{=} [c!x] \mathbf{0}$, $B \stackrel{\text{def}}{=} [c?y] P$ (where $c \notin P$). The SP is represented by the process $Sys \stackrel{\text{def}}{=} (A, B)_c$. Graph representations of processes in Sys is the following:



(where \textcircled{P} denotes a subgraph corresponded to the process P)



(where $\textcircled{A^0 P}$ and $\textcircled{A^1 P}$ denote subgraphs corresponded to copies of P (nodes

of these graphs are denoted by $A_i s$, where $i = 0, 1$, and $s \in S_P$), and the

arrow from $\bigcirc_{A^0 P}$ to $\bigcirc_{A^1 P}$ denotes a set of corresponding transitions from $A^0 s$ to $A^1 s$, where $s \in S_P$).

On the reason of theorem 4, the process $(A, B)_c$ is observationally equivalent to the process $\bigcirc_{A^0 B^0} \xrightarrow{y=x} \bigcirc_{A^1 P}$.

The process model allows formally describe and verify properties of integrity and secrecy of the above SP. These properties are as follows.

- An **integrity** of the SP is the following property: after a completion of the SP agent b receives the same message that has been sent by agent a .
- A **secrecy** of the SP is the following property:
 - for each pair x_1, x_2 of messages, which a can send b by this SP, and
 - for each two sessions of this SP, where the first session is a passing of x_1 , and the second one is a passing of x_2 ,
any external (i.e. different from a and b) agent, observing an execution of these sessions, is unable to extract from the observed information any knowledge about the messages x_1 and x_2 : whether the messages are the same or different (unless these knowledges are not disclosed by participants a, b). More accurately, the secrecy property can be described as follows: for any pair x_1, x_2 of messages, which a can send b by an execution of this SP
 - if for any external observer the process $[y = x_1] P$ (which describes a behavior of the agent b after receiving x_1) is indistinguishable from the process $[y = x_2] P$ (which describes a behavior of b after receiving x_2),
 - then for any sessions of an execution of this SP, where the first one is a passing of x_1 , and the second one is a passing of x_2 , any external agent, observing the execution of these sessions, can not determine, are identical or different messages transmitted in those sessions.

A formal description and verification of the properties of integrity and secrecy of this SP is as follows.

1. A property of **integrity** is described by the proposition

$$Sys \approx \tilde{Sys} \quad (2)$$

where \tilde{Sys} describes a SP which is defined like the original SP, but with the following modification of b : after a receiving a message and storing it in a fresh variable y' , a value of y is changed on a value that a sent really. A behavior of modified b is described by the process $\tilde{B} \stackrel{\text{def}}{=} [c ? y'] [y = x] P$, and the process \tilde{Sys} has the form $(A, \tilde{B})_c$.

Now we prove (2). The definition of operations on processes implies that

$$Sys \approx [y = x] P, \quad \tilde{Sys} \approx [y' = x] [y = x] P, \quad (3)$$

that implies (2), because $y' \notin [y = x] P$. ■

2. A property of **secrecy** of this SP is described by the implication

$$[y = x_1] P \approx [y = x_2] P \Rightarrow [x = x_1] Sys \approx [x = x_2] Sys \quad (4)$$

(where x_1, x_2 are fresh variables).

Now we prove (4). The the premise of implication (4) implies the statement

$$[y = x] [y = x_1] P \approx [y = x] [y = x_2] P,$$

which is equivalent to the statement

$$[x = x_1] [y = x] P \approx [x = x_2] [y = x] P. \quad (5)$$

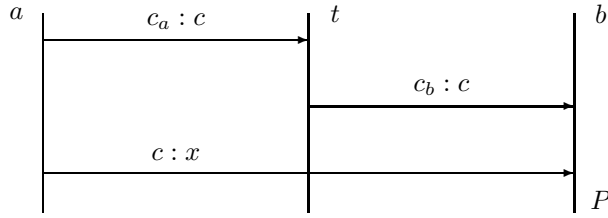
(5) and first proposition in (3) imply

$$[x = x_1] Sys \approx [x = x_1] [y = x] P \approx [x = x_2] [y = x] P \approx [x = x_2] Sys. \blacksquare$$

3.2 A SP with a creation of a new channel

Second SP consists of a message passing from a to b , with an assumption that a channel for this passing should be created during the execution of the SP. An auxiliary agent t is used in the SP (t is a trusted intermediary), and it is assumed that a name of a created channel must be known only to a , b , and t .

