New method of verifying cryptographic protocols based on the process model *

A. M. Mironov

Innopolis University,
Leading reserch center

amironov66@gmail.com

Abstract

A **cryptographic protocol (CP)** is a distributed algorithm designed to provide a secure communication in an insecure environment. CPs are used, for example, in electronic payments, electronic voting procedures, database access systems, etc. Errors in the CPs can lead to great financial and social damage, therefore it is necessary to use mathematical methods to justify the correctness and safety of the CPs. In this paper, a new mathematical model of a CP is introduced, which allows one to describe both the CPs and their properties. It is shown how, on the basis of this model, it is possible to solve the problems of verification of CPs.

1 Introduction

1.1 A concept of a cryptographic protocol

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

*This research has been financially supported 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 0000000007119P190002)

To ensure security properties of a CP (such as, for example, the secrecy of transmitted data), cryptographic transformations (encryption, electronic signature, hash functions, etc.) can be used in the CP.

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 encrypted texts without knowing of the corresponding cryptographic keys.

1.2 Vulnerabilities in cryptographic protocols

Many CP vulnerabilities are related not with poor cryptographic qualities of the cryptographic primitives used in them, but with logical errors in protocols. For example, a logical error was found in the CP for logging into a Google portal that allows a user to identify himself only once and then get access to various applications (such as Gmail or Google Calendar), allowing a dishonest service provider to impersonate any of its users.

There are many other examples of CPs (see for example [1]-[5]), which have been used for a long time in security-critical systems, but then it was discovered that these CPs contain vulnerabilities of the following type:

- participants of these CPs can receive distorted messages (or even lose them) as a result of interception, deleting or distorting of transmitted messages by the adversary, which violates the integrity property,
- the adversary can discover a secret information contained in the intercepted messages as a result of erroneous or malicious actions of CP participants.

Vulnerabilities were also detected in one of the most well-known CPs **Kerberos** [6]. The absence of vulnerabilities in the patched version of Kerberos was justified in [4]. There are many other examples of CP vulnerabilities used to authentication for cell phone providers, ATM cash withdrawals, e-passports, electronic elections, etc.

All of the above examples justify the fact that an informal analysis of the required properties is not enough for CPs used in the security critical systems, it is necessary

- to construct a **mathematical model** of the analyzed CPs,
- describe properties of analyzed CPs in the form of a mathematical objects called **specifications** of these CPs, and

- to construct proofs of statements that the analyzed CPs meet (or do not meet) the specifications, the procedure for constructing such proofs is called **verification** of the analyzed CPs.

In this work, a new mathematical model of CPs is constructed. In terms of this model it is possible to express such properties of correctness of CPs as, for example, integrity of transmitted messages (i.e., justification of the following property of the analyzed CPs: messages sent by one participant of a CP to another participant of this CP, reach the recipient in an undistorted form).

1.3 Historical overview of methods for verifying cryptographic protocols

Historically, first formal approach for CP verifying was the BAN-logic of Burrows M., Abadi M., and Needham R., [7]. This approach has very large limitations, in particular, it does not allow considering the case of unlimited generation of sessions of the analyzed protocol.

A more popular approach to CP verification is the strand spaces formalism developed by Joshua D. Guttman, Jonathan C. Herzog, F. Javier Thayer Fabrega, [8]-[10]. Among the works devoted to the description of various formalisms designed for modeling and verification of CPs, it should also be noted articles [11]-[29].

One of the CP verification formalisms is the approach associated with the use of Horn clauses and Constraint Systems, developed in the works of Abadi, Blanchet, Cortier and other specialists [30]. Among other CP models, the most popular ones are logic models (see for example [7], [32], [34]). These models make it possible to reduce the problems of CP verification to the problems of constructing proofs of theorems that CPs under analysis meet their specifications. Algebraic and logical approaches to CP verification are also considered in [35] - [37].

