A Distributed Process Model of Cryptographic Protocols

Andrew M. Mironov

Moscow State University, Faculty of Mechanics and Mathematics,
Russia, 119991, Moscow, GSP-1, 1 Leninskiye Gory

Innopolis University, Leading Research Centre,
Russia, 420500, Innopolis, 1 Universitetskaya

amironov66@gmail.com
Abstract

Cryptographic Protocols (CP) are distributed algorithms intended for secure communication in an insecure environment. They are used, for example, in electronic payments, electronic voting procedures, systems of confidential data processing, etc. Errors in CPs can bring great financial and social damage, therefore, it is necessary to use mathematical methods to substantiate the correctness and safety of CPs. In this paper, a distributed process model of CPs is presented, which allows one to formally describe CPs and their properties. It is shown how to solve the problems of verification of CPs on the basis of this model.

Keywords: cryptographic protocols, sequential processes, distributed processes, verification

1 Introduction

1.1 A concept of a cryptographic protocol

A cryptographic protocol (CP) is a distributed algorithm that describes an order in which messages are exchanged between communicating agents. Examples of such agents are computer systems, bank cards, people, etc.

To ensure security properties of CPs (for example, integrity and confidentiality of transmitted data), cryptographic transformations are used in CPs (encryption and decryption, hash functions, etc.). We assume that the cryptographic transformations used in CPs are ideal, i.e. satisfy some axioms expressing, for example, the impossibility of extracting plain texts from ciphertexts without knowing the corresponding cryptographic keys.

1.2 Vulnerabilities in cryptographic protocols

Many vulnerabilities in CPs are induced not by insufficient qualities of cryptographic primitives used in CPs, but by logical errors in CPs. The most striking example of a vulnerability in a CP is a vulnerability in the Needham-Schroeder Public Key Authentication CP [87NS]. This CP was published in 1978, and a logical error in this CP was discovered [95L] more than 16 years after the start of the use of this CP. This error was related with the possibility of dishonest behavior of participants of this CP. The peculiarity of this error is that this CP is extremely simple, consisting of only three actions, and in a visual analysis of this CP, the absence of errors in this CP did not raise any doubts. The error was discovered with use of an automated verification tool. Another example (taken from [4CK]) is the following: in the CP, logging into the Google portal, which allows a user to identify himself only once, and then access various applications (such as Gmail or Google calendar), a logic error has been detected that allows a dishonest service provider to impersonate any of its users to another service provider. There are many other examples of CPs (see for example [81DS], [87NS], [95AN] [08CJSTW]), in which vulnerabilities of the following types were found:

- participants of these CPs can receive corrupted messages as a result of interception and distortion of transmitted messages by an adversary, which violates the integrity property of the transmitted messages,

- an adversary can find out the secret information contained in the intercepted messages, as a result of that the confidentiality property of the transmitted messages is violated.

There are also examples of vulnerabilities in CPs used for authentication to mobile phone providers, for withdrawing money from ATMs, for working with electronic passports, conducting electronic elections, etc.

All these examples justify that in safety-critical systems there is not enough an informal analysis of required security properties of CPs used in them, it is necessary to build mathematical models of analyzed CPs and prove that the models satisfy (or do not satisfy) their properties, the procedure for constructing such proofs is called a verification of CPs.

In this work, a distributed process model of CPs is constructed. In terms of this model it is possible to express properties of CPs such as integrity and confidentiality of transmitted messages, and authentication of participants of CPs. We present new verification methods of CPs based on this model.
1.3 An overview of methods of modeling and verification of cryptographic protocols

The books [11CK] and [12CM] contain overviews of the most widely used methods for modeling and verification of CPs. The main classes of models of CPs and approaches to verification of CPs are the following.

1. Logical models.
   This class of models was the very first approach to modeling and verification of CPs. On the base of this class of models, the problem of verification of CPs is reduced to the problem of constructing proofs of theorems (in some logical calculi) that CPs being analyzed have given properties. In the work [90BAN], the first mathematical model of CPs (called BAN logic) was presented. This model has limitations: it assumes that participants of analyzed CPs are honest, i.e. exactly fulfill their requirements. In addition, this model does not allow analyzing CPs with unlimited generation of sessions. The BAN logic approach was developed in works [90GNY], [91AT], [93vO], [93SM], [94KMM], [96SvO], [02SW], etc. An important class of logical calculi for CP modeling and verification is the Protocol Composition Logic, see [01DMP], [07DDMR], [08C], [11DMRS], etc.

One of the classes of CP logical models is associated with logical programming. In these models, CPs are represented in the form of term rewriting rules, Horn clauses and constraint systems. This approach is described in [01B], [05AB], [14CK], etc.

An important class of logical methods for CP modeling and verification is the Paulson’s inductive method: [97P], [98P], [99P], [00B], etc.

2. Models based on process algebra.
   The source of this class of models is Milner’s fundamental work [85H] and its continuation [89M]. In these works, a model of communicating processes is constructed, in which processes are represented by terms. An observational equivalence on these terms allows to express various properties of processes related to security (in particular, secrecy and anonymity). First work, which expounds a CP model based on Milner’s approach, is [99AC]. Among other works related to this direction, it should be noted [00RS], [01AF], [05KR], [07ABF], [11RS], [16ABP], [16B], [17CW], [21CDS].

3. Models based on CSP.
   CSP (Communicating Sequential Processes) is a mathematical approach developed by Hoare [85H] and intended for modeling and verification of distributed communicating processes. On the basis of this approach, a method for CP modeling and verification is built, which is most fully described in the book [00RSGLR]. Deductive CP verification method based on this approach uses the concept of a rank function. Among the works related to this direction, it should be noted [96SS], [96S], [97LR], [97DS], [98S], [21RCSSS].

4. Models based on strand spaces.
   Strand spaces allow to represent processes related to an execution of a CP in the form of graphical objects (called strands), which indicate dependencies between actions of participants of the CP. Among the works related to CP modeling and verification based on strand spaces approach, it should be noted [98THG1], [98THG2], [99THG1], [99THG2], [99THG2], [00GT2], [02GT], [05CDLMS], [07DGT1], [07DGT2], [07DGT3], [12G], [13LP], [16YEMM].

1.4 A comparison of the proposed model of cryptographic protocols with other models

A distributed process model of CPs presented in this paper inherits the most essential qualities of models from the above classes.

- In this model, CPs are represented in the form of distributed processes (DPs). Each DP is a set of communicating sequential processes (SPs), where SPs are models of participants of CPs communicating by asynchronous message passing.
- Typically, these SPs are sequences of actions that can be graphically represented as strands, and an execution of the entire CP can be represented as a strand space, where strands are
connected by edges representing sending and receiving of messages, see for example (68), (95).

• Properties of CPs can be represented by logical formulas, which can be substantiated using algorithms of logical inference. In section 3.1 we present an inference algorithm related to the introduced model of CPs.

• Some properties of CPs (for example, a property of anonymity in CPs of e-voting or e-commerce) can be expressed as an observational equivalence between the corresponding DPs (section 2.3.4), similarly to how it is done in CP models based on process algebra.

Main advantages of the proposed distributed process model are the following.

1. Proofs of CP correctness properties based on this model are much shorter and simpler than proofs of these properties based on other CP models. To substantiate this statement, we give examples of verification of two CPs: Yahalom [00RSLGR] and a CP of message transmission with unlimited number of participants taken from [99AG]. Verification of these CPs in the above sources takes several dozen pages, while verification of the Yahalom CP based on the proposed model (items 3.2.3, 3.2.4, 3.2.5) takes less than 4 pages, and verification of the second CP (item 3.3.2) - less than 3 pages. In addition, the analysis of the proofs of the correctness of the CPs shows that these proofs are produced according to a template method and can be generated automatically.

2. If a CP consists of a finite number of components without cycles, then a verification of such CP can be carried out completely automatically. The method of verification of such CPs is based on the concept of a transition graph (TG) introduced in section 2.7. An application of this method is shown on four examples of CPs (sections 3.1.4, 3.1.5, 3.1.7, 3.1.8) taken from [99AG]. A verification of the CPs in [99AG] is a non-trivial mathematical reasoning, while in the present work the CPs are verified by a standard calculating of formulas related to nodes of TGs.

3. The language for describing DPs introduced in this paper makes it possible to construct such models of CPs that are substantially similar to the original CPs. This is essential when a flaw in a model of an analyzed CP is found, and it is necessary to correct the CP in such a way as to eliminate this flaw: for an elimination of the detected flaw

- the model of this CP can first be corrected,
- which is then easily converted to a correction of the original CP.

The language for describing DPs introduced in this paper can be considered as a new language for describing distributed algorithms.

2 Sequential and distributed processes

In this section, we introduce concepts of sequential and distributed processes. A sequential process is a model of a CP participant, and a distributed process is a model of the entire CP. The proposed model is a theoretical basis for CP verification methods described in section 3.

2.1 Auxiliary concepts

2.1.1 Types, constants, variables, function symbols

We assume that there are given sets Types, Con, Var and Fun, elements of which are called types, constants, variables, and function symbols (FSs), respectively. Each element $x$ of Con, Var and Fun is associated with a type $\tau(x) \in$ Types, and if $x \in$ Fun, then $\tau(x)$ has the form

$$\tau_1, \ldots, \tau_n \rightarrow \tau,$$

where $\tau_1, \ldots, \tau_n, \tau \in$ Types.

2.1.2 Terms

In this section we define a set of $Tm$ of terms, which are intended to describe messages sent during the execution of a CP. The set $Tm$ is defined inductively. Each term $e$ is associated with a type $\tau(e) \in$ Types. The definition of a term is as follows:

- $\forall x \in$ Con $\cup$ Var $\ x$ is a term of the type $\tau(x)$,
We assume that the term $e, e'$ of the type $\tau$ contains the following types:

- $\forall$, terms of this type are called **nonces**, they denote variables with unique values (\texttt{nonce = "number only used once"}),
- $\mathbf{P}$, terms of this type are called **processes**.

We use the following notations and conventions:

- $\mathbf{M}$ includes all other types from $\text{Types}$, i.e. a term of any type is also a term of type $\mathbf{M}$,
- $\text{Var}_{\mathbf{C}}$ contains a variable denoted by $\circ$ and called an **open channel**,
- $\forall n \geq 1, \forall \tau \in \text{Types}$ the set $\text{Types}$ contains type $\tau_n$, whose values are $n$-tuples consisting of terms of the type $\tau$.

### 2.1.4 Examples of function symbols

We assume that $\text{Fun}$ contains the following FSs.

- FSs $\text{encrypt}$ and $\text{decrypt}$ of the type

\[ (K, \mathbf{M}) \rightarrow \mathbf{M}. \]

Terms $\text{encrypt}(k, e)$ and $\text{decrypt}(k, e)$ denote messages obtained by encrypting (and decrypting, respectively) the message $e$ on the key $k$. Term $\text{encrypt}(k, e)$ is denoted by $k(e)$, and is called an **encrypted message** (EM).

- FSs $\text{shared}_k$ of the type

\[ A_n \rightarrow K, \text{ where } n \geq 2. \]

Term $\text{shared}_k(A_1, \ldots, A_n)$ is called a **shared key** of agents $A_1, \ldots, A_n$ and is denoted by $k_{A_1, \ldots, A_n}$.

- FSs $\text{shared}_c$ of the type

\[ A_n \rightarrow \mathbf{C}, \text{ where } n \geq 2. \]

Term $\text{shared}_c(A_1, \ldots, A_n)$ is called a **shared channel** of agents $A_1, \ldots, A_n$ and is denoted by $c_{A_1, \ldots, A_n}$.

We will use the following notation: $\forall e \in Tm$

\[ \text{Keys}(e) = \{ k \in \text{Var}_K \mid \exists e' \in Tm : k(e') \subseteq e \}. \]
2.1.5 Expressions
In this section we define a set $\text{Expr}$ of expressions, where each expression describes a set of terms. Such a set can be a set of terms that are currently available to some process, or a set of messages that are currently in a channel.

An expression is a notation of one of the following forms:

- $E$, where $E \subseteq Tm$,
- $[P]$ and $|c|$, where $P \in \text{Var}_P$, $c \in Tm_C$,
- $k^{-1}(E)$, where $k \in Tm_K$, and $E \in \text{Expr}$,
- $E \cap E'$, $E \cup E'$, where $E, E' \in \text{Expr}$.

$\forall E \in \text{Expr}$ $\text{Var}(E)$ is a set of all variables occurred in $E$.

Expressions $k^{-1}([P])$ and $k^{-1}(|c|)$ are denoted by $k^{-1}[P]$ and $k^{-1}[c]$ respectively. Expressions $\{e\}$, where $e \in Tm$, are denoted without braces.

2.1.6 Formulas
In this section we define a set $Fm$ of formulas, which are intended to describe properties of sets of terms. In this definition, a concept of elementary formula (EF) is used, which is a notation of one of the following forms:

1. $e \in E$, $E = E'$, $E \subseteq E'$, $E \supseteq E'$, where $e \in Tm$, $E, E' \in \text{Expr}$,
2. $E \perp c, P \perp K, P$, where $E \subseteq Tm, P \in \text{Var}_P$,
3. $at_P = i$, where $P \in \text{Var}_P, i \in \{0, 1, . . . \}$.

EFs express properties of values of expressions occurred in them. An example of a EF is:

\[
decrypt(k, k(e)) = e, \text{ where } k \in Tm_K, e \in Tm.
\]

A formula is a set of EFs. Each formula $\varphi = \{\varphi_i \mid i \in I\}$ expresses a statement that is a conjunction of statements expressed by EFs $\varphi_i \ (i \in I)$.

$\forall \varphi \in Fm$ the set of all variables occurred in $\varphi$ is denoted by $\text{Var}(\varphi)$.

$\forall \varphi_1, . . . , \varphi_n \in Fm$, the formula $\varphi_1 \cup \ldots \cup \varphi_n$ is denoted by $\{\varphi_1, \ldots, \varphi_n\}$.

A formula $\{E_0|p_1E_1, E_1p_2E_2, \ldots, E_{n-1}p_nE_n\}$, where $p_1, \ldots, p_n$ are symbols from $\{=, \subseteq\}$, is denoted by $E_0|p_1E_1p_2 \ldots p_nE_n$.

2.1.7 Bindings
A binding is a function $\theta : \text{Var} \rightarrow Tm$. We say that $\theta$ binds a variable $x \in \text{Var}$ with term $\theta(x)$.

We use the following notation:

- the set of all bindings is denoted by $\Theta$,
- $id$ denotes identical binding:

\[
\forall x \in \text{Var} \ id(x) = x,
\]

- $\forall X \subseteq \text{Var} \ \Theta(X) = \{\theta \in \Theta \mid \forall x \in \text{Var} \setminus X \ \theta(x) = x\}$,
- a binding $\theta \in \Theta$ can be denoted by

\[
x \mapsto \theta(x) \text{ or } (\theta(x_1)/x_1, \ldots, \theta(x_n)/x_n),
\]

second notation in (2) is used when $\theta \in \Theta(\{x_1, \ldots, x_n\})$.

- $\forall \theta \in \Theta, \forall e \in Tm \ e^\theta$ denotes a term obtained from $e$ by replacing $\forall x \in \text{Var}(e)$ each occurrence of $x$ in $e$ by the term $\theta(x)$, the term $e$ is called a template of $e^\theta$ with respect to $\theta$,

- $\forall \theta, \theta' \in \Theta$ the binding $x \mapsto (x^\theta)^{\theta'}$ is denoted by $\theta \theta'$.

If $X \subseteq X' \subseteq \text{Var}, \theta \in \Theta(X), \theta' \in \Theta(X')$, and $\forall x \in X \ \theta(x) = \theta'(x)$, then $\theta'$ is said to be an extension of $\theta$.