This SP is represented by the diagram



A behavior of agents a, t, b is represented by the processes A, T, B , where

$$A \stackrel{\text{def}}{=} [c_a ! c] [c ! x] \mathbf{0}, \quad T \stackrel{\text{def}}{=} [c_a ? c] [c_b ! c] \mathbf{0}, \quad B \stackrel{\text{def}}{=} [c_b ? c] [c ? y] P.$$

The SP is represented by the process $Sys \stackrel{\text{def}}{=} (A, T, B)_{c_a, c_b}$.

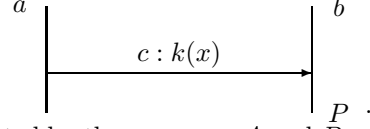
A formal description of integrity and secrecy of the SP is represented by propositions (2) and (4), where $\tilde{Sys} \stackrel{\text{def}}{=} (A, T, \tilde{B})_{c_a, c_b}$, $\tilde{B} \stackrel{\text{def}}{=} [c_b ? c] [c ? y'] [y = x] P$.

3.3 A passing of an encrypted message

Third example is a SP, which involves agents a and b having a common key k (only a and b know k), a and b can encrypt and decrypt messages by this key using a symmetric encryption system. The SP is as follows:

- a sends b a ciphertext $k(x)$ through an open channel c ,

- b receives the ciphertext, decrypts it, stores the extracted message x in the variable y , then behaves as a process P .



This SP is represented by the diagram

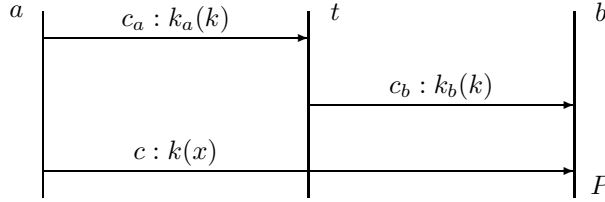
A behavior of agents a and b is represented by the processes A and B , where $A \stackrel{\text{def}}{=} [c!k(x)] \mathbf{0}$, $B \stackrel{\text{def}}{=} [c?k(y)] P$, and the SP is represented by $Sys \stackrel{\text{def}}{=} (A, B)_k$.

A formal description of the properties of integrity and secrecy of the SP is represented by (2) and (4), where $\tilde{Sys} \stackrel{\text{def}}{=} (A, \tilde{B})_k$, $\tilde{B} \stackrel{\text{def}}{=} [c?k(y')] [y = x] P$.

An integrity property of the SP is proposition (2), which in this case has the form $([c!k(x)] \mathbf{0}, [c?k(y)] P)_k \approx ([c!k(x)] \mathbf{0}, [c?k(y')] [y = x] P)_k$, and can be proven with use of theorem 3. To prove the secrecy property we prove implication (4). With use of theorem 3 it is not so difficult to prove that (3) and the premise of implication (4) imply $Sys \approx [y = x] P \approx [y = x'] P$, that proves (4).

3.4 Wide-Mouth Frog

A SP **Wide-Mouth Frog (WMF)** is intended for a passing of an encrypted message $k(x)$ from an agent a to an agent b with use of a trusted agent t , open channels c_a , c_b , c , and keys k_a , k_b , k , where k_a should be known only to a and t , k_b should be known only to b and t , and k should be known only to a , b and t . This SP is represented by the diagram



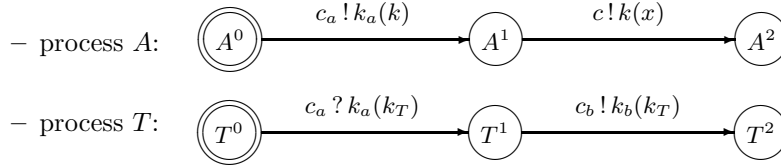
A behavior of agents a, t, b is represented by processes A, T, B , where $A \stackrel{\text{def}}{=} ([c_a!k_a(k)] [c!k(x)] \mathbf{0})_k$, $T \stackrel{\text{def}}{=} [c_a?k_a(k_T)] [c_b!k_b(k_T)] \mathbf{0}$,