2 Sequential and distributed processes

In this paper, we outline the concepts of sequential and distributed processes. These concepts are basic mathematical objects for building a CP process model. This model is a development of the **Calculus of Cryptographic Protocols** of Abadi-Gordon (**SPI-calculus**, [36]). It can serve as a theoretical basis for a new method for verifications of CPs, where CP verification means the construction of a mathematical proof that an analyzed

CP has the desired properties. Examples of such properties are **integrity** and **secrecy** properties. In the process model described in this text, CPs and their formal specifications are represented as distributed processes.

One of the most important advantages of the proposed CP process model is the low complexity of proofs of CP correctness. In particular, this model eliminates the need to build the sets of all reachable states of the analyzed CPs. This provides an important advantage when analyzing sets of states of the analyzed CP in the case when sets of these states are potentially unlimited. Another important advantage of the proposed CP model is the high degree of automation of solving the CP verification problem based on this model.

2.1 Auxiliary concepts

2.1.1 Types, constants, variables, function symbols

We assume that there are given sets $Types$, Con, Var and Fun. The elements of these sets are called **types**, **constants**, **variables**, and **function symbols (FS)**, respectively. Each element x of Con, Var and Fun is associates with some type $\tau(x) \in Types$, and if $x \in Fun$, then $\tau(x)$ has the form $(\tau_1, \ldots, \tau_n) \to \tau$, where $\tau_1, \ldots, \tau_n, \tau \in Types$.

2.1.2 Terms

The concept of a **term** 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 $\tau(x)$,
- if $f \in Fun$, $e_1, \ldots, e_n$ are terms, and $\tau(f)$ has the form

  $$(\tau(e_1), \ldots, \tau(e_n)) \to \tau,$$

  then the notation $f(e_1, \ldots, e_n)$ is the term of the type τ.

We will use the following notations:

- Tm denotes the set of all terms,
- $\forall\, e \in Tm \;\; Var_e$ denotes the set of all variables, occurred in e,
- $\forall\, X \subseteq Var \;\; Tm(X)$ denotes the set $\{e \in Tm \mid Var_e \subseteq X\}$,
- $\forall\, E \subseteq Tm, \forall\, \tau \in Types \; E^\tau$ denotes the set $\{e \in E \mid \tau(e) = \tau\}$.

Let $e, e' \in Tm$. The term e is called a **subterm** of the term e', if either $e = e'$, or e' has the form $f(e_1, \ldots, e_n)$, where $f \in Fun$, and $\exists\, i \in \{1, \ldots, n\}$: e is a subterm of the term e_i.

The notation $e \subseteq e'$, where $e, e' \in Tm$, means that e is a subterm of e'.

Below, for each considered function of the form $\varphi : E \to E'$, where $E, E' \subseteq Tm$, we will assume that $\forall\, e \in E \;\; \tau(\varphi(e)) = \tau(e)$.

2.1.3 Examples of types

We shall assume that *Types* has the following types:

- type **A**, terms of this type are called **agents**,
- type **C**, terms of this type are called **channels**, they denote communication channels used by agents for communication with each other by sending messages,
- type **K**, terms of this type are called **keys**, they denote cryptographic keys, that agents can use to encrypt or decrypt messages,
- type **M**, terms of this type are called **messages**, they denote messages, that agents can send to each other in the work flow,
- type **P**, terms of this type are called **processes**.

The notations *Agents*, *Channels*, *Keys* and *Processes* denote the sets of all agents, channels, keys, and processes, respectively.

We will use the following conventions and notations:

- *Channels* has a constant denoted by $\circ$, and called an **open channel**,
- an occurrence of a key k in a term e is said to be **hidden**, if this occurrence is first occurrence of k in a subterm of the form $k(e') \subseteq e$,
- $\forall\, A \in Var^{\mathbf{A}}$ the set Var has the variable $A^- \in Var^{\mathbf{K}}$, called the **private key** of agent A,
- type **M** includes any other types from *Types*, i.e. a term of any type is also a term of type **M**,
- $\forall\, n \geq 1$ set *Types* has type $\mathbf{M}_n$, whose values are tuples of length n, consisting of values of type **M**,
- set Var contains **shared variables**, each such variable has the form $x_{P_1 \ldots P_n}$, where $P_1, \ldots, P_n$ are different constants of the type **P**.