2.2 Sequential processes
2.2.1 Actions
Actions are notations of the following forms:

- $cle, c\!\!e, e := e'$, where $c \in Tm_C, e, e' \in Tm$,

which is called a sending $e$ to $c$, a receiving $e$ from $c$, and an assignment, respectively.

The set of all actions is denoted by $\text{Act}$.

$\forall \alpha \in \text{Act}$ the set of all variables occurred in $\alpha$ is denoted by $\text{Var}(\alpha)$.

If $\theta \in \Theta, \alpha \in \text{Act}$, then $\alpha^\theta$ denotes the action $\alpha^\theta|e^\theta, e^\theta|e^\theta$ and $e^\theta := (e')^\theta$, if $\alpha = cle, c\!\!e$ and $e := e'$, respectively.

Actions can be written in round parentheses:

$(cle), (c\!\!e), (e = e')$. 

2.2.2 A concept of a sequential process

A sequential process (SP) is a 4-tuple $(P, A, X, \bar{X})$, whose components have the following meanings:

- $P$ is a graph with a selected node (called an initial node, and denoted by $Init(P)$), each edge of which is labeled by some action,
- $A$ is an agent associated with this SP,
- $X \subseteq Var$ is a set of initialized variables,
- $\bar{X} \subseteq X$ is a set hidden variables, they denote secret keys, hidden channels, or nonces, these variables are initialized with unique values.

A SP is a description of a behavior of a system, a work of which consists of sending or receiving messages, and an initialization of uninitialized variables.

For each SP $(P, A, X, \bar{X})$

- this SP can be denoted by the same notation $P$ as its graph, the set of nodes of the graph $P$ is also is denoted by $P$,
- $Agent(P), X(P), \bar{X}(P)$ denote corresponding components of $P$,
- $Var(P)$ is a set of all variables occurred in $P$,
- $\bar{X}(P) = X(P) \setminus \bar{X}(P)$,
- $\bar{X}(P) = Var(P) \setminus X(P)$.

Each SP is associated with a variable of the type $P$, called a name of this SP. We will denote names of SPs by the same notations as the SPs.

If $P$ has no edges and $X(P) = \emptyset$, then $P$ is denoted by $0$.

Actions of the form $\circ e$ and $\circ ? e$ will be abbreviated as $\circ e$ and $\circ ? e$ respectively.

2.2.3 A state of a sequential process

A state of a SP $P$ is a 5-tuple

$$s = (at, \alpha, [P], \theta, [e] \mid e \in Tm_C),$$

where

- $at \in P$ is a node of the graph $P$ in $s$,
- $\alpha \in \{init\} \cup \text{Act}$ is an action before transition to $s$,
- $[P] \subseteq Var$ is a set of initialized variables in $s$,
- $\theta \in \Theta([P])$ is a binding in $s$, and
- $\forall c \in Tm_C \ [e] \subseteq Tm$ is a content of the channel $c$ in $s$.

The components of $s$ are denoted by $at_s, \alpha_s, [P]_s, \theta_s, [e]_s$ respectively.

The set $Tm([P]_s)$ is denoted by $(P)_s$.

A state of SP $P$ is said to be initial, and is denoted by $0_P$, if it has the form

$$(Init(P), init, X(P), id, \{\emptyset| e \in Tm_C\}).$$

2.2.4 Values of expressions and formulas in states

Let $P$ be a SP, $s$ be a state of $P$, $E \in Expr$, and $\varphi \in Fm$.

The notation $E^s$ denotes a set of terms, called a value of $E$ in $s$, and defined as follows:

- $\forall e \subseteq Tm \ E^s = \{e^\theta \mid e \in E\}$,
- $\forall e \in Tm$ the set $\{e\}^s$, and the only element of this set, is denoted by $e^s$,
- $[P]^s = ([P]_s)^s$, $(P)^s = ((P)_s)^s$, $[e] = [e]^s$, where $P \in Var_P, e \in Tm_C$,
- $k^{-1}(E)^s = \{e \in Tm \mid \exists e' \in E : k^s(e) \subseteq e'\}$,
- $(E \cap E')^s = E^s \cap (E')^s$, $(E \cup E')^s = E^s \cup (E')^s$.

The notation $s \models \varphi$ denotes the statement $\varphi$ holds at $s$.

This statement is true if $Var(\varphi)_P \subseteq \{P\}$, and one of the following conditions holds:

- $\varphi$ has the form $(e \in E), (E = E'), (E \subseteq E')$, or $(E \supseteq E')$,

where $e \in Tm, E, E' \in Expr$, and

$$e^s \in E^s, E^s = (E')^s, E^s \subseteq (E')^s, E^s \supseteq (E')^s,$$

respectively.
\( \varphi = (E \perp_c P) \), \( \forall e \in E^s \) \( \text{Agent}(P) \not\in e \), and
\[
\begin{align*}
\forall x \in E^s_X, \forall y \in [P]_s, \ x \not\in y^s \\
\forall x \in E^s_{\overline{X}}, \forall e \in Tm_C \\
\text{if } \exists e \in [c]_s : x \in e, \text{ then } e \in E^s
\end{align*}
\]
\[3\] can be interpreted as the statement: each variable from \( E^s_X \)
- is not occurred in terms available to \( P \) in the state \( s \), and
- is occurred in terms from the content of only those channels that are not available for \( P \).

\( \varphi = (E \perp_K P) \), \( \forall e \in E^s \) \( \text{Agent}(P) \not\in e \), and
\[
\begin{align*}
\forall x \in E^s_K, \forall y \in [P]_s, \ x \perp_{K,E} y^s \\
\forall x \in E^s_{\overline{K}}, \forall e \in Tm_C, \forall e \in [c]_s \\
x \perp_{K,E} e
\end{align*}
\]
where \( x \perp_{K,E} e \) means that
\( k(...) \subseteq e \), where \( k \in E^s_K \)
\[4\] can be interpreted as the statement: variables from \( E^s_K \) are occurred
- in terms available to \( P \) in the state \( s \), and
- in terms from a content of any channel, in a “secure” form, i.e. are occurred in sub-
terms of the form \( k(...) \), where \( k \in E^s_K \),

- \( \varphi = (at_p = i) \), and \( at_s = i \)
- \( \varphi = \{ \varphi_i \mid i \in I \} \) and \( \forall i \in I \ s \models \varphi_i \).

2.2.5 An execution of a sequential process

An execution of a SP \( P \) is a walk in the graph \( P \), starting from \( Init(P) \), with an execution of actions that are labels of passed edges. Each step of the execution is associated with a state \( s \) of \( P \), called a current state of \( P \) at this step (a current state at first step is \( 0_P \)). If a step of the execution is not final, then the current state \( s \) is replaced by the state \( s' \), which will be a current state at the next step of the execution, for this

1. either an edge \( e \) outgoing from \( at_s \) is selected, whose label \( a \) has the following properties:

- if \( a^\rho e \) contains a subterm \( shared_key(...) \) or \( shared_channel(...) \), then \( \text{Agent}(P) \) occurs in this subterm,
- one of the following conditions holds:
  - (a) \( \alpha = cle, \ c, e \in (P)_s \)
  - (b) \( \alpha = e^c, \ c \in (P)_s, \ Keys(e^c) \subseteq [P]_s, \exists \theta \in \Theta(Var(c) \setminus [P]_s) : \ (e^\theta)^c \in [c]^s \)
  - (c) \( \alpha = \ (e := e'), \ e' \in (P)_s, \ Keys(e') \subseteq [P]_s, \exists \theta \in \Theta(Var(c) \setminus [P]_s) : e^\theta = e' \)

and \( s' \) is defined as follows: \( at_{s'} \) is an end of the edge \( e \), \( \alpha_{s'} = \alpha \), and

- if (a) in \([5]\) holds, then
  \( [P]_{s'} = [P]_s, \theta_{s'} = \theta_s, \)
  \( [c]^s_{s'} = [c]^s_s \cup \{ e' \}, \)
  \( \forall e' \in Tm_C \setminus \{ e' \} \ [c']_{s'} = [c]_{s}, \)

- if (b) or (c) in \( [6] \) holds, then
  \( [P]_{s'} = [P]_s \cup Var(c), \theta_{s'} = \theta_s, \)
  \( \forall e' \in Tm_C \ [c']_{s'} = [c]_{s}, \)

(we say that each \( x \in Var(c) \setminus [P]_s \) is initialized with the value \( x^\theta_{s'} \) in the transition from \( s \) to \( s' \)).

2. or all components of \( s' \), with the exception of the last component, are equal to the corresponding components of \( s \), and \( \forall e \in Tm_C \) the set \( [c]_{s'} \) either is equal to \( [c]_s \), or is obtained by adding a term to \( [c]_s \) as a result of an execution of a step by another SP.

If first (second) of the above situations takes place, then we say that \( s' \) is obtained by an active (passive, respectively) transition from \( s \), and denote this by \( s \rightarrow s' \) (\( s \Rightarrow s' \), respectively).

Variables in \( Var(P) \) have the following features:
\( \forall x \in Var(P) \)

- if \( x \in X(P) \) (or \( x \in \overline{X}(P) \)), then \( x \) is initialized (or not initialized, respectively) at the initial moment of each execution of \( P \),
- if \( x \in \overline{X}(P) \), then \( x \) is initialized by a unique value at the initial moment of each execution of \( P \).
Conditions (a), (b) and (c) in (6) have the following meaning:

(a) is related to sending a message $c!e$:
- a name $e^a$ of a channel to which the message is sent is available to $P$ in the state $s$,
- the sent message $e^a$ is a term whose components are available to $P$ in the state $s$,
- each EM in the received message, which * must be decrypted during the receiving this message, and
  * is not encrypted on a shared key, has the form $k(...)$, where $k$ must be available to $P$ in the state $s$, this property is expressed in second line of (6)(b),
- term $e$ is a template of some term from $[e]$, with respect to some extension of $\theta$, this property is expressed in last line of (6)(b),

(b) is related to receiving a message $c?e$:
- a name $e^a$ of a channel from which the message is received, is available to $P$ in the state $s$,
- each EM in the received message, which * must be decrypted during the receiving this message, and
- term $e$ is a template of some term from $[e]$, with respect to some extension of $\theta$, this property is expressed in second line of (6)(b),

(c) is related to an assignment $e := e'$:
- each component of $(e')^a$ must be available to $P$ in the state $s$,
- a meaning of property in second line of (6)(c) coincides with a meaning of corresponding properties in (6)(b): each EM in $(e')^a$, which must be decrypted during the assignment, has the form $k(...)$ or $Agent(P)(...)$, and
  * either $k$ is a shared key,
  * or $k \in Var_P$, and $k$ is available to $P$ in the state $s$,
- $e$ is a template of $e'$ with respect to some $\theta \in \Theta(Var(e) \setminus [P]_s)$.

2.2.6 An adversary

An adversary is a SP $P_1$ with the properties:
- the graph $P_1$ consists of a single node,
- $\forall \tau \in Types$ the sets $\bar{X}(P_1)$, and $\bar{X}(P_1)$, are countable,
- $\forall \alpha \in Act \ P_1$ has an edge labelled by $\alpha$.

We assume that $P_1$ is the only SP whose graph has cycles.

2.2.7 Renamings

A renaming is an injective map $\eta : X \to X'$, where $X, X' \subseteq Var$.

For each renaming $\eta : X \to X'$, each $e \in Tm$ and each SP $P$, notations $e^\eta$ and $P^\eta$ denote a term or a SP respectively, obtained from $e$ or $P$ respectively by replacing $\forall x \in X$ each occurrence of $x$ with $\eta(x)$.

If a renaming $\eta$ has the form

\[
\eta : \bar{X}(P) \cup \bar{X}(P) \to Var \setminus \bar{X}(P),
\]

then SPs $P$ and $P^\eta$ are considered as equal.

2.3 Distributed processes

In this section we introduce a concept of a distributed process (DP). DPs are models of CPs. All CPs considered in the paper are represented as DPs.

2.3.1 A concept of a distributed process

A distributed process (DP) is a family of SPs: $\mathcal{P} = \{P_i \mid i \in I\}$. Each DP is associated with a variable of the type $P$, called a name of this DP.

A DP is a model of a distributed algorithm, components of which communicate with each other by message passing through channels.

Let $\mathcal{P}$ be a DP. We will use the following notations and assumptions:

- $Var(\mathcal{P}) = \bigcup_{P \in \mathcal{P}} Var(P)$, the sets $X(\mathcal{P})$, $\bar{X}(\mathcal{P})$, $\bar{X}(\mathcal{P})$ are defined similarly,
- we will assume that

  \[
  \text{components of the family}
  \{X(P) \cup \bar{X}(P) \mid P \in \mathcal{P}\}
  \text{are disjoint sets}
  \text{and do not intersect with } \bar{X}(\mathcal{P})
  \]

  (if this is not the case, then replace each $P \in \mathcal{P}$ by an equal SP in the sense described at the end of 2.2.7 so that (7) will be satisfied),
A DP $\mathcal{P} = \{P_i \mid i \in I\}$ can be denoted by
- $\{P_1, \ldots, P_n\}$, if $I = \{1, \ldots, n\}$ (in the case $n = 1$ the brackets can be omitted, i.e. the DP $\{P_1\}$ is denoted by $P_1$), and
- $P^*$, if $I = \{1, 2, \ldots\}$, and all SPs in $\mathcal{P}$ are equal to $P$,
- the notation $\mathcal{P}_I$ denotes the DP $\{\mathcal{P}, P_1\}$,
- if $\{P_i \mid i \in I\}$ is a family of DPs, and for each $i \in I$ $\mathcal{P}_i = \{P_{i'} \mid i' \in I_i\}$, where sets of indices $I_i$ ($i \in I$) are disjoint (if this is not the case, then we replace them with the corresponding disjoint copies), then the notation $\{\mathcal{P}_i \mid i \in I\}$ denotes also the DP $\{P_{i'} \mid i' \in \bigsqcup_{i \in I} I_i\}$.

We will use the following convention:
- if, in some reasoning related to a DP of the form $P^*$, some SP is the first of the considered SPs from $P^*$, then this SP and all its variables are denoted by the same notations as in $P$,
- if, in addition to this SP, another SP from $P^*$ is considered, then it is denoted by $\hat{P}$, and in the notation of those of its variables that correspond to variables from $\hat{X}(P) \cup \hat{X}(P)$ are used backstrokes, etc.

### 2.3.2 A concept of a state of a distributed process

A state of a DP $\mathcal{P}$ is a family $s = \{s_P \mid P \in \mathcal{P}\}$ of states of SPs occurred in $\mathcal{P}$ such that $\forall c \in Tm_C$ all components of the family $\{[c]_{s_P} \mid P \in \mathcal{P}\}$ are the same (we will denote them by $[c]_s$).

Let $s = \{s_P \mid P \in \mathcal{P}\}$ be a state of a DP $\mathcal{P}$. Then
- $s$ is said to be an initial state of $\mathcal{P}$, and is denoted by $0\mathcal{P}$ (or more briefly by 0, if the DP $\mathcal{P}$ is clear from the context), if $\forall P \in \mathcal{P}$ $s_P = 0_P$,
- $at_s = \{at_{s_P} \mid P \in \mathcal{P}\}$, $[P]_s = \bigcup_{P \in \mathcal{P}} [P]_s$, $(\mathcal{P})_s = Tm([\mathcal{P}]_s)$,
- $\theta_s$ denotes a binding from $\Theta([\mathcal{P}]_s)$ such that
  \[
  \forall P \in \mathcal{P}, \forall x \in [P]_s \theta_{s_P}(x) = \theta_s(x)
  \]
  (an existence of such a binding follows from assumption (1))

### 2.3.3 An execution of a distributed process

An execution of a DP $\mathcal{P}$ is a non-deterministic alternation of executions of SPs occurred in $\mathcal{P}$. At each step of the execution only one SP from $\mathcal{P}$ executes an active transition, and other SPs from $\mathcal{P}$ execute passive transitions.