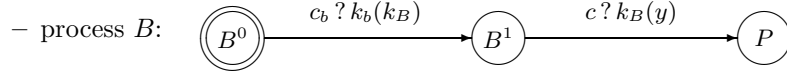
$B \stackrel{\text{def}}{=} [c_b?k_b(k_B)] [c?k_B(y)] P$. The SP is represented by $Sys \stackrel{\text{def}}{=} (A, T, B)_{k_a, k_b}$.

A formal description of the properties of integrity and secrecy of the SP is represented by the propositions (2) and (4), where

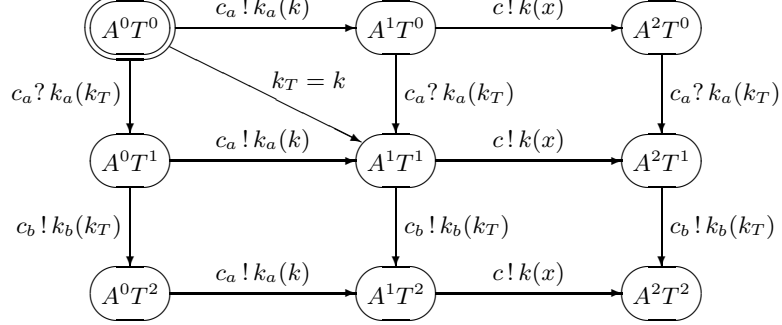
$$\tilde{Sys} \stackrel{\text{def}}{=} (A, T, \tilde{B})_{k_a, k_b}, \quad \tilde{B} \stackrel{\text{def}}{=} [c_b?k_b(k_B)] [c?k_B(y')] [y = x] P.$$

Graph representations of processes involved in Sys have the following form:



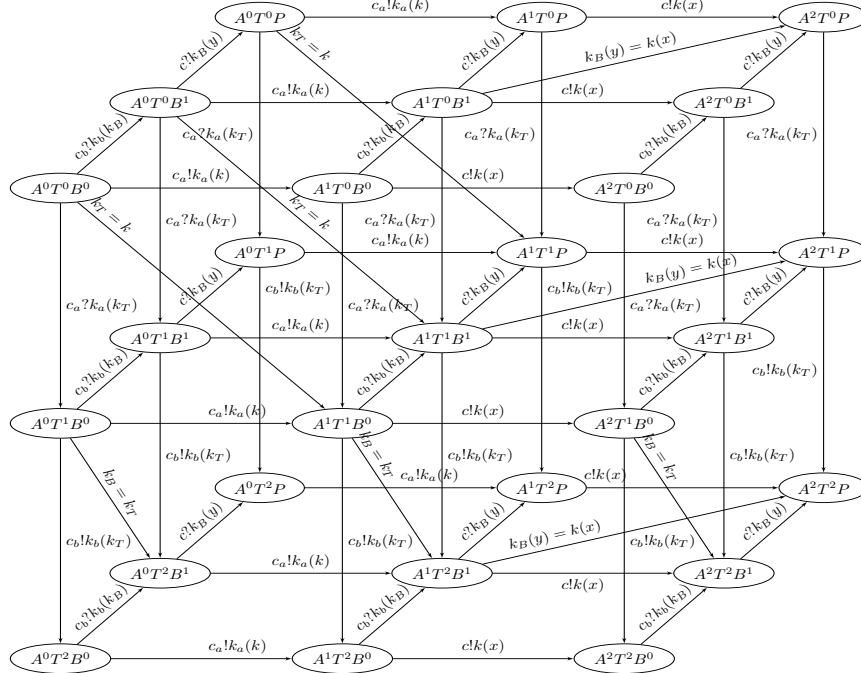


– process (A, T) :



(the diagonal transition in the diagram corresponds to a joint execution of the action $c_a ! k_a(k)$ of A , and the action $c_a ? k_a(k_T)$ of T),