2.1.4 Examples of function symbols

We will assume that Fun contains the following FSs.

- FS $tuple_n$, where $n \geq 1$ and $\tau(tuple_n) = (\underbrace{\mathbf{M}, \ldots, \mathbf{M}}_{n}) \to \mathbf{M}_n$.

  For each list $(e_1, \ldots, e_n)$ of terms the term $tuple_n(e_1, \ldots, e_n)$ will be denoted by a shorter notation $(e_1, \ldots, e_n)$.

- FS $pr_{n,i}$, where $n \geq 1$, $i \in \{1, \ldots, n\}$, and $\tau(pr_{n,i}) = \mathbf{M}_n \to \mathbf{M}$.

  $\forall\, e \in Tm^{\mathbf{M}_n}$ the term $pr_{n,i}(e)$ is the i–th component of the tuple e, this term will be denoted by the notation $(e)_i$.

- FS h (possibly with indices) of type $\mathbf{M} \to \mathbf{M}$ type.

  The term $h(e)$ denotes the **hash function** value of the message e.

- FSs *encrypt* and *decrypt* of type $(\mathbf{K}, \mathbf{M}) \to \mathbf{M}$.

  Terms of the form $encrypt(k, e)$ and $decrypt(k, e)$ denote messages received by encrypting (and decrypting, respectively) the message e on the key k.

- FS *public_key* of type $\mathbf{A} \to \mathbf{K}$.

  Term of the form $public\_key(A)$ is called the **public key** of agent A.

  Terms of the form $encrypt(k, e)$ and $encrypt(public\_key(A), e)$ will be denoted by the notations $k(e)$ and $A(e)$ respectively, this terms are called **encrypted messages**.

- FS *dig_signature* of type $(\mathbf{M}, \mathbf{A}) \to \mathbf{M}$.

  A term of the form $dig\_signature(e, A)$ denotes a **digital signature** of the message e, made by agent A.

  The triple $(e, A, dig\_signature(e, A))$ will be denoted by $(e)_A$.

2.1.5 Expressions

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

- any set of terms $E \subseteq Tm$,
- X_P, where $P \in Processes$,
- M_c, where $c \in Channels$,

- $k^{-1}(E)$, where $k \in Keys$, and E is an expression,
- $E \cap E'$, $E \cup E'$, $\neg E$, where E, E' are expressions.

The set of all expressions is denoted by $Expr$. $\forall\, E \in Expr$ the notation Var_E denotes the set of all variables occurred in E.

If $E = \{e\}$, where $e \in Tm$, then such an expression will be denoted without brackets.

2.1.6 Formulas

An **elementary formula (EF)** is a notation of one of the following forms:

- $E = E'$, $E \subseteq E'$, $E \supseteq E'$, where $E, E' \in Expr$,
- $x \perp P$, $x \perp C$, where $x \in Var$, $P \in Processes$, $C \subseteq Channels$,
- $k \perp_{\mathbf{K}} P$, $k \perp_{\mathbf{K}} C$, where $k \in Keys$, $P \in Processes$, $C \subseteq Channels$.

Examples of EFs:

$$\left.\begin{array}{ll} decrypt(k, k(e)) = e, & \text{where } k \in Var^{\mathbf{K}}, e \in Tm \\ decrypt(A^-, A(e)) = e, & \text{where } A \in Var^{\mathbf{A}}, e \in Tm \\ pr_{n,i}(e_1, \ldots, e_n) = e_i, & \text{where } n > 0, i \in \{1, \ldots, n\}, \\ & e_1, \ldots, e_n \in Tm. \end{array}\right\} \quad (1)$$

A **formula** is a set of EFs. The set of all formulas is denoted by the notation Fm. $\forall\, \beta \in Fm$ the notation Var_β denotes the set of all variables, occurred in β.