An execution of a DP $\mathcal{P}$ can be understood as a generation of a sequence of states of $\mathcal{P}$ (starting from $0\mathcal{P}$), in which each pair $(s, s')$ of adjacent states belongs to a transition relation, which means the following: $\exists P \in \mathcal{P}$:

\[
s_P \xrightarrow{\alpha_P} s'_P, \forall P' \in \mathcal{P} \setminus \{P\} s_{P'} \xrightarrow{\alpha_{P'}} s'_{P'},
\]
where $s = \{s_P \mid P \in \mathcal{P}\}$, $s' = \{s'_P \mid P \in \mathcal{P}\}$.

The property (8) is denoted by $s \xrightarrow{\alpha} s'$, where $\alpha = \alpha_{s_P}$.

A set of states of a DP $\mathcal{P}$ can be considered as a graph in which there is an edge from $s$ to $s'$ labeled by $\alpha_P$ iff $s \xrightarrow{\alpha_P} s'$. The designation $P$ in the label $\alpha_P$ can be omitted.

For each pair $(s, s')$ of states of a DP $\mathcal{P}$
- $s \rightarrow s'$ means that $(s, s')$ belongs to the transition relation, and
- $s \Rightarrow s'$ means that there is a sequence $s_1, \ldots, s_n$ of states of $\mathcal{P}$ such that $s_1 = s$, $s_n = s'$, and $\forall i = 1, \ldots, n-1$ $s_i \rightarrow s_{i+1}$.

A state $s$ of $\mathcal{P}$ is said to be reachable if $0\mathcal{P} \Rightarrow s$.

The set of reachable states of $\mathcal{P}$ is denoted by $\Sigma_P$. $\forall s, s' \in \Sigma_P$, the notation $s \leq s'$ means that $s$ is a path such that $s, s' \in \pi$, and either $s = s'$, or $s$ is
located on $\pi$ to the left of $s'$. The notation $s <_{\pi} s'$ means that $s \leq_{\pi} s'$ and $s \neq s'$. If $\pi$ is clear from the context, then the designation of $\pi$ in $\leq_{\pi}$ and $<_{\pi}$ can be omitted.

2.3.4 Observational equivalence of distributed processes

A concept of an observational equivalence of DPs has the following meaning: DPs $\mathcal{P}$ and $\mathcal{P}'$ are observationally equivalent if for each observer who analyzes an execution of $\mathcal{P}_1$ and $\mathcal{P}'_1$ by observing the contents of $\mathcal{C}$, these DPs are indistinguishable. Let $\mathcal{P}$, $\mathcal{P}'$ be DPs, $s \in \Sigma_{\mathcal{P}_1}$, $s' \in \Sigma_{\mathcal{P}'_1}$, and $\eta : X \to X'$ be a renaming. The notation $s \sim_{\eta} s'$ means that $[s]_{\eta} \subseteq Tm(X)$ and $[s]_{\eta} = \{e^\eta : e \in [s]_{\eta}\}$.

$\mathcal{P}$ and $\mathcal{P}'$ are said to be observationally equivalent if there is a set $\mu$ of triples $(s, s', \eta)$, where $s \in \Sigma_{\mathcal{P}_1}$, $s' \in \Sigma_{\mathcal{P}'_1}$, $s \sim_{\eta} s'$, such that

- $(0_{\mathcal{P}_1}, \eta_{\mathcal{P}_1}, \emptyset) \in \mu$ (where $\emptyset$ is a function with empty domain), and

- $\forall (s, s', \eta) \in \mu$ if $s \to \tilde{s}$ or $s' \to \tilde{s}'$, then $\exists \tilde{s}, \tilde{s}', \tilde{\eta} \in \mu$: $\tilde{\eta}$ is an extension of $\eta$, and $s' \Rightarrow \tilde{s}'$ or $s \Rightarrow \tilde{s}$, respectively.

Note that the above definition is not the only possible definition of an observational equivalence, and can be modified depending on the problem being solved. In some problems, a more appropriate definition of an observational equivalence is a coarsening of the equivalence defined above, such that, for example, the DPs $\mathcal{P} = \{P\}$ and $\mathcal{P}' = \{P'\}$ are equivalent, where $P = \overrightarrow{\text{seq}} \bullet$, $P' = \overrightarrow{\text{seq}} \bullet$, $k \in \bar{X}(P)_K$, $k' \in \bar{X}(P')_K$.

2.4 Preservation theorems for values of formulas

In this section, we formulate and prove theorems about a preservation of values of some formulas under transitions of DPs. These theorems are used for verification of DPs. In examples of applications of these theorems below,

- if names of some channels are secure with respect to $\mathcal{P}_1$, then contents of these channels cannot be changed by $\mathcal{P}_1$, and

- if some keys are secure with respect to $\mathcal{P}_1$, then contents of EMs encrypted with these keys are inaccessible to $\mathcal{P}_1$.

2.4.1 Secure channel theorems

First theorem is related to a preservation of values of formulas of the form

$$E \perp_{C} P,$$

under transitions of DPs. This theorem can be interpreted as the following statement: if $\mathcal{P}$ is a DP, $P \in \mathcal{P}$, there is no messages in $E$ which are available for $P$, and in a current state of $\mathcal{P}$ there are no messages from $E$ in channels available to $P$, then no own activity of $P$ will lead to availability for $P$ messages from $E$. Channels whose names are occurred in $E$ can be interpreted as channels secure with respect to $P$.

**Theorem 1.**

Let $\mathcal{P}$ be a DP, $P \in \mathcal{P}$, and $s, s'$ be states of $\mathcal{P}$ such that $s \xrightarrow{e} s'$.

Then $\forall E \subseteq (P)_0$ the following implication holds:

$s \models E \perp_{C} P \Rightarrow s' \models E \perp_{C} P$.

**Proof.**

$s \models E \perp_{C} P$ means that $\forall e \in E Agent(P) \notin e,$ and $\forall x \in E_X, \forall c \in Tm_C$

$$\forall y \in [P]_s, x \notin y \iff$$

if $\exists e \in [c]_s : x \in e$, then $c \in E$

$$\bigg\{ \bigg\}$$

(10)

Prove that (10) implies that $s' \models E \perp_{C} P$, i.e.

$\forall x \in E_X, \forall c \in Tm_C$

$$\forall y \in [P']_{s'}, x \notin y \iff$$

if $\exists e \in [c]_{s'} : x \in e$, then $c \in E$

$$\bigg\{ \bigg\}$$

(11)

1. If first statement in (11) is false, then first statement in (10) implies $[P]_s \neq [P']_{s'}$, and one of the following two cases holds.

- First case:

  $$\alpha = c? c, \text{ where}$$

  $$c \in (P)_s, e^c \in [c]_s,$$

  $$[P']_{s'} = [P]_s \cup Var(e),$$

  $$\exists x \in E_X, \exists y \in Var(e) : x \in y'. \bigg\{ \bigg\}$$

(12)
The statement \( x \in y^s \subseteq e^s \subseteq [c]^s \) and second statement in (10) imply \( c^s \in E \).

If \( c^s \notin E_X \), then \( c \) has the form \( \text{shared}_\text{channel}(...) \), and in this case, the definition of an execution of a SP in 2.2.5 implies that \( \text{Agent}(P) \in c^s \). But this fact and the statement \( c^s \in E \) contradict the assumption \( \forall e \in E \ Agent(P) \notin e \).

Thus, \( c^s \in E_X \), that implies \( c \in [P]^s \). According to first statement in (10) (in which we take \( c^s \) and \( c \) as \( x \) and \( y \), respectively), the statement \( c^s \notin \) holds, but this is false.

- **Second case:**

  \[
  \begin{align*}
  \alpha &= (e := e'), \quad \text{where} \\
  e' &\in (P)^s \cup [c]^s, \\
  [P]^s_{e'} &= [P]_s \cup \text{Var}(e). \\
  \exists x \in E_X, \exists y \in \text{Var}(e) : x \in y^{s'}
  \end{align*}
  \]

  (13)

  Since \( x \in y^{s'} \subseteq e^{s'} = (e')^s \in (P)^s \), then

  \[
  \exists z \in [P]^s : x \in z^s.
  \]  

  (14)

  (14) contradicts first statement in (10).

2. If second statement in (11) is false, i.e.

  \[
  \exists x \in E_X, \exists e' \in Tm_C, \exists [c']^s : x \in e', \quad \text{but} \quad c' \notin E
  \] 

  then second statement in (10) implies that \( [c']^s \notin [c]^s \), and

  \[
  \alpha = cle, \quad \text{where} \quad c, e \in [P]^s, \quad x \in e' = e^s.
  \]

  From \( x \in e^s \) it follows that \( \exists y \in [P]^s : x \in y^s \), which contradicts first statement in (10).

The following theorem is a strengthening of theorem 1. It states that under conditions of theorem 1 the lower and upper bounds on the contents of secure channels do not change when actions of \( P \) are executed.

**Theorem 2.**

Let \( P \) be a DP, \( P \in \mathbb{P} \), and \( s, s' \) be states of \( P \) such that \( s \xrightarrow{c} s' \).

Then \( \forall E \subseteq (P)^0, \ E', E'' \subseteq Tm, \ c \in E_C \) the following implication holds:

\[
\begin{align*}
& s \models \varphi \Rightarrow s' \models [c] \subseteq E''.
& \varphi = \{ E \perp_{-} P, E' \subseteq [c] \subseteq E'' \}.
\end{align*}
\]

**Proof.**

According to theorem 1 \( s \models E \perp_{C} P \) implies \( s' \models E \perp_{C} P \). In addition, \([c]^s \subseteq [c]^s' \). Thus, to prove the theorem, it suffices to prove implication \( s \models \varphi \Rightarrow s' \models [c] \subseteq E'' \).

If the conclusion of implication (15) does not hold, then \([c]^s \neq [c]^s' \).

By assumption \( c \in E_C \subseteq (P)^0 \), so \([c]^s \neq [c]^s' \) is possible only if \( \alpha = c'e \), where \( (c')^s = c \).

If \( c \) is not a variable, then \( c \) is a shared channel, and by definition of an execution the action \( c'e \), in this case the condition \( \text{Agent}(P) \in c \) holds, which contradicts the assumption \( s \models E \perp_{C} P \) (because, in particular, \( \text{Agent}(P) \) has no occurrences in terms from \( E \)). Thus, \( c \in \text{Var}, \ c' \in \text{Var}, \ c' \in [P]^s \).

Since \( c \in E_X \) and \( c' \in [P]^s \), then by assumption \( s \models E \perp_{C} P \) we have \( c \notin c \), which is false. 

**2.4.2 Secure key theorems**

In this subsection, we prove theorems similar to theorems 1 and 2. Now we consider secure keys instead of secure channels.

First theorem is related to a preservation of values of formulas

\[
E \perp_{K} P, \quad \text{where} \quad E \subseteq Tm, \quad \text{and} \quad P \text{ is a SP} \quad (16)
\]

under transitions of DPs. This theorem can be interpreted as the following statement: if \( P \) is a DP, \( P \in \mathbb{P} \), there is no messages in \( E \) which are available for \( P \), and in a current state of \( P \) holds, then no own activity of \( P \) will lead to availability for \( P \) keys from \( E \). These keys can be interpreted as secure keys with respect to \( P \).

**Theorem 3.**

Let \( P \) be a DP, \( P \in \mathbb{P} \), and \( s, s' \) be states of \( \mathbb{P} \) such that \( s \xrightarrow{c} s' \).

Then \( \forall E \subseteq (P)^0 \) the following implication holds:

\[
\begin{align*}
& s \models E \perp_{K} P \Rightarrow s' \models E \perp_{K} P.
\end{align*}
\]

**Proof.**

\[
\begin{align*}
& s \models E \perp_{K} P \text{ means that} \forall e \in E \ Agent(P) \notin e, \text{ and} \\
& \forall x \in E_X, \forall e \in Tm_C
\end{align*}
\]

(17)
Prove that (17) implies that \( s' \models E \perp e \), i.e. 
\[
\forall x \in E_K, \forall c \in Tm_C
\]
\[
\forall y \in [P]_s', s' \models x \perp_{K,E} y' \}
\]
\[
\forall e \in [c]_s', s' \models x \perp_{K,E} e \}
\]
(18)

1. If first statement in (18) is wrong, then first statement in (17) implies \([P]_s \neq [P]_s'\), and one of two cases does hold:

- First case: the following statement holds

\[
\exists \alpha = c' \, e, e \in \langle P \rangle_s, e' \in [e']_s, [P]_{s'} = [P]_s \cup \text{Var}(e),
\exists y \in \text{Var}(e),
\exists \text{an occurrence of } x \text{ in } y'\text{ that is not contained in any subterm of the form } k(\ldots) \subseteq y', \text{ where } k \in E_K
\]
(19)

Since the occurrence of \( x \) mentioned in (19) occurs in the term \( y'' \subseteq e' \subseteq [e']_s \), then second statement in (17) implies that this occurrence of \( x \) occurs in the subterm \( k(\tilde{e}) \subseteq e' \), where \( k \in E_K \).

(19) implies that \( k(\tilde{e}) \) is not a subterm of \( y'' \). Since the terms \( k(\tilde{e}) \) and \( y'' \) have a non-empty intersection (both contain the above occurrence of \( x \)), then (11) implies that \( y'' \subseteq k(\tilde{e}) \). Thus,

\[
y'' \subseteq k(\tilde{e}) \subseteq e'.
\]
(20)

Prove by induction on the structure of \( e \) that (20) implies

\[
\exists z \in \text{Var}(e) : k(\tilde{e}) \subseteq z'' \subseteq e'.
\]
(21)

If \( e \in \text{Con} \cup \text{Var} \), then (21) holds.

If \( e = f(e_1, \ldots, e_n) \), where \( f \in \text{Fun} \), then

- if \( f = \text{encrypt} \), i.e. \( e = k_1(e_1) \), then

\( k_1 \in \text{Keys}(e) \), and (3)(b) implies the inclusion \( \text{Keys}(e) \subseteq [P]_s \), thus

\( k_1 \in [P]_s \), and the following cases are possible:

* \( k(\tilde{e}) = e'' = k_1^*(e'_1) \), in this case \( k = k_1^* \in [P]_s \), but since \( k \in E_K \), then, according to first statement in (17), the occurrence of \( k \) in \( k \) occurs in a subterm of the form \( k(\ldots) \subseteq k \), which is impossible,

* \( k(\tilde{e}) \subseteq k_i^* \), this case is impossible by the definition of terms of the type \( K \).

* \( k(\tilde{e}) \subseteq e'_1 \), in this case statement (21) follows from the inductive hypothesis,

- if \( f \neq \text{encrypt} \), then \( \exists i \in \{1, \ldots, n\} : k(\tilde{e}) \subseteq e'_i \), and statement (21) follows from inductive hypothesis.

From (20) and (21) it follows that

\[
y'' \subseteq k(\tilde{e}) \subseteq z'' \subseteq e'.
\]
(22)

Thus, the term \( e \) contains occurrences of the variables \( y \) and \( z \) with the following property: \( y'' \subseteq z'' \), whence for these occurrences the inclusion \( y \subseteq z \) holds, which is impossible.