– process (A, T, B) :



this diagram has diagonal transitions, related to a joint execution of

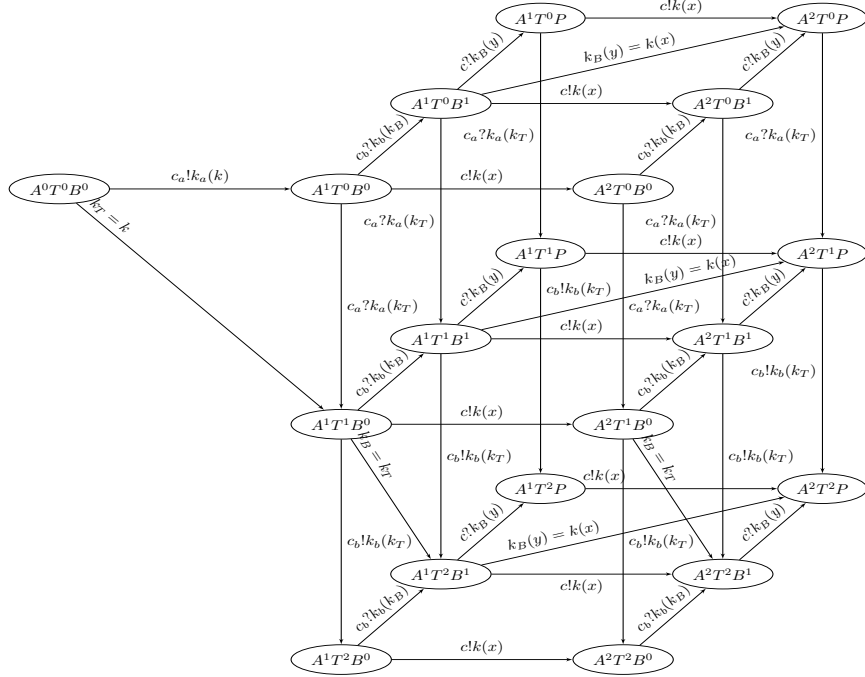
- action $c_b ! k_b(k_T)$ of (A, T) , and action $c_b ? k_b(k_B)$ of B (transitions of the form $A^0T^1B^0 \rightarrow A^0T^2B^1$, $A^1T^1B^0 \rightarrow A^1T^2B^1$, $A^2T^1B^0 \rightarrow A^2T^2B^1$, labelled by $k_B = k_T$), and
- action $c ! k(x)$ of (A, T) , and action $c ? k_B(y)$ of B (transitions of the form $A^1T^0B^1 \rightarrow A^2T^0P$, $A^1T^1B^1 \rightarrow A^2T^1P$, $A^1T^2B^1 \rightarrow A^2T^2P$, labelled by $k_B(y) = k(x)$),

Process $Sys \stackrel{\text{def}}{=} (A, T, B)_{k_a, k_b}$ has the same graph representation as the above process (A, T, B) . Its initial state is $A^0 T^0 B^0$.

First reduction of Sys is based on an applying of theorem 4 for the cases

- the edge is $A^0 T^0 B^0 \xrightarrow{c_a ? k_a(k_T)} A^0 T^1 B^0$, $D_{A^0 T^0 B^0} = \{c_a, c_b, c\}$, and
- the edge is $A^0 T^0 B^0 \xrightarrow{c_b ? k_b(k_B)} A^0 T^0 B^1$, $D_{A^0 T^0 B^0} = \{c_a, c_b, c\}$.