Each formula $\beta \in Fm$ defines a congruence $\sim_\beta$ on Fun –algebra Tm: $\sim_\beta$ is an intersection of all congruences $\sim$ on Tm satisfying the condition: $\forall\, (e = e') \in \beta \quad e \sim e'$.

Below, the equality of terms is understood up to the congruence $\sim_\beta$, where β consists of EFs whose form coincides with one of the forms in (1).

2.1.7 Bindings

A **binding** is a function of the form $\theta : Var \to Tm$.

We say that a binding θ binds the variable $x \in Var$ with the term $\theta(x)$.

We will use the following notations:

- the set of all bindings is denoted by the symbol Θ,
- id denotes identical binding: $\forall\, x \in Var \;\; id(x) = x$,

- $\forall X \subseteq Var$ notation Θ_X denotes the set

$$\{\theta \in \Theta \mid \forall x \in Var \setminus X \;\; \theta(x) = x\},$$

- a binding $\theta \in \Theta$ can be denoted by the notations

$$x \mapsto \theta(x) \quad \text{or} \quad (\theta(x_1)/x_1, \ldots, \theta(x_n)/x_n)$$

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

- $\forall \theta \in \Theta$, $\forall e \in Tm$ the notation e^θ denotes a term derived from e by replacing $\forall x \in Var_e$ each occurrence of x in e by the term $\theta(x)$,

- $\forall \theta \in \Theta$, $\forall E \subseteq Tm$ the notation E^θ denotes the set $\{e^\theta \mid e \in E\}$,

- $\forall \theta, \theta' \in \Theta$ the notation $\theta\theta'$ denotes the binding $x \mapsto (x^\theta)^{\theta'}$.

2.2 Sequential processes

2.2.1 Actions

An **action** is a notation of one of the following forms:

$$c!e, \quad c?e, \quad e := e', \quad \text{where } c \in Channels,\; e, e' \in Tm,$$

which are called **sending** message e to channel c, **receiving** message e from channel c, and **assignment**, respectively.

Actions of the form $c!e$ and $c?e$ are called **external** actions, and actions of the form $e := e'$ are called **internal** actions.

The set of all actions is denoted by the notation Act. $\forall \alpha \in Act$ the set of all variables occurred in α, is denoted by the notation Var_α.

If $\theta \in \Theta$ and $\alpha \in Act$, then the notation α^θ denotes an action $c^\theta!e^\theta$, $c^\theta?e^\theta$ and $e^\theta := (e')^\theta$, if $\alpha = c!e$, $c?e$ and $e := e'$, respectively.

In some cases, to facilitate a perception, actions can be written in brackets, i.e., for example, instead of $c!e$, the notation $(c!e)$ might be used, etc.

2.2.2 A concept of a sequential process

A **sequential process (SP)** is a triple $(P, X, \bar{X})$, whose components have the following meaning:

- P is a graph with a selected node (called an **initial** node, and denoted by P^0), each edge of which is labeled by an action $\alpha \in Act$,

- $X \subseteq Var \cup Con$ is a set of **initialized variables** and constants, $\circ \in X$,

- $\bar{X} \subseteq X \cap Var$ is a set of **hidden variables**, these variables denote secret keys, hidden channels, and objects with unique values called **nonces**.

A SP is a formal description of the behavior of a dynamic system, which works by sequentially performing actions related to sending/receiving messages and initializing uninitialized variables.

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

- this SP can be abbreviated by the same symbol P as the corresponding graph, the set of nodes of the graph P also is denoted by P,
- nodes of graph P, which have no outgoing edges, are said to be **terminal** and are denoted by $\otimes$,
- notations X_P, $\bar{X}_P$ denote the corresponding components of the SP P,
- Var_P denotes the set of all variables occurred in P,
- if P has no edges and $X_P = \emptyset$, then P is denoted by $\mathbf{0}$.