- Second case: the following statement holds

\[
\exists \alpha = (e := e'), e' \in \langle P \rangle_s, e'' = (e')^s, [P]_{s'} = [P]_s \cup \text{Var}(e),
\exists y \in \text{Var}(e),
\exists \text{an occurrence of } x \text{ in } y'\text{ that is not contained in any subterm of the form } k(\ldots) \subseteq y', \text{ where } k \in E_K
\]
(23)

Since the occurrence of \( x \) mentioned in (23) is contained in the term \( y'' \subseteq e'' = (e')^s \), then this occurrence of \( x \) is contained in the subterm \( (z')^s \subseteq (e')^s \), where \( z' \in \text{Var}(e') \).

By assumption, \( e' \in \langle P \rangle_s \), therefore \( \text{Var}(e') \subseteq [P]_s \), so \( z' \in [P]_s \). From first statement in (17) it follows that the occurrence of \( x \) in \( (z')^s \) mentioned in (23) is contained in some subterm \( k(\tilde{e}) \subseteq (z')^s \), where \( k \in E_K \).

From (23) it follows that \( k(\tilde{e}) \) is not a subterm of \( y'' \).

Since the terms \( k(\tilde{e}) \) and \( y'' \) have a non-empty intersection (both contain the occurrence of \( x \) mentioned in (23)), then from (11) it follows that \( y'' \subseteq k(\tilde{e}) \).

The equality \( e' = e'' \) implies that \( \exists z \in \text{Var}(e) : \text{the above occurrence of } z' \text{ in} \)}
Let \( P \) be a DP such that \( \text{Var}(P)_C = \{\bar{o}\} \), \( P \in \mathcal{P} \), \( s, s' \) be states from \( \Sigma_P \) such that \( s \xrightarrow{\alpha} s' \), and \( E \subseteq (P)_0 \). Then \( \forall k \in E_K \) the following implication holds:

\[
\alpha \Vdash \varphi \Rightarrow s' \Vdash \varphi
\]

where \( \varphi = \{E \perp_K P, k^{-1}[P] \subseteq k^{-1}[\bar{o}]\} \).

**Proof.**

By theorem 3, the statement \( s \Vdash E \perp_K P \) implies \( s' \Vdash E \perp_K P \). Thus, to prove theorem 4 it suffices to prove the implication

\[
s \Vdash \varphi \Rightarrow s' \Vdash k^{-1}[P] \subseteq k^{-1}[\bar{o}]. \tag{26}
\]

If (26) does not hold, then \( \alpha \) is not a sending, \( [P]_{s'} = [P]_s \cup \text{Var}(e') \), and \( \exists e \in k^{-1}[P] \) such that

\[
e \not\subseteq k^{-1}[\bar{o}], \text{i.e. } \exists x \in [P]_{s'} : k(e) \subseteq x, \forall \bar{e} \in [\bar{o}], k(e) \not\subseteq \bar{e}. \tag{27}
\]

The assumption \( s \Vdash \varphi \) and (27) imply that \( x \in \text{Var}(e') \), whence we get: \( k(e) \subseteq x \subseteq (e')^s \).

Consider separately each of the two possible types of \( \alpha \).

1. \( \alpha = ?e' \), in this case \( k(e) \subseteq (e')^s \in [\bar{o}]_s \). Setting in (27) the term \( \bar{e} \) be equal to \( (e')^s \), we get a contradiction.

2. \( \alpha = (e' := e'') \), in this case \( e'' \in (P)_s \), \( (e')^s = (e'')^s \), so

\[
k(e) \subseteq (e'')^s. \tag{28}
\]

Prove that \( k \not\subseteq e'' \) and \( k \in E_X \).

Suppose \( k \in e'' \). If \( k = \text{shared_key}(...) \), then, according to the definition of SP execution in section 2.2.5, the statement \( \text{Agent}(P) \in E_K \) holds, which contradicts first condition of the property \( s \Vdash E \perp_K P \). Recall that this property has the form: \( \forall \bar{e} \in E \text{ Agent}(P) \not\subseteq \bar{e} \), and

\[
\forall x \in E_X, \forall y \in [P]_s, x \perp_{K,E} y^s
\]

\[
\forall x \in E_X, \forall \bar{e} \in [\bar{o}], x \perp_{K,E} \bar{e}
\]

Therefore, \( k \in E_X \), so \( k \in e'' \in (P)_s \) implies \( k \in [P]_s \). However, assuming in first statement in (29) \( x \) and \( y \) be equal to \( k \), we get \( k \perp_{K,E} k \), which is false by definition (15).

Similarly to the proof of the implication (26) \( \Rightarrow (29) \) in theorem 3, we prove that the statements \( k \not\subseteq e'' \) and (28) imply

\[
\exists y \in \text{Var}(e'') \subseteq [P]_s : k(e) \subseteq y^s. \tag{30}
\]

By assumption, \( s \Vdash k^{-1}[P] \subseteq k^{-1}[\bar{o}] \), i.e. \( k^{-1}[P] \subseteq k^{-1}[\bar{o}] \). From (30) it follows that \( e \in k^{-1}[P] \). Therefore, \( e \in k^{-1}[\bar{o}] \), which contradicts the assumption (27).

The following theorem is a strengthening of theorem 4. It states that under the conditions of theorem 4 the lower and upper bounds for the set of EMs contained in open channel and encrypted with secure keys do not change when actions of the SP \( P \) are executed.

**Theorem 5.**

Let \( P \) be a DP such that \( \text{Var}(P)_C = \{\bar{o}\} \), \( P \in \mathcal{P} \), \( s, s' \) be states from \( \Sigma_P \) such that \( s \xrightarrow{\alpha} s' \), and \( E \subseteq (P)_0 \), \( E', E'' \subseteq \mathcal{T}m \).
Then \( \forall k \in E_K \) the following implication holds:

\[
\begin{align*}
  s \models \varphi & \Rightarrow s' \models \varphi, \text{ where} \\
  \varphi & = \left\{ \begin{array}{l}
  E \perp_K P, k^{-1}[P] \subseteq k^{-1}[\varnothing] \\
  E' \subseteq k^{-1}[\varnothing] \subseteq E''
  \end{array} \right\}. 
\end{align*}
\]  

(31)

**Proof.**

By theorem \( k \) to prove (31) it is enough to prove that

\[
\begin{align*}
  s \models \varphi & \Rightarrow s' \models \left\{ E' \subseteq k^{-1}[\varnothing], k^{-1}[\varnothing] \subseteq E'' \right\}. 
\end{align*}
\]  

(32)

- The statement \( s' \models \left\{ E' \subseteq k^{-1}[\varnothing] \right\} \) follows from \( [\varnothing]_{s} \subseteq [\varnothing]_{s'} \).
- Prove the statement \( s' \models k^{-1}[\varnothing] \subseteq E'' \). If it is not true, then \( [\varnothing]_{s} \neq [\varnothing]_{s'} \). This is possible only if \( \alpha = \ell e \), where \( e \in \{P\}_{s} \), and

\[
\[\varnothing]_{s'} = [\varnothing]_{s} \cup \{ e^{s} \},
\]  

\[\exists e' \in (E'')_{s} \ni k^{-1}[\varnothing] : k(e') \subseteq e^{s}. \]

As in theorem \( k \) we prove that \( k \in E_{X} \), and \( e \in \{P\}_{s} \) implies \( e \not\in e \).

Similarly to the proof of implication (20) \( \Rightarrow \) (21) in theorem \( k \), we can prove that the statements \( k \not\in e \) and \( k(e') \subseteq e^{s} \) imply

\[
\exists x \in \text{Var}(e) \subseteq \{P\}_{s} : k(e') \subseteq x^{s},
\]

therefore \( e' \in k^{-1}[P]^{s} \). Hence, using the assumption \( s \models \varphi \), which results to the inclusion \( k^{-1}[P]^{s} \subseteq k^{-1}[\varnothing] \), we get: \( e' \in k^{-1}[\varnothing] \), which contradicts (33). \( \blacksquare \)

### 2.5 Theorem for proving a correspondence property

A theorem stated in this section can be used to prove a correspondence property of authentication protocols, which has the following meaning: if one of participants of an authentication protocol, after executing this protocol, has come to the conclusion that declared name and parameters of other participant of this protocol are authentic, then then this is indeed the case. A theorem proved below is used to prove that if

- a DP \( P \) uses only open channel \( o \) for communication, and

- in some state \( s \in \Sigma_{P} \) this channel contains a message containing a subterm \( k(e) \), where the key \( k \) is secure in \( s \) with respect to \( P \in \mathcal{P} \),

then in some state \( s^{'<} s \) another SP \( P' \in \mathcal{P} \) sent a message to \( o \), containing the subterm \( k(e) \).

In sections 3.2 and 3.3 we consider examples of applying this theorem to verification of the Yahalom CP, and to verification of the CP of EMs passing between several agents.

**Theorem 6.**

Let \( P \) be a DP, \( \text{Var}(P)_{C} = \{\varnothing\} \), \( P \in \mathcal{P} \), \( E \) be a subset of \( \{P\}_{0} \) and \( s \in \Sigma_{P} \) be a state such that

- \( s \models E \perp_K P \), and

- \( [\varnothing]_{s} \) contains a term with a subterm \( k(e) \), where \( k \in E_{K} \).

Then, for each path \( \pi \) from \( 0_{P} \) to \( s \), there is a SP \( P' \in \mathcal{P} \setminus \{P\} \) such that \( \pi \) has an edge of the form

\[
\dot{s} \xrightarrow{(\ell e)^{s}} s', \quad \text{where} \quad k(e) \subseteq e^{s}. \]  

(34)

**Proof.**

Let \( s' \) be a first state on \( \pi \) such that \( [\varnothing]_{s'} \) has a term \( e' \) with the subterm \( k(e) \). Since \( [\varnothing]_{0} = \varnothing \), then \( s' \neq 0 \).

Let \( \dot{s} \xrightarrow{\alpha^{s}} s' \) be an edge on \( \pi \) ending at \( s' \). Since \( e' \notin [\varnothing]_{s'} \), then \( \alpha = \ell e \), where \( \bar{e}^{s} = e'. \) If \( P' \neq P \), then the theorem is proven.

Prove that another possible case \( (P' = P) \) is impossible.

Suppose \( P' = P \), i.e. \( \dot{s} \xrightarrow{(\ell e)^{s}} s' \).

Prove that \( k \in E_{X} \). If \( k \notin E_{X} \), i.e. \( k \) is a shared key, then, by the definition of an execution of a SP in \( 2.2.3 \), \( \text{Agent}(P) \subseteq k \in E_{K} \), which contradicts the assumption \( \forall \dot{e} \in E \text{ Agent}(P) \not\subseteq \dot{e} \).

The statement \( k \notin [P]_{s} \) and condition \( \dot{e} \in (P)_{s} \) (which is true according to (4)(a)) imply \( k \notin \dot{e} \).

Similarly to the proof of the implication (20) \( \Rightarrow \) (21) in theorem \( k \), we can prove that statements \( k(e) \subseteq e' = \bar{e}^{s} \) and \( k \not\in \dot{e} \) imply

\[
\exists x \in \text{Var}(\dot{e}) \subseteq [P]_{\dot{e}} : k(e) \subseteq x^{s} \in [P]_{\dot{e}}. \]  

(35)
Let $s''$ be a first state on $\pi$ such that $[P]s''$ has a term with the subterm $k(e)$, that is,

$$\exists x \in [P]s'' : k(e) \subseteq x^{s''}. \quad (36)$$

implies that $s''$ is to the left of $s'$ on $\pi$. It is clear that $s'' \neq 0$, so there is an edge $\tilde{s} \xrightarrow{s''} s''$ on $\pi$. From the choice of $s''$ it follows that $x \notin [P]s$, thus $P'' = P$, and two cases are possible:

1. $\alpha = \tilde{s}, x \in Var(\tilde{e}), \tilde{e}^{s''} \in [\tilde{o}]_{\tilde{s}}$,
   since $k(e) \subseteq x^{s''} \subseteq \tilde{e}^{s''} \in [\tilde{o}]_{\tilde{s}}$, then we get a contradiction with the choice of $s'$ as the first state on $\pi$ such that $[\tilde{o}]_{\tilde{s}}$ contains the term $e'$ with the subterm $k(e)$: the state $\tilde{s}$ has the same property, and is located to the left of $s'$.

2. $\alpha = (\tilde{e} := \tilde{e}), x \in Var(\tilde{e}), \tilde{e} \in (P)_{\tilde{s}}, \tilde{e}^{s''} = \tilde{e}^\bullet$
   since
   
   - $k(e) \subseteq x^{s''} \subseteq \tilde{e}^{s''} = \tilde{e}^\bullet$
   - $\tilde{e}$ does not contain $k$, because it was proven above that $k \notin [P]_{\tilde{s}}$, therefore, taking into account the property $\tilde{s} \leq \tilde{s}$, which implies the inclusion $[P]_{\tilde{s}} \subseteq [P]_{\tilde{s}}$, we get: $k \notin [P]_{\tilde{s}}$, and therefore the term $\tilde{e} \in (P)_{\tilde{s}}$ also does not contain $k$.

then, similarly to the proof of the implication $\text{(20)} \Rightarrow \text{(21)}$ in theorem 3, we can prove that

$$\exists y \in [P]_{\tilde{s}} : k(e) \subseteq y^\bullet,$$

which contradicts the choice of $s''$ as a first state on $\pi$ with the property $\tilde{s}$: $\tilde{s}$ has the same property and is located to the left of $s''$.

\[\blacksquare\]

### 2.6 Diagrams of distributed processes

#### 2.6.1 Prefix sequential processes

A SP $P$ is said to be a prefix SP if

$$P = 0 \xrightarrow{a_1} 1 \ldots n-1 \xrightarrow{a_n} n \xrightarrow{P'} \quad (37)$$

i.e. $P$ contains nodes numbered by natural numbers $0, 1, \ldots, n$ ($n \geq 1$), and $Init(P) = 0, \forall i = 0, \ldots, n-1$ there is exactly one outgoing edge from node $i$ with the end $i + 1$ and labeled by $a_i$. The notation $P'$ in (37) denotes a subgraph of the graph $P$, consisting of the nodes and edges of the graph $P$, with the exception of the nodes $0, \ldots, n - 1$ and edges associated with these nodes.

The subgraphs $0 \xrightarrow{a_1} 1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} n$ and $P'$ of the graph (37) are called a prefix and a postfix of the SP $P$, respectively, and are denoted by $Pref(P)$ and $Post(P)$ respectively. The last node of $Pref(P)$ is called a final node of this prefix. If $Post(P)$ consists of one node, then it is denoted by $0$.

If a SP $P$ has the form (37), then we will denote this fact by

$$P = a_1; \ldots; a_n; P'. \quad (38)$$

#### 2.6.2 A concept of a diagram of a distributed process

Let $\mathcal{P}$ be a DP, such that each $P \in \mathcal{P}$ is a prefix SP, and for each sending (each receiving) in $Pref(P)$ it is assumed that

- an intended receiver (an intended sender) of the message that is sent (received) when performing this action is some $P' \in \mathcal{P}$, and
- an action of the SP $P'$ corresponding to the receiving (sending) of this message is in $Pref(P')$.

These dependencies between the actions can be expressed by a diagram of the DP $\mathcal{P}$, which has the following form:

- each SP $P \in \mathcal{P}$ is represented by a thread in this diagram, i.e. by a vertical line on which points corresponding to nodes of $Pref(P)$ are marked, the upper point corresponds to $Init(P)$, and
  - each point has the number of the corresponding node,
  - a name $P$ of the SP $P$ is indicated near the upper point,
  - if $Post(P) = P' \neq 0$, then $P'$ is indicated at the bottom point,
  - near to each segment $l$ connecting adjacent points on the thread, there is a label $a_l$ of an edge from $Pref(P)$ corresponding to $l$,
- for each segment \( l \) connecting adjacent points of the thread, if \( \alpha_i \) is a sending, \( \alpha_{i'} \) is an intended receiving for \( \alpha_i \), then the diagram contains an arrow, a start of which lies on \( l \), and an end of which lies on \( l' \).