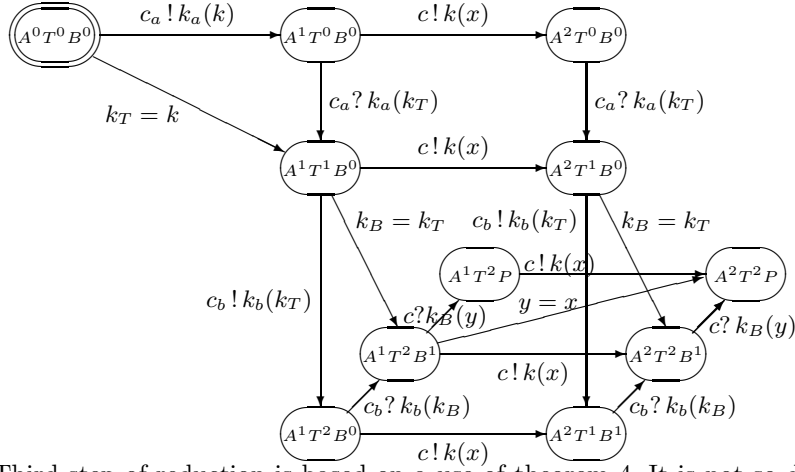
Removing the above edges and all nodes and edges which become unreachable from $A^0 T^0 B^0$ will result the graph



This graph also can be reduced with use of theorem 4 for the following cases:

- the edge is $A^1 T^0 B^0 \xrightarrow{c_b ? k_b(k_B)} A^1 T^0 B^1$, $D_{A^1 T^0 B^0} = \{c_a, c_b, c, k_a(k)\}$,
- the edge is $A^1 T^1 B^0 \xrightarrow{c_b ? k_b(k_B)} A^1 T^1 B^1$, $D_{A^1 T^1 B^0} = \{c_a, c_b, c, k_a(k)\}$,
- the edge is $A^2 T^0 B^0 \xrightarrow{c_b ? k_b(k_B)} A^2 T^0 B^1$, $D_{A^2 T^0 B^0} = \{c_a, c_b, c, k_a(k), k(x)\}$,
- the edge is $A^2 T^1 B^0 \xrightarrow{c_b ? k_b(k_B)} A^2 T^1 B^1$, $D_{A^2 T^1 B^0} = \{c_a, c_b, c, k_a(k), k(x)\}$.

Removing the above edges and all nodes and edges which become unreachable from $A^0 T^0 B^0$ will result to the graph

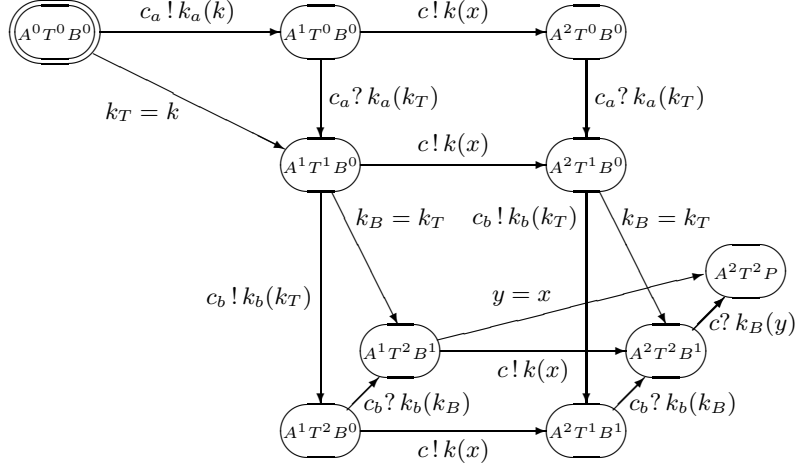


Third step of reduction is based on a use of theorem 4. It is not so difficult that there is a labelling for the process presented by the above graph:

$$\begin{cases} D_{A^0 T^0 B^0} = \{c_a, c_b, c\}, \\ D_{A^1 T^0 B^0} = D_{A^1 T^1 B^0} = \{c_a, c_b, c, k_a(k)\}, \\ D_{A^2 T^0 B^0} = D_{A^2 T^1 B^0} = \{c_a, c_b, c, k_a(k), k(x)\}, \\ D_{A^1 T^2 B^0} = D_{A^1 T^2 B^1} = D_{A^1 T^2 B^2} = \{c_a, c_b, c, k_a(k), k_b(k_T)\}, \\ D_{A^2 T^2 B^0} = D_{A^2 T^2 B^1} = D_{A^2 T^2 P} = \{c_a, c_b, c, k_a(k), k_b(k_T), k(x)\}, \\ b_{A^0 T^0 B^0} = b_{A^1 T^0 B^0} = b_{A^2 T^0 B^0} = \top, \\ b_{A^1 T^1 B^0} = b_{A^2 T^1 B^0} = b_{A^1 T^2 B^0} = b_{A^2 T^2 B^0} = (k = k_T), \\ b_{A^1 T^2 B^1} = b_{A^2 T^2 B^1} = (k = k_T) \wedge (k_T = k_B), \\ b_{A^2 T^2 P} = (k = k_T) \wedge (k_T = k_B) \wedge (x = y). \end{cases}$$