Each SP is associated with a constant from $Processes$, called a **name** of this process. In order to simplify notations, we will denote the names of processes with the same notations that denote the processes themselves.

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

2.2.3 Adversary process

The **adversary process** is a SP P_* with the following features:

- the SP graph P_* has a single node,
- $Con \subseteq X_{P_*}$, $\forall\,\tau \in Types$ the sets $\bar{X}_{P_*}$ and $X_{P_*} \setminus \bar{X}_{P_*}$ have a countable set of variables of the type τ,
- $\forall\,\alpha \in Act$ graph P_* has an edge labeled by α.

Below we assume that P_* is the only SP under consideration, whose graph has cycles.

2.2.4 States of sequential processes

Let P be a SP. A **state** of P is a 4-tuple $s = (v, \alpha, X, \theta)$, where

- $v \in P$ is a **current node**,
- $\alpha \in \{init\} \sqcup Act$ is a **current action**,
- $X \subseteq Var$ is a **current set of initialized variables**, and
- $\theta \in \Theta$ is a **current binding**.

Components of s are denoted by v_s, α_s, X_s, and θ_s, respectively.

A state of the SP P is said to be **initial**, and is denoted by $\odot$, if it has the form $(P^0, init, X_P, id)$.

2.2.5 An execution of a sequential process

Let P be a SP. An **execution** of P can be understood as a walk through the graph P, starting from P^0, with the execution of actions that are labels of traversed edges.

Each step of an execution of P is associated with

- a state of P, called a **current state** at this step (a current state at first step is $\odot$), and
- a **current channels state**, which is a family of sets

$$M = \{M_c \subseteq Tm \mid c \in Channels\}.$$

If a current step of the execution of P is not a final step, then the following actions are performed at this step:

- the current state s on this step is changed on a state s', which will be a current state at the next step of the execution: if s has the form (v, α, X, θ), then there is selected an edge of P outgoing from v, whose label α' meets one of the folllowing conditions:

$$\left.\begin{array}{ll} \text{(a)} & \alpha' = c!e,\ c^\theta \in X^\theta,\ e \in Tm(X) \\ \text{(b)} & \alpha' = c?e,\ c^\theta \in X^\theta,\ \exists\,\hat{\theta} \in \Theta_{Var\setminus X} : (e^{\hat{\theta}})^\theta \in M_{c^\theta} \\ \text{(c)} & \alpha' = (e := e'),\ e' \in Tm(X),\ \exists\,\hat{\theta} \in \Theta_{Var\setminus X} : (e^{\hat{\theta}})^\theta = (e')^\theta \end{array}\right\} \quad (2)$$

  and components of $s' = (v', \alpha', X', \theta')$ have the following form: v' is the end of the selected edge, α' is the label of the selected edge, and

  - if (a) in (2) holds, then $X' = X$, $\theta' = \theta$,
  - if (b) or (c) in (2) holds, then $X' = X \cup Var_e$, $\theta' = \hat{\theta}\theta$, and

- a replacement of the current channels state M with the channels state M', which will be the current channels state at the next step of the execution: M' either is equal to M, or is obtained by adding terms to the sets from M, and
  - this adding can be performed by P as well as those SPs that use shared channels with P, and
  - if (a) in (2) holds, then one of such addings is that P adds the term e^θ to the set M_{c^θ}.

We will say that s' is obtained by a **transition** from s, and denote this by the notation $s \to s'$.