For example if \( P_1 = \alpha_1; \ldots \alpha_n; P'_1 \), where \( \alpha_1 \) is a sending, and \( \alpha_n \) is a receiving, then SP \( P_1 \) corresponds to a thread

\[
\begin{array}{c}
0 \quad P_1 \\
\alpha_1 \\
\vdots \\
n-1 \\
\alpha_n \\
n \quad P'_1
\end{array}
\]

\[
(39)
\]

Note that the arrows depict only the desired connection between sendings and receivings, but they have no relation with real communication: it is possible that the sent message will be received by a SP which is different from a SP to which it was intended.

For the sake of greater clarity, we will use the following convention in the notation of variables:

- we will indicate a horizontal bar above a designation of a variable \( x \) if \( x \in \hat{X}(P) \) (i.e. this variable is denoted by \( \hat{x} \)),

- if \( P \) is a SP of the form \( \{A,B\}, x \in \hat{X}(P) \), and \( i \) is the first index such that \( x \in \alpha_i \) (i.e. \( \forall i' = 1, \ldots, i - 1 \ x \not\in \alpha_{i'} \)), then occurrences of \( x \) in the label \( \alpha_i \) of the \( i \)-th segment of the thread of \( P \) are denoted by \( \hat{x} \).

These variable designations also will be used in notations of the form \( \{A,B\} \).

### 2.6.3 Examples of diagrams of distributed processes

1. First example is DP \( \mathcal{P}_1 = \{A,B\} \), which is a model of a transmission from \( A \) to \( B \) a message \( x \) through a channel \( c_{AB} \), where only \( A \) and \( B \) know the name of this channel, i.e.

\[
c_{AB} = \text{shared\_channel}(A,B).
\]

This DP works as follows:

- \( A \) sends the message \( x \) to \( c_{AB} \),
- \( B \) receives this message from \( c_{AB} \) and writes it to the variable \( y \), after which it behaves like SP \( P \).

SPs \( A \) and \( B \) are defined as follows:

\[
A = c_{AB}!x; 0, \ B = c_{AB}?y; P.
\]

A diagram of \( \mathcal{P}_1 \) has the following form:

\[
\begin{array}{c}
0 \quad A \\
1 \quad P
\end{array}
\]

\[
c_{AB}!x \\
c_{AB}?y
\]

(40)

2. Second example is DP \( \mathcal{P}_2 = \{A,B\} \), which is a model of transmission from \( A \) to \( B \) EM \( k_{AB}(x) \) through an open channel \( \circ \). It is assumed that \( A \) and \( B \) have a shared secret key \( k_{AB} \), on which they can encrypt and decrypt messages using a symmetric encryption system, and only \( A \) and \( B \) know \( k_{AB} \), i.e.

\[
k_{AB} = \text{shared\_key}(A,B).
\]

This DP works as follows:

- \( A \) sends \( \circ k_{AB}(x) \) to \( \circ \),
- \( B \) receives \( \circ k_{AB}(x) \) from \( \circ \), decrypts it, writes the extracted message \( x \) to variable \( y \), after which it behaves like SP \( P \).

SPs \( A \) and \( B \) are defined as follows:

\[
A = \circ k_{AB}(x); 0, \ B = \ast k_{AB}(y); P.
\]

A diagram of \( \mathcal{P}_2 \) has the following form:

\[
\begin{array}{c}
0 \quad A \\
1 \quad P
\end{array}
\]

\[
\circ k_{AB}(x) \\
\ast k_{AB}(y)
\]

(41)

3. Third example is DP \( \mathcal{P}_3 = \{A,B,J\} \), which is a model of transmission from \( A \) to \( B \) a message \( x \) over secret channel \( \hat{c} \) with use of a trusted intermediary \( J \), where \( A \) and \( J \) (\( B \) and \( J \)) interact via channel \( c_{AJ} (c_{BJ}) \), and only \( A \) and \( J \) \((B \) and \( J \)) know the name \( c_{AJ} \) \((c_{BJ}) \), i.e.

\[
c_{AJ} = \text{shared\_channel}(A,J),
\]

\[
c_{BJ} = \text{shared\_channel}(B,J).
\]

This DP works as follows:
4. Fourth example is $\mathcal{P}_4 = \{A, B, J\}$ (called Wide-Mouth Frog (WMF) protocol), which is a model of a transmission from $A$ to $B$ EM $k(x)$ through open channel $\circ$ with use of a trusted intermediary $J$, with whom $A$ and $B$ interact through $\circ$. $A$ creates a secret key $k$, sends $J$ this encrypted key for $B$, and then sends $B$ EM $k(x)$.

It is assumed that $A$ and $J$ ($B$ and $J$) have a shared secret key $k_AJ$ ($k_BJ$), on which they can encrypt and decrypt messages using a symmetric encryption system, and only $A$ and $J$ ($B$ and $J$) know $k_AJ$ ($k_BJ$), i.e.

$$k_AJ = shared\_key(A, J),$$
$$k_BJ = shared\_key(B, J).$$

This DP works as follows.

- $A$ creates a secret key $\bar{k}$ (at first only $A$ knows this key) and sends $J$ EM $k_AJ(\bar{k})$ through $\circ$, then $A$ sends $B$ EM $\bar{k}(x)$,
- $J$ receives a message from $A$, decrypts it, then encrypts the extracted key $\bar{k}$ with the key $k_BJ$, and sends $B$ the EM $k_BJ(\bar{k})$,
- $B$ extracts the key $\bar{k}$ from the received message from $J$, and then uses this key to extract the message $x$ from the received message from $A$, writes it to the variable $y$, and then behaves like SP $P$.

SPs $A$, $B$ and $J$ are defined as follows:

$$\begin{align*}
A &= \alpha_1; \alpha_2; 0, \quad \alpha_1 = k_AJ(\bar{k}), \quad \alpha_2 = \bar{k}(x), \\
J &= j_1; j_2; 0, \quad j_1 = k_AJ(\hat{u}), \quad j_2 = k_BJ(u), \\
B &= \beta_1; \beta_2; P, \quad \beta_1 = k_BJ(\bar{v}), \quad \beta_2 = \bar{v}(\hat{y}).
\end{align*}
$$

A diagram of $\mathcal{P}_4$ has the form (43).

2.7 Transition graphs of distributed processes

In this section, we consider DPs consisting of a finite number of SPs. For a visual representation of an execution of such DPs, the concept of transition graph of a DP is introduced. An execution of a DP can be presented as a walk in a GP corresponding to this DP.

Below in this section, the symbol $P$ denotes a DP consisting of a finite number of SPs, each of which is different from $P_i$.

2.7.1 A concept of a transition graph of a distributed process

Let $P$ be a DP of the form $\{P_1, \ldots, P_n\}$. A transition graph (TG) of $P$ is a graph $G_P$, where

- each node of $G_P$ is a list $at = (at_1, \ldots, at_n)$, where $\forall i = 1, \ldots, n$ $at_i \in P_i$,
- each edge of $G_P$ has the form

$$(at_1, \ldots, at_n) \xrightarrow{\alpha} (at_1', \ldots, at_n'),$$

where $i \in \{1, \ldots, n\}$, $P_i$ has edge $at_i \xrightarrow{\alpha} at_i'$, and $at_{i'} = at_{i'}'$ for $i' \neq i$.

A node $Init(G_P) = (Init(P_1), \ldots, Init(P_n))$ of $G_P$ is said to be initial. $
\forall s \in \Sigma_P$, the component $at_s$ of $s$ can be considered as a node of $G_P$. 

18
Let there is given an execution of a DP \( \mathcal{P} \), and \( s_0, s_1, \ldots, s_n \) be a sequence of states generated in this execution. From the definition of a transition relation \( \mathcal{R} \) it follows that this sequence corresponds to a walk in TG \( G_\mathcal{P} \), which can be considered as a representation of the execution of \( \mathcal{P} \):

\[
\text{Init}(G_\mathcal{P}) = at_{s_0} \xrightarrow{\alpha_1} at_{s_1} \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} at_{s_n}
\]

Recall that \( \mathcal{P}_1 = \{ \mathcal{P}, \mathcal{P}_1 \} \). The TG \( G_\mathcal{P}_1 \) can be considered as a graph obtained from \( G_\mathcal{P} \) by adding edges at \( \alpha \) at, where \( \alpha \in \mathcal{A} \).

A node \( at \in G_\mathcal{P} \) is said to be reachable if for some \( s \in \Sigma_\mathcal{P}_1 \), \( at = at_s \).

An edge of \( G_\mathcal{P} \) is said to be realizable, if it lies on a path corresponding to some execution of \( \mathcal{P}_1 \).

The following conventions will be used in graphical representation of TGs:

- each node \( at = (at_1, \ldots, at_n) \) of a TG is represented by an oval, with a list \( at_1 \ldots at_n \) of components of \( at \) inside this oval,
- an initial node is represented by a double oval,
- a black circle on an edge of a TG means that this edge is unreachable (this unrealizability should be justified by special reasoning),
- in order to abbreviate notations, a label of an edge \( \alpha \) of a TG can be denoted simply by the action \( \alpha \) in this label (without specifying the SP \( \mathcal{P} \) performing the action \( \alpha \) on this transition).

### 2.7.2 Examples of transition graphs of distributed processes

In this section we present TGs for DPs defined in 2.6.3. We use the following convention: if \( A \) is a name of a SP occurred in some of these DPs, and \( i \) is a number of a point on a thread corresponding to this SP, then the node of the graph \( A \) corresponding to this point is denoted by \( A^i \).

1. TGs for \( \mathcal{P}_1, \mathcal{P}_2 \) described by (40), (41), have the form

\[
\begin{array}{c}
A^0 B^0 \xrightarrow{c_{AB}} \cdots \\
A^1 B^0 \xrightarrow{c_{AB}} \cdots
\end{array}
\]

(45)

2. A TG for DPs \( \mathcal{P}_3, \mathcal{P}_4 \), described by (43) has the form (47).

\[
\begin{array}{c}
A^0 B^0 \xrightarrow{c_{AB}} \cdots \\
A^1 B^0 \xrightarrow{c_{AB}} \cdots
\end{array}
\]

(46)

where slant arrows indicate edges of the TGs outgoing from the corresponding nodes, as well as parts of the TGs that are reachable after traversing those edges that are not represented in this graphical representation, this convention will also be used in other TG examples.

### 2.7.3 Validity of formulas at nodes of transition graphs of distributed processes

Let \( \mathcal{P} \) be a DP. \( \forall at \in G_\mathcal{P}, \forall \varphi \in \mathcal{F}m \), the notation \( at \models \varphi \) denotes the statement

\[
\varphi \text{ is true at the node } at.
\]

(48) is valid iff \( \forall s \in \Sigma_\mathcal{P}_1 \), \( at_s = at \Rightarrow s \models \varphi \).

It is easy to prove that TGs \( G_\mathcal{P}_1 \), where \( \mathcal{P}_1 (i = 1, \ldots, 4) \) are DPs defined in section 2.6.3, have the following properties:

\[
\begin{align*}
\text{Init}(G_\mathcal{P}_1) &\models \{ \varphi_1, [c_{AB}] = \emptyset \} \\
\text{Init}(G_\mathcal{P}_2) &\models \{ \varphi_2, k_{AB}^{-1}[c] = \emptyset \} \\
\text{Init}(G_\mathcal{P}_3) &\models \{ \varphi_3, [c_{AB}] = \emptyset, [c_{AB}] = \emptyset \} \\
\text{Init}(G_\mathcal{P}_4) &\models \{ \varphi_4, k_{AB}^{-1}[c] = \emptyset, k_{AB}^{-1}[c] = \emptyset \}
\end{align*}
\]

where

\[
\begin{align*}
\varphi_1 &\models \{ c_{AB} \downarrow [P] \} \\
\varphi_2 &\models \{ k_{AB}^{-1}[P] \subseteq k_{AB}^{-1}[c] \} \\
\varphi_3 &\models \{ c_{AB}, c_{AB}, c \downarrow [P] \} \\
\varphi_4 &\models \{ k_{AB}^{-1}[P] \subseteq k_{AB}^{-1}[c], k_{AB}^{-1}[P] \subseteq k_{AB}^{-1}[c] \}
\end{align*}
\]

Indeed, for any DP \( \mathcal{P} \), a state \( s \in \Sigma_\mathcal{P} \), the property \( at_s = \text{Init}(G_\mathcal{P}) \) if \( s = 0 \), or there is a path from 0 to \( s \) with edge labels of the form \( \alpha P_i \), and for each formula \( \psi_i = \{ \varphi_i, \ldots \} \) in (49).
3 Verification of cryptographic protocols

Methods of verification of CPs presented in this section are based on a representation of CPs in the form of DPs. To prove properties of DPs, we use theorems from previous section. First method of verification of CPs described below is based on the concept of a TG, and the second method is based on theorem 3 and is most suitable for verification of authentication CPs.

In this section we assume that the symbol $P$ denotes a DP that does not contain $P_i$.

3.1 Verification method based on transition graphs

3.1.1 Method description

Some properties of DPs can be expressed by formulas, related to reachable nodes of corresponding TGs. For example, one of properties of DP $P_i$, where $i = 1, \ldots, 4$, defined in section 2,6,3 has the following form:

$$\forall s', s'' \in \Sigma P_i : s' \xrightarrow{\alpha_i} s'' \quad (s' \models \psi_i \Rightarrow s'' \models \psi_i)$$

(50)

which follows from theorem 2 (i = 1, 3), or theorem 3 (i = 2, 4).

We will use statements (10) in solutions for the verification problem for some properties of DPs $P_i$ ($i = 1, \ldots, 4$) presented below.

- the truth of $\psi_i$ in the initial state of DP $(P_i)_i$ follows from the definitions of the concept of an initial state and the SP $P_i$, and
- the truth of $\psi_i$ in a state $s$, to which there is a path from 0 with edge labels of the form $\alpha_{P_i}$, is substantiated by the statement

$$\forall s', s'' \in \Sigma P_i : s' \xrightarrow{\alpha_i} s'' \quad (s' \models \psi_i \Rightarrow s'' \models \psi_i)$$

(50)

which follows from theorem 2 (i = 1, 3), or theorem 3 (i = 2, 4).

We will use statements (10) in solutions for the verification problem for some properties of DPs $P_i$ ($i = 1, \ldots, 4$) presented below.
• if an execution of the DP \( \langle P_i \rangle_1 = \{ A, B, P_1 \} \) or \( \{ A, J, B, P_1 \} \) has reached a state where the receiver \( B \) ends a part of its execution, related to a receiving of a message from the sender \( A \),

• then for any opposition of the adversary \( P_i \) the transmitted message \( x \) in \( A \) is equal to the value that the variable \( y \) in \( B \) will be assigned.

If a property of a DP \( P \) has the form \( at \models \varphi \), where \( at \) is a reachable node of \( G_P \), then a method for verification this property is the following:

• for each reachable node \( at' \) located on some path from \( Init(G_P) \) to \( at \), a formula \( \varphi_{at'} \) which is true in \( at' \), is calculated, and

• the property \( \varphi_{at} \leq \varphi \) is checked.

The need to calculate the above formulas for all nodes on paths from \( Init(G_P) \) to \( at \) is due to the fact that to calculate \( \varphi_{at} \) you need to know the formulas \( \varphi_{at'} \) for each reachable node \( at' \) from which there is a realizable edge to \( at \), and so on.