On the reason of theorem 4, the edge $A^1 T^2 B^1 \xrightarrow{c ? k_B(y)} A^1 T^2 P$ in the last diagram can be removed, because $b_{A^1 T^2 B^1} \leq (k_B = k)$, and $(k \in D_{A^1 T^2 B^1}) = \perp$.

The result of such removing is the process below:



It is not so difficult to prove that the property $b_{A^2 T^2 P} \leq (x = y)$ and the equivalence $[y = x] [y = x] P \approx [y = x] P$ imply $Sys \approx \tilde{Sys}$.

The secrecy property is a direct consequence of the integrity property. ■

References

1. Milner R. A Calculus of Communicating Systems // Lecture Notes in Computer Science, 1980. Vol. 92. 172 p.
2. Mironov A., A Method of a Proof of Observational Equivalence of Processes, Proceedings of the Fourth International Valentin Turchin Workshop on Meta-computation, Pereslavl-Zalessky, 2014, p. 194-222. See also <http://meta2014.pereslavl.ru/papers>, <https://arxiv.org/abs/1009.2259>
3. Abadi M., Gordon A., A Calculus for Cryptographic Protocols: The Spi Calculus, Proceedings of the Fourth ACM Conference on Computers and Communications Security, (1997) 36-47, ACM Press.
4. Denning D., Sacco G., Timestamps in Key Distribution Protocols, Communications of the ACM, Vol. 24, No. 8, (1981) 533-536.
5. Needham R., Schroeder M., Using Encryption for Authentication in large networks of computers, Communications of the ACM, 21(12), (1978) 993-999.
6. Needham R., Schroeder M., Authentication revisited, Operating Systems Review, Vol. 21, No. 1, (1987).
7. Cervesato I., Jaggar A.D., Scedrov A., Tsay J.-K., Walstad C., Breaking and fixing public-key Kerberos, Information and Computation Volume 206, Issues 2-4, (2008), Pages 402-424.
8. Lowe G., Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR, In Proceedings of TACAS, (1996) 147-166, Springer Verlag.
9. Burrows M., Abadi M., Needham R., A Logic of Authentication, ACM Transactions on Computer Systems, 8(1), (1990) 18-36.
10. Syverson P., van Oorschot P.C., On Unifying some Cryptographic Protocol Logics, Proceedings of the 1994 IEEE Computer Security Foundations Workshop VII, (1994) 14-29, IEEE Computer Society Press.
11. Syverson P., Meadows C., A Logical Language for Specifying Cryptographic Protocol Requirements, Proceedings of the 1993 IEEE Computer Security Symposium on Security and Privacy, (1993) 165-177, IEEE Computer Society Press.
12. Paulson L., Proving Properties of Security Protocols by Induction, Proceedings of the IEEE Computer Security Foundations Workshop X, (1997) 70-83, IEEE Computer Society Press.
13. Brackin S., A State-Based HOL Theory of Protocol Failure, (1997), ATR 98007, Arca Systems, Inc., <http://www.arca.com/paper.htm>.
14. Mark D. Ryan and Ben Smyth, Applied pi calculus, in: Formal Models and Techniques for Analyzing Security Protocols, Edited by Veronique Cortier, 2011 IOS Press, p. 112-142.
15. M. Abadi, B. Blanchet, C. Fournet. The Applied Pi Calculus: Mobile Values, New Names, and Secure Communication. [Research Report] ArXiv. 2016, pp.110. [hal-01423924](https://arxiv.org/abs/1609.03003), <https://arxiv.org/abs/1609.03003>
16. Ricardo Corin, Analysis Models for Security Protocols, Enschede, The Netherlands, 2006.