During each execution of each SP P the variables from Var_P have the following features: $\forall\, x \in Var_P$

1. if $x \not\in X_P$, then at the initial step of each execution of P the variable x is not initialized, i.e. there is no value associated with x,
2. if $x \in \bar{X}_P$ and x is not a shared variable, then at first step of each execution *Exec* of P this variable is associated with a **unique value**, i.e. such a value that differs from values associated with other initialized variables at *Exec*, and from values associated with initialized variables at any execution $Exec' \neq Exec$ of any SP,
3. if a variable from $\bar{X}_P$ is shared and has the form $x_{P_1 \ldots P_n}$, then
   - $P_1, \ldots, P_n$ is a list of names of all SPs, executed together with P (and P is one of the SPs in this list), which have the variable $x_{P_1 \ldots P_n}$ among his hidden variables, and
   - at the initial moment of each joint execution of SPs from the list $P_1, \ldots, P_n$ variable $x_{P_1 \ldots P_n}$ is initialized in all these SPs with the same value, which is unique, i.e. has the properties described in the point 2.

2.3 Operations on sequential processes

2.3.1 Prefix action

A **refined action** is a triple $\tilde{\alpha} = (\alpha, \hat{X}, \bar{X})$, where $\alpha \in Act$, and $\hat{X}, \bar{X}$ are disjoint subsets of the set Var_α.

We will denote the refined action $\tilde{\alpha} = (\alpha, \hat{X}, \bar{X})$ by the notation obtained from the notation of the action α by replacing each variable $x \in Var_\alpha$ to $\hat{x}$ or $\bar{x}$, if $x \in \hat{X}$ or $x \in \bar{X}$, respectively.

Let $\tilde{\alpha} = (\alpha, \hat{X}, \bar{X})$ be a refined action and P be a SP. An operation of a **prefix action** maps the pair $(\tilde{\alpha}, P)$ to a SP $\tilde{\alpha}.P$, having the following components:

- a graph of the SP $\tilde{\alpha}.P$ is obtained by adding
  - a new node v to P, which will be an initial node in $\tilde{\alpha}.P$, and
  - an edge $v \xrightarrow{\alpha} P^0$,
- $X_{\tilde{\alpha}.P} = (X_P \cup Var_\alpha) \setminus \hat{X}, \quad \bar{X}_{\tilde{\alpha}.P} = \bar{X}_P \cup \bar{X}$.

Below we will omit the symbol $\sim$ in the notations of the refined actions.

2.3.2 Choice

Let $P_I = \{P_i \mid i \in I\}$ be a family of SPs.

The notation $\sum_{i \in I} P_i$ denotes a SP $(P, X, \bar{X})$, called a **choice** from P_I. Its components are defined as follows:

- the graph P is obtained by adding to the union of disjoint copies of graphs from P_I
  - a new node P^0, which will be the initial one in P, and
  - edges $P^0 \xrightarrow{\alpha} v$, corresponding to edges of the form $P_i^0 \xrightarrow{\alpha} v$,
- X and $\bar{X}$ are unions of the corresponding components of SPs from P_I.

If the set of indices I has the form $\{1, \ldots, n\}$, then SP $\sum_{i \in I} P_i$ can also be denoted by $P_1 + \ldots + P_n$.

2.3.3 Renaming

A renaming is a partial injective function $\zeta : Var \to Var$, where for each shared variable $x_{P_1 \ldots P_n} \in Dom(\zeta)$ the variable $\zeta(x_{P_1 \ldots P_n})$ has the form $y_{P_1 \ldots P_n}$.

For each renaming ζ, each term e and each SP P the notations e^ζ and P^ζ denote a term or a SP respectively, obtained from e or P by replacing $\forall\, x \in Dom(\zeta)$ of each occurrence of x by $\zeta(x)$.

If $Var_P \subseteq Dom(\zeta)$, then the SPs P and P^ζ are assumed to be the same.

2.4 Distributed processes

2.4.1 A concept of a distributed process

Let $P_I = \{P_i \mid i \in I\}$ be a family of SPs.

$\forall\, i \in I$ let $\tilde{X}_{P_i}$ be a set of variables from Var_{P_i}, which either do not belong to X_{P_i}, or belong to $\bar{X}_{P_i}$ and are not shared.

We shall assume that for each family of SPs P_I under consideration the sets $\tilde{X}_{P_i}$ are disjoint (if this is not the case, then we rename accordingly variables in SPs from the family P_I).