The method for calculating \( \varphi_{at} \) is as follows:

• if \( \varphi_{at'} \) is calculated for some reachable node \( at' \) such that there is an edge \( at' \xrightarrow{\alpha} at \), then the formula \( \alpha(\varphi_{at'}) \) is calculated, the meaning of which is as follows: if \( \varphi_{at'} \) is true in state \( s \), and \( s \xrightarrow{\alpha} s' \), then \( \alpha(\varphi_{at'}) \) is true in each state to which there is a path from \( s' \) with edge labels of the form \( \alpha P_1 \),

• \( \varphi_{at} \) is defined as an analogue of a disjunction of formulas from the set \( \{ \alpha(\varphi_{at'}) \mid at' \xrightarrow{\alpha} at \} \).

For an initial node \( at^0 = Init(G_P) \) the corresponding formula \( \varphi_{at^0} \) is assumed to be given. For example, for DPs \( \langle P_i \rangle_1 \) (i = 1, \ldots, 4) defined in [26,3] such formulas can be taken from the corresponding statements in [49].

Below we present examples of using this method for a verification of DPs, in which secure channels (\( P_1 \) and \( P_3 \)) or secure keys (\( P_2 \) and \( P_4 \)) are used. Before verification of these DPs, we state theorems used for verification of these DPs.

### 3.1.2 Theorems for verification of processes with secure channels

In this section, we will use the following notation:

\[
\forall E \subseteq Tm, \forall \alpha \in Act, \forall c \in Tm_C \quad E_{c,\alpha} = E \cup \{ c \}, \text{ if } \alpha = cle, \text{ and } E_{c,\alpha} = E, \text{ otherwise.} \quad (52)
\]

**Theorem 7.**

Let \( P \) be a DP, \( E \) be a subset \( \langle P \rangle_0 \), \( s, s' \) be states from \( \Sigma_P \), such that \( s \xrightarrow{\alpha P} s' \), where \( P \in \mathcal{P} \), if \( \alpha = cle \), then the implication holds:

\[
c' \notin E \Rightarrow Var(e') \cap E = \emptyset. \quad (53)
\]

Then \( \forall E', E'' \subseteq Tm, \forall c \in E_C \) the following implication holds:

\[
s \models \left\{ \begin{array}{l}
E \perp_{C} P_1 \\
E' \subseteq [c] \subseteq E''
\end{array} \right. \Rightarrow \quad \Rightarrow s' \models \left\{ \begin{array}{l}
E' \perp_{C} P_1 \\
E'_{c,\alpha} \subseteq [c] \subseteq E''_{c,\alpha}
\end{array} \right.
\]

**Proof.**

According to (3), the value of the formula \( E \perp_{C} P_1 \) in \( s \) depends only on the sets \( [P_1]^s \) and \([c]^s \ (\forall c \in Tm_C)\), and when passing from \( s \) to \( s' \)

• if \( \alpha = c?e \) or \( \alpha = \{ e := e' \} \), then these sets do not change,

• if \( \alpha = cle \), then only \([c]^s \) can change by adding the term \( e^s \) to it,

therefore (53) implies that

\[
s \models E \perp_{C} P_1 \Rightarrow s' \models E \perp_{C} P_1.
\]

Implication

\[
s \models E' \subseteq [c] \subseteq E'' \Rightarrow s' \models E'_{c,\alpha} \subseteq [c] \subseteq E''_{c,\alpha}
\]

follows from definition (52). \( \blacksquare \)

**Theorem 8.**

Let there are given

• DP \( P \), subset \( E \subseteq \langle P \rangle_0 \), node \( at \in G_P \),

• set \( \{ at_i \xrightarrow{\alpha} at \mid i \in I \} \) of edges of TG \( G_P \) (with a common end \( at \)), and if \( G_P \) contains an edge of the form \( at' \xrightarrow{\alpha} at \), which does not belong this set, then \( at' \) is unreachable,
• set \{ \varphi_i \mid i \in I \} of formulas corresponding to the above edges, where \( \forall i \in I \; at_i \models \varphi_i \), and \( \varphi_i \) consists of the following EFs:

- \( E \perp \mathcal{C} P_1 \),
- \( E'_{i,c} \subseteq \left[ e \right] \subseteq E''_{i,c}, \) where \( e \in E \mathcal{C} \), and \( E'_{i,c}, E''_{i,c} \subseteq Tm_i \),
- equalities \( e = e' \), where \( e, e' \in Tm \).

Then \( at \models \varphi \), where \( \varphi \) consists of the EFs

- \( E \perp \mathcal{C} P_1 \),
- \( \bigcap_{i \in I} (E'_{i,c})_{c,a} \subseteq \left[ e \right] \subseteq \bigcup_{i \in I} (E''_{i,c})_{c,a} \), and
- equalities \( e = e' \), occurred in each \( \varphi_i \) \( (i \in I) \).

Proof. This theorem is a consequence of theorems \( 7 \) and \( 2 \) \( \blacksquare \)

3.1.3 Reduction of transition graphs

If analyzed property of a TG \( G_P \) is related with its reachable nodes, then unreachable nodes and edges associated with these nodes can be removed from this TG. Such operation of a removing is called a reduction of a TG. The resulting graph is called a reduced TG, and is denoted by the same notation \( G_P \). If the reduced TG has unreachable nodes, then this TG can be reduced again, and so on.

Unrealizable edges and unreachable nodes of TGs can be detected with use of the following theorems (we omit proofs of these theorems).

Theorem 10.

Let \( P \) be a DP, and \( at \in G_P \). If

- \( at \models \{ [c] = \emptyset, c = c' \} \), where \( c, c' \in Tm_{\mathcal{C}} \), or
- \( at \models \{ k^{-1}[\alpha] = \emptyset, k = k' \} \), where \( k, k' \in Tm_{\mathcal{K}} \),

and there is an edge at \( \overset{c}{\rightarrow} at' \), where \( \alpha \) has the form \( c?'e \) or \( ?k'(e) \) respectively, then this edge is unrealizable. \( \blacksquare \)

Theorem 11.

Let \( P \) be a DP, and \( at \in G_P \), then

- if all edges ending in \( at \) are unrealizable, then \( at \) is unreachable, and
- if \( at \) is unreachable, then all edges starting with \( at \) are unrealizable. \( \blacksquare \)

3.1.4 Verification of \( P_1 \)

Let us apply the theorems stated above to verification property \( (51) \) for DP \( P \), described by diagram \( (10) \). TG \( G_{P_1} \) has the form \( (15) \).

Theorem 10 and first statement in \( (10) \), which has the form

\[
A^0 B^0 \models \{ \varphi_1, [c_{AB}] = \emptyset \} \tag{54}
\]

justify the unrealizability of the edge marked with a black circle in TG \( (15) \). By theorem \( (11) \) this implies unreachability of node \( A^0 B^1 \).

After reduction of TG \( (15) \) by removing unreachable nodes and associated edges we get the graph

\[
\begin{align*}
A^0 B^0 \xrightarrow{c_{AB}} & A^1 B^0 \xrightarrow{c_{AB}} A^1 B^1 \xrightarrow{?c} \cdots
\end{align*}
\tag{55}
\]

In \( (55) \) there is only one node \((A^1 B^1)\) satisfying the condition in \( (51) \). Thus, it is required to prove that

\[
A^1 B^1 \models x = y. \tag{56}
\]

From \( (54) \) by theorem \( (8) \) we get

\[
A^1 B^0 \models \{ \varphi_1, [c_{AB}] = \{ x \} \},
\]

whence, by theorems \( (8) \) and \( (9) \) we get \( (56) \).

3.1.5 Verification of \( P_3 \)

Now consider the problem of proving \( (51) \) for DP \( P_3 \), described by \( (14) \), in which actions are defined according to \( (12) \). TG \( G_{P_3} \) has the form \( (17) \).

Theorem 10 and third statement in \( (14) \), which has the form

\[
A^0 J^0 B^0 \models \{ \varphi_3, [c_{AJ}] = \emptyset, [c_{BJ}] = \emptyset, [\in] = \emptyset \} \tag{57}
\]
justify the unrealizability of the edges marked with black circles in (47). By theorem 11, this implies unrealachability of all nodes of the upper tier in TG (47) except for the node \( A^0 J^0 B^0 \).

After reduction of TG (47) by removing the unreachable top tier nodes and associated edges we get the reduced TG (55).

Below we give a list of statements, each of which follows from the previous ones (the first follows from (57)) according to theorems 8 and 9:

\[
A^1 J^0 B^0 \models \{ \varphi_3, [c_{AJ}] = \{e\}, [c_{BJ}] = \emptyset, \quad [c] = \emptyset \}
\]

\[
A^2 J^0 B^0 \models \{ \varphi_3, [c_{AJ}] = \{e\}, [c_{BJ}] = \emptyset, \quad [c] = \{x\} \}
\]

\[
A^1 J^1 B^0 \models \{ \varphi_3, [c_{AJ}] = \{e\}, [c_{BJ}] = \emptyset, \quad [c] = \emptyset, u = e \}
\]

\[
A^2 J^1 B^0 \models \{ \varphi_3, [c_{AJ}] = \{e\}, [c_{BJ}] = \emptyset, \quad [c] = \{x\}, u = c \}
\]

By theorem 10 these statements imply unrealizability of the edges marked with black circles in TG (55). After removing these edges and the corresponding unreachable nodes (using theorem 11), we get the reduced TG (60).

From the last statement in (59) using theorems 8 and 9 we get:

\[
A^1 J^2 B^0 \models \{ \varphi_3, [c_{AJ}] = \{e\}, [c_{BJ}] = \{u\}, \quad [c] = \emptyset, u = c \}
\]

\[
A^2 J^2 B^0 \models \{ \varphi_3, [c_{AJ}] = \{e\}, [c_{BJ}] = \{u\}, \quad [c] = \{x\}, u = c \}
\]

By theorem 10 from the last statement in (61) it follows that the edge in TG (60) marked with a black circle is unrealizable. Removing this edge and the corresponding unreachable nodes (to find which we use theorem 11), we get the reduced TG (62).

Applying theorems 8 and 9 we calculate the formulas corresponding to the remaining nodes:

\[
A^2 J^2 B^0 \models \{ \varphi_3, [c_{AJ}] = \{e\}, [c_{BJ}] = \{u\}, \quad [c] = \{x\}, u = c \}
\]

\[
A^2 J^2 B^1 \models \{ \varphi_3, [c_{AJ}] = \{e\}, [c_{BJ}] = \{u\}, \quad [c] = \{x\}, u = c, v = u \}
\]

Since

- in TG (62) node \( A^2 J^2 B^2 \) is the only node that satisfies the condition in (61), and

- according to the last statement in (65), this node satisfies the statement in (61), then the problem of proving property (61) for DP \( P_3 \) is solved.

### 3.1.6 Theorems for verification of distributed processes with secure keys

In this section, we will use the following notation:

\[
\forall E \subseteq Tm, \forall \alpha \in Act, \forall k \in Tm_k \quad E_{k,\alpha} = E \cup \{e\}, \text{ if } \alpha = k(e), \text{ and } \quad E_{k,\alpha} = E, \text{ otherwise.}
\]

**Theorem 12.** Let \( P \) be a DP, \( \text{Var}(P)_C = \{c\}, P \in P, E \) be a subset of \( (P)_0 \), \( s, s' \) be states from \( \Sigma_{P_1} \), such that \( s \xrightarrow{E,\alpha} s' \), and if \( \alpha = l_e \), then

\[
\forall x \in E_X \ x \perp_{K_E} e^s.
\]

Then \( \forall E', E'' \subseteq Tm, \forall k \in E_K \) the following implication holds:

\[
s \models \left\{ \begin{array}{l}
E \perp_{K} P_1 \cup k^{-1}[P_1] \subseteq k^{-1}[o]
\end{array} \right\} \Rightarrow
\]

\[
\Rightarrow s' \models \left\{ \begin{array}{l}
E' \subseteq k^{-1}[o] \subseteq E''
\end{array} \right\}
\]

**Proof.**

Values of formulas \( E \perp_{K} P_1 \cup k^{-1}[P_1] \subseteq k^{-1}[o] \) in \( s \) depend only on sets \( [P_1]^s \) and \( [o]^s \) (for \( E \perp_{K} P_1 \) this follows from (4)), and

- if \( \alpha = ? e \) or \( \alpha = (e := e') \), then \( [P_1]^s \) and \( [o]^s \) do not change when passing from \( s \) to \( s' \), and

- if \( \alpha = l_e \), then only \( [o]^s \) changes by adding \( e^s \) when passing from \( s \) to \( s' \),

then theorems (65) implies the implication

\[
s \models \{ E \perp_{K} P_1 \cup k^{-1}[P_1] \subseteq k^{-1}[o] \} \Rightarrow
\]

\[
\Rightarrow s' \models \{ E' \subseteq k^{-1}[o] \subseteq E'' \}
\]

**Implication**

\[
s \models E' \subseteq k^{-1}[o] \subseteq E'' \Rightarrow
\]

\[
\Rightarrow s' \models E'_{k,\alpha} \subseteq k^{-1}[o] \subseteq E''_{k,\alpha}
\]

follows from (64).
• $E \perp C P_1$ on $\{E \perp K P_1, k^{-1}[P_1] \subseteq k^{-1}[o]\}$,

• $[c]$ on $k^{-1}[o]$,

• $E_{i,c}$ on $E_{i,k}, (E_{i,c})_{c,\alpha}$ on $(E_{i,k})_{k,\alpha}$,

and an analog of theorem 10 holds, with the replacement $c ? x$ on $k(x), [c']$ on $(k')^{-1}[o], c = c'$ on $k = k'$.

3.1.7 Verification of $P_2$

A proof of property (51) for DP $P_2$ described by diagram (11) is carried out similarly to the proof of this property for DP $P_1$ in section 3.1.3. TG $G_{P_2}$ has the form (10). An unrealizability of the edge marked with a black circle in $G_{P_2}$ is substantiated with use of theorem 10 and second statement in (49). By theorem 11 we get an unreachability of the node $A^0 B^1$. After reduction of TG $G_{P_2}$ we get the TG

$$A^0 B^0 \xrightarrow{k_{AB}(x)} A^1 B^0 \xrightarrow{\hat{k}_{AB}(y)} A^1 B^1 \cdots$$ (66)

In (66) there is a single node $(A^1 B^1)$ that satisfies the condition in (51). Thus, it is required to prove that $A^1 B^1 \models x = y$. This property follows from the statement $A^1 B^0 \models \{\varphi_2, k_{AB}^1 [o] = \{x\}\}$.

3.1.8 Verification of $P_3$

Proof of property (51) for DP $P_3$ described by diagram (13), where actions $\alpha_i, \beta_i, j_i$ $(i = 1, 2)$ are defined according to (13), is carried out similarly to the proof of this property for DP $P_3$ in section 3.1.3. TG $G_{P_3}$ has the form (17). After reduction of TG (17) we get the same TGs (58), (60), (62), as in the case of verification of DP $P_3$. We will not describe in detail a solution of the verification problem for DP $P_4$, we will only present statements associated with nodes of TG (62) for this case.

$A^1 J^0 B^0 \models \{\varphi_4, k_{A\hat{J}}^{-1}[o] = \{\hat{k}\}, k_{B\hat{J}}^{-1}[o] = \emptyset, k^{-1}[o] = \emptyset\}$

$A^2 J^0 B^0 \models \{\varphi_4, k_{A\hat{J}}^{-1}[o] = \{\hat{k}\}, k_{B\hat{J}}^{-1}[o] = \emptyset, k^{-1}[o] = \{x\}\}$

$A^1 J^1 B^0 \models \{\varphi_4, k_{A\hat{J}}^{-1}[o] = \{\hat{k}\}, k_{B\hat{J}}^{-1}[o] = \emptyset, k^{-1}[o] = \emptyset, u = k\}$

$A^2 J^1 B^0 \models \{\varphi_4, k_{A\hat{J}}^{-1}[o] = \{\hat{k}\}, k_{B\hat{J}}^{-1}[o] = \emptyset, k^{-1}[o] = \emptyset, u = k \vee v = u\}$

$A^1 J^2 B^1 \models \{\varphi_4, k_{A\hat{J}}^{-1}[o] = \{\hat{k}\}, k_{B\hat{J}}^{-1}[o] = \emptyset, k^{-1}[o] = \emptyset, u = k \vee v = u\}$

$A^2 J^2 B^2 \models \{x = y\}$.

3.2 Yahalom protocol verification

In this and next section, we consider a method for verifying CPs, based on theorem 10. This method is not explicitly described, since it can be understood by examples of verification of Yahalom CP (in this section) and a CP of message transmission with unlimited number of participants (in section 3.3).

3.2.1 Description of Yahalom protocol

Yahalom protocol is designed to authenticate agents communicating over the open channel $o$. It is assumed that

• there are given a set $A_g \subseteq Var_A$, and a trusted intermediary $J \in Var_A$, these agents can communicate through channel $o$,

• each $A \in Ag$ has a shared secret key $k_{AJ}$ with $J$, on which $A$ and $J$ can encrypt and decrypt messages using a symmetric encryption system, and only $A$ and $J$ know the key $k_{AJ}$.

The following agents participate in each Yahalom session: an initiator $A \in Ag$, a trusted intermediary $J$, and a responder $B \in Ag$. Each agent from $Ag$ can be an initiator in some sessions, and a
The same agent may be both an initiator and a responder in the same session (i.e., it is possible that $A = B$). A Yahalom session with an initiator $A$, a responder $B$ and a trusted intermediary $J$ is a set of four message transfers:

1. $A \rightarrow B : A, n_A$
2. $B \rightarrow J : B, k_{BJ}(A, n_A, n_B)$
3. $J \rightarrow A : k_{AJ}(B, k, n_A, n_B), k_{BJ}(A, k)$
4. $A \rightarrow B : k_{BJ}(A, k), k(n_B)$

Transfers in (67) have the following meaning:

1. $A$ sends $B$ a request for an authentication and a generation of a session key $k$, this request consists of the agent name $A$ and nonce $n_A$.
2. $B$ sends $J$ a request to generate a session key $k$, in its request it includes its name, the name of the agent $A$, for communication with which this key is needed, the received nonce $n_A$, and its nonce $n_B$.
3. $J$ generates session key $k$ and sends $A$ a pair of messages,
   - from first message $A$ can extract $k$,
   - and second message is intended for $A$ to forward it to $B$,
4. $A$ sends $B$ a pair of messages,
   - first of which it received from $J$, $B$ can extract a session key $k$ from this message,
   - using $k$, $B$ decrypts second message, if a result of the decryption matches its nonce $n_B$, then this is a proof for him that a sender of this message is exactly $A$.

A Yahalom session is described by diagram (68). In this diagram,

- left and right threads correspond to SPs $I_A$ and $R_B$, describing a behavior of the initiator $A$ and the responder $B$, respectively,
- middle thread corresponds to a SP, describing a behavior of the intermediary $J$, this SP is denoted by the same symbol $J$.

The meaning of variables in these SPs is seen from the comparison of actions in these SPs with the corresponding actions in (67). Superscripts $i$ and $r$ on variables mean that these variables presumably contain an information about an initiator ($i$) or a responder ($r$) of this session.

We assume that

$\text{Agent}(I_A) = A, \text{Agent}(R_B) = B, \text{Agent}(J) = J$.

A DP $\mathcal{P}$ corresponding to Yahalom has the form

$\mathcal{P} = \{\{I_A | A \in Ag\}, \{R_B | B \in Ag\}, J^*\}$.  

We will use the following notations:

- if $\mathcal{P}$ is a DP, and $\pi$ is a path in $\Sigma_{P_i}$, then $\pi \ni P_{i,i'} : s \ni s'$ means that $\pi$ contains the edge $s \rightarrow s'$, and $a_t_{\pi} = i$, $a_{t_{\pi'}} = i'$,

$\text{•} \ s \models E \perp_{E} e$ denotes the statement

$$\forall x \in E \ X \ s \models E \perp_{E} e$$

It is not hard to prove that

$$s \models E \perp_{E} (e, e') \Leftrightarrow \left\{ \begin{array}{ll}
\text{•} & s \models E \perp_{E} e \\
\text{•} & s \models E \perp_{E} e'.
\end{array} \right.$$  

(70)

### 3.2.2 Properties of Yahalom protocol

The following properties of DP (69) will be verified:

**secrecy** of keys and nonces $n'_B$:

$$\forall s \in \Sigma_{P_i} \ s \models E \perp_{E} P_i,$n'_B \text{ where } E = \{k_{BJ}, k_J, n'_B | B \in Ag\} \quad (71)$$

**authentication of the initiator to the responder**: $\forall R_B \in \mathcal{P}, \forall s \in \Sigma_{P_i}$, if $s \models a_{t_{R_B}} = 3$, then $\exists I_A \in \mathcal{P}$:

$$s \models \{a_{t_{I_A}} = 3, n'_A = B, a'_B = A, n'_A = n'_B, n'_A = n'_B, k'_A = k'_B\} \quad (72)$$

**authentication of the responder to the initiator**: $\forall I_A \in \mathcal{P}, \forall s \in \Sigma_{P_i}$, if $s \models a_{t_{I_A}} = 2$, then $\exists R_B \in \mathcal{P}$:

$$s \models \{a_{t_{R_B}} = 2, a'_A = B, a'_B = A, n'_A = n'_B, n'_A = n'_B\}.$$  

(73)
3.2.3 Secrecy of keys and nonces $n_B^r$

Prove (71) by contradiction.

Suppose $S = \{ s \in \Sigma_P | s \not\models \varphi \} \neq \emptyset$, where $\varphi$ is a formula in (71).

$\forall s \in S$ denote by $\pi_s$ a path of minimum length from 0 to $s$. Let $s$ be a state from $S$ with the least length of $\pi_s$. Since $0 \models \varphi$, then $s \neq 0$.

Let $s^{e_1}$ be an edge from $\pi_s$ ending in $s$.

From the definition of $s$ it follows that $s' \models \varphi$, $s \not\models \varphi$. If $P = P_1$, then from theorem 3 it follows that $s \models \varphi$, i.e. we have a contradiction.

Therefore, $P = \{ I_A, R_B, J \mid A, B \in Ag \}$, and

$$\alpha_P = \{ e \in [s_0] \cup \{ e \} \}.$$  

There is only justification of the existence of the edge $s^{e_1} \rightarrow s$ with properties (74):

$$\pi_s \models I_A^{2,3} : s^{e_1} \rightarrow s, \text{ where } e = (x, k_A^r(n_A^r)). \quad (75)$$

Since $s' \models at_{I_A} = 2$, then $\exists s_1 \leq s_s$, $s'$:

$$\pi_s \models I_A^{2,3} : s_1^{e_1} \rightarrow s_1, \text{ where } e_1 = (k_A^r(x, k_A^r(n_A^r), \hat{x}).$$  

Since $s_1 \leq s_s$, $s'$ and $s' \models \varphi$, then $s_1 \models \varphi$. In particular, $s_1 \models E \perp_k e_1$. By (70), this implies $s_1 \models E \perp_k x$.

By theorem 6 $s_1 \models \varphi$, $e_1^r \in [o]_{s_1}$, and $k_AJ \in E$ imply: $\exists s_2 \leq s_s$, $s'_1$:

- $\pi_s$ contains the edge $s_2^{e_2} \rightarrow s_2$, where $P \in P$,
- first component $k_AJ(\ldots)$ of the term $e_1^r$ is a subterm of $e_2^r$.

There is only justification the existence of an edge with such properties:

$$\pi_s \models J_1^{1,2} : s_2^{e_2} \rightarrow s_2, \text{ where }$$

$$e_2 = (k_BJ(a_B^r, k_B^r(n_B^r)), \ldots), \quad (77)$$

(ellipse in (77) and below denotes a component of a pair that is not of interest for consideration).

From the above property $s_1 \models E \perp_k e$ and (70) we get: $s \models E \perp_k x$ and (71) we get: $s \models E \perp_k x$, i.e. $s \models E \perp_k e$, which contradicts the assumption $s \not\models E \perp_k e$. □

Proven property $\forall s \in \Sigma_P, s \models \varphi$ will be used below. In proofs presented below, for each application of theorem 6 there is only one way to justify the existence of the edge (74) in the graph $\Sigma_P$, and we will not mention the uniqueness of such a justification. This uniqueness is ensured by a suitable definition of actions of the form $!e$ in SPs occurred in DPs under consideration.

3.2.4 Authentication of the initiator to the responder

Let a SP $R_B \in P$ and a state $s \in \Sigma_P$ are such that $s \models at_{R_B} = 3$. Prove that $\exists I_A \in P: (72)$ holds.

Let $\pi$ be a path from 0 to $s$. The statement $s \models at_{R_B} = 3$ implies that $\exists s_1 \leq \pi s$:

$$\pi \models R_B^{2,3} : s_1^{e_1} \rightarrow s_1, \text{ where } \quad (78)$$

By theorem 6 from $s_1 \models \varphi$, $e_1^r \in [o]_{s_1}$, $k_BJ \in E$ it follows that $\exists s_2 \leq \pi s'_1$:

$$\pi \models J_1^{1,2} : s_2^{e_2} \rightarrow s_2, \text{ where }$$

$$e_2 = (\ldots, k_BJ(a_B^r, k_B^r)).$$
Second equality in (78) implies that
\[(a_j')^s = B, (a_j')^s = (a_B')^s, \bar{k}_J = (k_B')^s. \tag{79}\]
From \(s_2' \models at_J = 1\) it follows that \(\exists s_3 \leq s_2'\):
\[
\pi \ni J_0^{1.1} : s_3 \ni s, \text{ where } e_3 = (\ldots, k_{a_j'J}(\bar{a}_j', \bar{n}_j, \bar{n}_j)). \tag{80}\]
From (79) and (80) it follows that
\[k_{BJ} (3 \text{ terms}) \subseteq e_3' \ni [0]_{s_3},\]
whence by theorem 5 with considering \(s_3 \models \varphi\) and \(k_{BJ} \in E\) we get: \(\exists s_4 \leq s_3'\):
\[
\left\{ \begin{array}{l}
\pi \ni \varphi_{1.2} : s_4' \ni s_4, \text{ where } \varepsilon_4 = (\ldots, k_{BJ}(a_B', n_B', \bar{n}_B)) \\
(k_{BJ}(a_B')^s, (n_B')^s, \bar{n}_B) = \ni k_{BJ}((a_B')^s, (n_B')^s, \bar{n}_B)^s \end{array} \right. \tag{81}\]
Second equality in (81) it follows that
\[B = B, (n')^s = (\bar{n}')^s, \bar{n}_B = (\bar{n}_B)'^s. \tag{82}\]
By theorem 6 from
\[s_1 \models \varphi, (k_B'(n_B'))^s \subseteq e_1' \ni [0]_{s_1}, (k_B')^s = \bar{k}_J \in E\]
it follows that \(\exists s_5 \leq s_1'\):
\[
\left\{ \begin{array}{l}
\pi \ni \varphi_{2.3} : s_5' \ni s_5, \text{ where } \varepsilon_5 = (\ldots, k_A'(n_A')) \varepsilon_5 = k_J(n_B) \end{array} \right. \tag{83}\]
From second equality in (83) it follows that
\[k_A'(a_A')^s = k_J, (n_A')^s = n_B. \tag{84}\]
From \(s_5 \models at_{I_A} = 2\) it follows that \(\exists s_6 \leq s_5'\):
\[
\pi \ni I_{A1}^{1.2} : s_6 \ni s_6, \text{ where } \varepsilon_6 = (k_{A,J}(a_A', k_J, n_A', \bar{n}_A)). \tag{85}\]
From (84) and (85) it follows that
\[k_{A,J}(a_A'(k_A')^s, (n_A')^s, \bar{n}_A)^s) = \ni k_{A,J}(a_A', k_J, n_A', \bar{n}_A)^s \subseteq e_6' \ni [0]_{s_6}.\]
By theorem 6 from \(s_6 \models \varphi, k_{A,J} \in E, \text{ and } (86)\) it follows that \(\exists s_7 \leq s_6'\):
\[
\left\{ \begin{array}{l}
\pi \ni J_1^{1.2} : s_7' \ni s_7, \text{ where } \varepsilon_7 = (k_{a_{i,j}J}(a_{i,j}', k_{J}, n_{i,j}', \bar{n}_{i,j}')) \\
k_{a_{i,j}J}(a_{i,j}', (n_{i,j}')^s, k_{J}, (n_{i,j}')^s, \bar{n}_{i,j}' \ni k_{A,J}(a_A', k_J, n_A', \bar{n}_A) \end{array} \right. \tag{87}\]
From second equality in (87) it follows that
\[(a_j')^s = A, (a_j')^s = a_A', \bar{k}_J = (k_A')^s. \tag{88}\]
\[\bullet \quad \text{(72) follows from (74), (72), (81), (84).}\]

### 3.2.5 Authentication of the Responder to the Initiator

Let a SP \(I_A \in \mathcal{P}\) and a state \(s \in \Sigma_{\mathcal{P}}\) are such that \(s \models at_{I_A} = 2\). Prove that \(\exists R_{B} \in \mathcal{P}\) (83) holds.

Let \(\pi\) be a path from 0 to \(s\). From \(s \models at_{I_A} = 2\) it follows that \(\exists s_1 \leq s\):
\[
\pi \ni I_{A1}^{1.2} : s_1 \ni s_1, \text{ where } \varepsilon_1 = (k_{A,J}(a_A', k_A', n_A', \bar{n}_A')). \tag{89}\]
By theorem 6 from
\[s_1 \models \varphi, k_{A,J} (4 \text{ terms}) \subseteq e_1' \ni [0]_{s_1}, k_{A,J} \in E\]
it follows that \(\exists s_2 \leq s_1'\):
\[
\left\{ \begin{array}{l}
\pi \ni \varphi_1^{2.3} : s_2' \ni s_2, \text{ where } \varepsilon_2 = (k_{a_{i,j}J}(a_{i,j}', k_J, n_{i,j}', \bar{n}_{i,j}')). \\
k_{a_{i,j}J}(a_{i,j}', k_J, (n_{i,j}')^s, (n_{i,j}')^s, \bar{n}_{i,j}') \ni k_{A,J}(a_A', k_A', n_A', (n_A')^s) \end{array} \right. \tag{90}\]
From second equality in (90) it follows that
\[(a_j')^s = A, (a_j')^s = a_A', \bar{k}_J = (k_A')^s, (n_j')^s = n_A', (n_j')^s = n_A'. \tag{91}\]
From \(s_2 \models at_J = 1\) it follows that \(\exists s_3 \leq s_2'\):
\[
\pi \ni J_0^{1.1} : s_3 \ni s_3, \text{ where } e_3 = (\ldots, k_{a_{i,j}J}(\bar{a}_{i,j}', \bar{n}_{i,j}, \bar{n}_{i,j}')). \tag{92}\]
From (91) and (92) it follows that
\[k_{a_{i,j}J}(A, n_A', (n_A')^s) \subseteq e_3' \ni [0]_{s_3},\]
whence by theorem 6 considering \(s_3 \models \varphi\), and \(k_{a_{i,j}J} \in E\), we get: \(\exists s_4 \leq s_3'\):
\[
\left\{ \begin{array}{l}
\pi \ni R_{B1}^{1.2} : s_4' \ni s_4, \text{ where } \varepsilon_4 = (\ldots, k_{B,J}(a_B', n_B', \bar{n}_B)) \\
k_{B,J}((a_B')^s, (n_B')^s, \bar{n}_B) \ni k_{a_{i,j}J}(A, n_A', (n_A')^s) \end{array} \right. \tag{93}\]
Second equality in (93) implies equalities from which \(\bullet\) follows:
\[B = a_A'(a_B')^s = A, (n_B')^s = n_A', \bar{n}_B = (\bar{n}_A')^{s}. \]
3.3 Verification of the protocol of message transmission with unlimited number of participants

In this section we consider an example of verification of a CP intended for EM transmission with unlimited number of participants. This CP is a generalization of the Wide-Mouth Frog CP $P_4$, considered in section 2.6.3.

3.3.1 Protocol Description

Participants of this CP are agents from the set $Ag \subseteq Var_A$ and a trusted intermediary $J \in Var_A$. Each $A \in Ag$ uses the key $k_{AJ}$ to communicate with $J$, which is available only to $A$ and $J$. The encrypted transmission of the message $x$ from $A \in Ag$ to $B \in Ag$ consists of the following actions:

- exchange of messages between $A$ and $J$, as a result of which $J$ learns the name $A$ of the sender, the name of $B$ of the recipient, and the key $k$, on which $x$ will be encrypted,
- exchange of messages between $B$ and $J$, as a result of which $B$ learns the name $A$ of the sender of the message that $B$ will receive from $A$, and the key $k$ on which this message will be encrypted,
- transfer of $EM$ $k(x)$ from $A$ to $B$.

An execution of a session of this CP with the initiator $A$, the responder $B$ and the trusted intermediary $J$ is the following set of message transfers:

\[
\begin{align*}
1. & \quad A \rightarrow J : \quad k_{AJ}(A, n_A) \\
2. & \quad J \rightarrow A : \quad k_{AJ}(n_A, n_J) \\
3. & \quad A \rightarrow J : \quad k_{AJ}(n_J, k) \\
4. & \quad J \rightarrow B : \quad k_{BJ}(n_A) \\
5. & \quad B \rightarrow J : \quad k_{BJ}(n_A, n_B, B) \\
6. & \quad J \rightarrow B : \quad k_{BJ}(A, n_B, k) \\
7. & \quad A \rightarrow B : \quad k(x)
\end{align*}
\]

This session is represented by the diagram (95).

A DP $P$ corresponding to this CP has the form (99). Properties of this CP that must be verified:

- secrecy of keys, transmitted messages and nonces:
  \[\forall s \in \Sigma_P, s \models E \perp_k P_1, \text{ where } E = \{k_{AJ}, k^j_A, x^j_A, n^j_A \mid A \in Ag\}\]

- integrity of transmitted messages:
  \[
  \begin{align*}
  \forall R_B \in P, \forall s \in \Sigma_P, & \text{ if } s \models at_{R_B} = 4, \text{ then } \exists I_A \in P: \\
  s \models \{at_{I_A} = 4, a^*_A = B, n^*_A = A, \\
  n^*_A = n^*_B, k^*_A = k^*_B, x^*_A = x^*_B\}
  \end{align*}
  \]

3.3.2 Verification of the protocol

The proof of secrecy property (100) coincides with the beginning of the reasoning in section 3.2.3 with the only difference that there is no way to justify the existence of edge $s^{a \rightarrow \sigma}$ with properties (98). Prove the integrity property (97). We will use in this proof property (96) proven above.

Let a SP $R_B \in P$ and a state $s \in \Sigma_{P_3}$ are such that $s \models at_{R_B} = 4$. Prove that $\exists I_A \in P$: the statement in third and fourth lines (97) holds.

Let $\pi$ be a path from 0 to $s$. From $s \models at_{R_B} = 4$ it follows that

\[
\exists s_1 \leq \pi s : \pi \ni R^{3,4}_B : s_1 \xrightarrow{e_1} s_1,
\]

where $e_1 = k^1_B(x^1_B)$.\quad (97)

\[
\exists s_2 \leq \pi s'_1 : \pi \ni R^{2,3}_B : s'_2 \xrightarrow{e_2} s_2,
\]

where $e_2 = k_B((\bar{a}_B, n^B_B, k^B_1))$.\quad (98)

By theorem 6 from second statement in (98), $e^*_2 \in [0]\sigma_2$, and $k_{BJ} \in E$, it follows

\[
\begin{align*}
\exists s_3 \leq \pi s'_2 : & \pi \ni J^{4,6}_B : s'_3 \xrightarrow{e_3} s_3, \\
& \text{where } e_3 = k^2_B((a^*_J, n^*_j, k_j)), \\
k_{(a^*_j)^*} & = (a^*_j)^*, \\
n_{(j^*_j)^*} & = n_{(j^*_j)^*}, \\
(k_{(j^*_j)^*})^* & = k_{(j^*_j)^*}.
\end{align*}
\]

From second equality in (99) it follows that

\[
\begin{align*}
(a^*_j)^* & = B, \quad (a^*_j)^* = (a^*_B)^*, \\
n_{(j^*_j)^*} & = n_{(j^*_j)^*}, \quad (k_{(j^*_j)^*})^* = k_{(j^*_j)^*}.
\end{align*}
\]

(100)

From first statement in (99) and (100) we get

\[
\begin{align*}
\exists s_4 \leq \pi s'_3 : & \pi \ni J^{4,5}_B : s'_4 \xrightarrow{e_4} s_4, \\
& \text{where } e_4 = k_{BJ}(n^*_j, n^B_B, B).
\end{align*}
\]

(101)

By theorem 6 from (101), $e^*_4 \in [0]\sigma_4$, $k_{BJ} \in E$, it follows that

\[
\begin{align*}
\exists s_5 \leq \pi s'_4 : & \pi \ni B^{1,2}_1 : s'_5 \xrightarrow{e_5} s_5, \\
& \text{where } e_5 = k_{BJ}(n^*_B, \bar{n}^*_B, \bar{B}), \\
k_{BJ}(n^*_B)^* & = n^*_B, k_{BJ}(\bar{n}^*_B, \bar{B}) = (\bar{n}^*_B, \bar{B}).
\end{align*}
\]

(102)
From second equality in (107) we get:

\[
A = (n_j)^s, \quad (n_j)^{s'} = n_j, \quad \bar{k}_A = (k_j)^s.
\]

From first statement in (108) and (110) we get:

\[
e_{10}^* = k_{AJ}(n_j, n_j^*, k_j).
\]

By theorem 6 from (107), \( e_{10}^* \in [c]_{s_{AJ}}, k_{AJ} \in E \) it follows that

\[
\begin{align*}
\exists s_{01} \leq s_{01}': \pi \ni \lambda_{2,0}^{s_0} : s_{01}^{s_{01}'} \ni s_{01}, \\
\text{where } e_{01} = k_{AJ}(n_j^*, n_j^*), \\
k_{AJ}(n_j^n, n_j^n) = k_{AJ}(n_j^n, n_j^n).
\end{align*}
\]

From second equality in (108) we get:

\[
(a_j)^s = A, \quad (n_j)^s = n_j^*, \quad n_j^* = (n_j^*)^s.
\]
$s \models at_{1A} = 4$ follows from (112) and (113):
$s_{11} \models at_{1A} = 4$, $\hat{A} = A$, $s_{11} \leq s$,

- $s \models a^r_A = B$ follows from (119),
- $s \models a^B_B = A$ follows from (100) and (109),
- $s \models n^i_A = n^i_B$ follows from (103) and (109),
- $s \models k^i_A = k^i_B$ follows from (111),
- $s \models x^i_A = x^i_B$ follows from (113).

4 Conclusion

In this work, a new model of cryptographic protocols was built, and examples of its use for solving verification problems of properties of integrity, secrecy and correspondence are presented.

For further activities on the development of this model and verification methods based on it, the following research directions can be named:

- development of specification languages for description of CP properties, allowing to express, for example, properties of zero knowledge in authentication CPs, properties of non-traceability in CPs of electronic payments, properties of an anonymity and a correctness of vote counting in CPs of electronic voting, and development of methods for verification properties expressed in these languages,

- construction of methods for automated synthesis of CPs by describing properties that they must satisfy.

5 Compliance with Ethical Standards

This study was funded by the Ministry of Digital Development, Communications and Mass Media of the Russian Federation and Russian Venture Company (Agreement No.004/20 dated 20.03.2020, IGK 000000007119P190002).

The author declares that he has no conflict of interest.

This article does not contain any studies with human participants performed by the author.

References

[21CDS] Veronique Cortier, Stephanie Deleune, and Vaishnavi Sundararajan. A Decidable Class of Security Protocols for Both Reachability and Equivalence Properties. Journal of Automated Reasoning, 65:479–520, April 2021.

[21RCSSS] Roggenbach, M., Cerone, A., Schlingloff, H., Schneider, G., Shaikh, S.A., Formal verification of security protocols, in: Formal Methods for Software Engineering: Languages, Methods, Application Domains (Texts in Theoretical Computer Science. An EATCS Series) 1st ed., Springer International Publishing, 2021.

[17CW] Veronique Cortier and Cyrille Wiedling. A formal analysis of the Norwegian E-voting protocol. Journal of Computer Security, 25(15777):21–57, 2017.

[16ABF] 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

[16B] Bruno Blanchet, Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif, 2016.

[16YEMM] Fan Yang, Santiago Escobar, Catherine A. Meadows, Jose Meseguer. Strand Spaces with Choice via a Process Algebra Semantics. PPDP ’16: Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming, September 2016, pages 76–89.

[14CK] Veronique Cortier, Steve Kremer. Formal Models and Techniques for Analyzing Security Protocols: A Tutorial. Foundations and Trends in Programming Languages, 1(3):151–267, (2014)

[13LP] Yongjian Li, Jun Pang. An inductive approach to strand spaces. Formal Aspects of Computing, Vol. 25, No. 4, 2013.
[01AF] M. Abadi, C. Fournet, Mobile values, new names, and secure communication, in 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL ’01), ed. by C. Hankin, D. Schmidt, London, UK (ACM, New York, 2001), pp. 104–115

[01B] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In 14th IEEE Computer Security Foundations Workshop (CSFW), pp. 82-96, 2001.

[01DMP] N.A. Durgin, J.C. Mitchell, D. Pavlovic, A compositional logic for protocol correctness, in 14th IEEE Computer Security Foundations Workshop (CSFW ’01), Cape Breton, Canada (IEEE Computer Society, Los Alamitos, 2001), pp. 241–272

[00AR] M. Abadi, P. Rogaway, Reconciling two views of cryptography (the computational soundness of formal encryption), in IFIP International Conference on Theoretical Computer Science (IFIP TCS ’00), ed. by J. van Leeuwen, O. Watanabe, M. Hagiya, P.D. Mosses, T. Ito, Sendai, Japan (2000), pp. 3–22

[00B] G. Bella. Inductive Verification of Cryptographic Protocols. PhD thesis, Cambridge University, 2000.

[00GT2] J. D. Gutman and F. J. Thayer. Authentication tests and the normal, efficient penetrator. IEEE Computer Society Symposium on Research in Security and Privacy, 2000.

[00RSGLR] P.Y.A. Ryan, S.A. Schneider, M.H. Goldsmith, G. Lowe and A.W. Roscoe. The Modelling and Analysis of Security Protocols: the CSP Approach, Addison-Wesley, 2000.

[00RS] P. Y. A. Ryan and S. A. Schneider. Process algebra and non-interference. Journal of Computer Security, 2000.

[99AG] M. Abadi, A.D. Gordon, A calculus for cryptographic protocols: the Spi calculus. Inf. Comput. 148, 1–70 (1999)

[99M] R. Milner. Communicating and Mobile Systems: The π-calculus. Cambridge, UK: Cambridge University Press, 1999.

[99P] L. C. Paulson. Inductive Analysis of the Internet Protocol TLS. In ACM Trans. on Information and System Security, 2(3), pp. 332-351, 1999.

[99THG1] F. J. Thayer, J. C. Herzog, and J. D. Gutman. Strand spaces: Proving security protocols correct. Journal of Computer Security, 7(2/3):191-230, 1999.

[99THG2] F.J. Thayer, J.C. Herzog, J.D. Gutman, Mixed Strand Spaces, in 12th IEEE Computer Security Foundations Workshop (CSFW ’99), IEEE Computer Society, Los Alamitos, 1999, pp. 72–82

[98P] L.C. Paulson, The inductive approach to verifying cryptographic protocols. J. Comput. Secur. 6(1–2), 85–128 (1998)

[98THG1] F.J. Thayer, J.C. Herzog, J.D. Gutman, Honest ideals on Strand Spaces, in 11th IEEE Computer Security Foundations Workshop (CSFW ’98), Rockport, USA (IEEE Computer Society, Los Alamitos, 1998), pp. 66–77

[98THG2] F. J. Thayer, J. C. Herzog, and J. D. Gutman. Strand spaces: why is a security protocol correct? IEEE Computer Society Symposium on Security and Privacy, 1998.

[98S] S. A. Schneider. Verifying authentication protocols in CSP. IEEE Transactions on Software Engineering, 1998.

[97DS] B. Dutertre and S. A. Schneider. Embedding CSP in PVS. An application to authentication protocols. Theorem proving in Higher Order Logics, number 1275 in LNCS. Springer, 1997.

[97LR] G. Lowe and A. W. Roscoe. Using CSP to detect errors in the TMN protocol. IEEE Transactions in Software Engineering, 23(10), 1997.

[97P] L.C. Paulson, Proving properties of security protocols by induction, in 10th IEEE Computer Security Foundations Workshop (CSFW ’97), Rockport, Massachusetts (IEEE Computer Society, Los Alamitos, 1997), pp. 70–83
[96S] S. Schneider, Security properties and CSP, in 17th IEEE Symposium on Security & Privacy (S&P ’96), Oakland, USA (IEEE Computer Society, Los Alamitos, 1996), pp. 174–187.

[96SvO] P.F. Syverson, P.C. van Oorschot, A unified cryptographic protocol logic. CHACS Report 5540-227 NRL (1996)

[96SS] S. A. Schneider and A. Sidiropoulos. CSP and anonymity. European Symposium on Research in Computer Security, 1996.

[95AN] R. Anderson and R. Needham. Programming Satan's computer. In J. van Leeuwen (ed.) Computer Science Today, volume 1000 of LNCS. Springer, 1995.

[95L] Gavin Lowe. An attack on the Needham-Schroeder public key authentication protocol. Information Processing Letters, 56(3):131–136, November 1995.

[94KMM] R.A. Kemmerer, C. Meadows, J.K. Millen, Three systems for cryptographic protocol analysis. J. Cryptol. 7, 79–130 (1994)

[93vO] P.C. van Oorschot, Extending cryptographic logics of belief to key agreement protocols, in 1st ACM Conference on Computer and Communications Security (ACM CCS ’93), ed. by D.E. Denning, R. Pyle, R. Ganesan, R.S. Sandhu, V. Ashby, Fairfax, USA (ACM, New York, 1993), pp. 232–243

[93SM] 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.

[91AT] M. Abadi, M. Tuttle, A semantics for a logic of authentication, in 10th ACM Symposium on Principles of Distributed Computing (PODC ’91), Montreal, Canada (ACM, New York, 1991), pp. 201–216

[90BAN] Burrows M., Abadi M., Needham R., A Logic of Authentication. In ACM Transactions on Computer Systems, 8(1), (1990) 18-36.

[87NS] Needham R., Schroeder M., Authentication revisited, Operating Systems Review, Vol. 21, No. 1, (1987).

[85H] C. A. R. Hoare. Communicating Sequential Processes. Prentice-Hall, 1985.

[81DS] Denning D., Sacco G., Timestamps in Key Distribution Protocols, Communications of the ACM, Vol. 24, No. 8, (1981) 533-536.

[80M] R. Milner, A Calculus of Communicating Systems, Springer Verlag, 1980.

[78NS] Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. Communications of the ACM, 21(12), December 1978.