A **distributed process (DP)** corresponding to the family P_I is an object denoted by the notation $\prod_{i\in I} P_i$. A DP is a model of a distributed algorithm, components of which are SPs from the family P_I, interacting by transmitting messages through channels. The meaning of a DP concept is explained in section 2.4.3.

If P is a DP of the form $\prod_{i\in I} P_i$, then

- $Var_P = \bigcup_{i\in I} Var_{P_i}$, $X_P = \bigcup_{i\in I} X_{P_i}$, $\bar{X}_P = \bigcup_{i\in I} \bar{X}_{P_i}$,
- if ζ is a renaming, then
  - the notation P^ζ denotes the DP $\prod_{i\in I} P_i^\zeta$,
  - if $Var_P \subseteq Dom(\zeta)$, then P and P^ζ are assumed to be the same,
- P can be denoted by the notation
  - $(P_1, \ldots, P_n)$, if $I = \{1, \ldots, n\}$, or
  - Q^∞, if I is a set of natural numbers, and all SPs in the family P_I coincide with the SP Q.

If $P_I = \{P_i \mid i \in I\}$ is a family of DPs, and each DP P_i in P_I corresponds to a family of SPs $\{Q_{i'} \mid i' \in I_i\}$, where the sets I_i $(i \in I)$ are disjoint (if this is not the case, then we will replace these sets with appropriate disjunctive copies), then the notation $\prod_{i\in I} P_i$ denotes a DP corresponding to the family of SPs $\{Q_i \mid i \in \bigsqcup_{i\in I} I_i\}$.

If DP P has the form $\prod_{i\in I} P_i$, then the notation P^* denotes the DP $\prod_{i\in I\sqcup\{*\}} P_i$, where P_* is the adversary process.

2.4.2 A concept of a state of a distributed process

Let P be a DP of the form $\prod_{i\in I} P_i$.

A **state** of P is a pair S of the following objects:

- a set $\{s^S_{P_i} \mid i \in I\}$ of states of SPs from P_I,

- a **channel state**: $M^S = \{M^S_c \subseteq Tm \mid c \in Channels\}$.

A state S of DP P is said to be **initial**, and is denoted by $\odot$, if

$$\forall\, i \in I \;\; s^S_{P_i} = \odot, \quad \forall\, c \in Channels \;\; M^S_c = \emptyset.$$

If S is a state of the DP $P = \prod_{i \in I} P_i$, and $i \in I$, then

- notations $v^S_{P_i}$, $\alpha^S_{P_i}$, $X^S_{P_i}$, $\theta^S_{P_i}$ denote the corresponding components of the state $s^S_{P_i}$,
- notation V^S denotes the set $\{v^S_{P_i} \mid i \in I\}$,
- notation θ^S denotes a binding, such that

$$\forall\, i \in I,\; \forall\, x \in X^S_{P_i} \quad \theta^S(x) = \theta^S_{P_i}(x).$$

2.4.3 An execution of a distributed process

Let P be a DP of the form $\prod_{i \in I} P_i$.

An **execution** of P can be understood as non-deterministic interliving of executions of SPs from P_I. At each step of an execution of P

- at most one SP from P_I performs its current action, and
- other SPs from P_I are in the waiting status.

An execution of a DP P can be formally defined as a generation of a sequence of states of this DP (starting with $\odot$), in which each state S that is not terminal, is associated with the next state S' by a **transition relation**, which means the following: $\exists\, i \in I$:

$$\begin{array}{l} s^S_{P_i} \to s^{S'}_{P_i}, \quad \forall\, i' \in I \setminus \{i\} \quad s^{S'}_{P_{i'}} = s^S_{P_{i'}}, \quad \text{and if } s^{S'}_{P_i} = (v, \alpha, X, \theta), \text{ then} \\ \text{if } \alpha = c!e, \text{ then } \left\{ \begin{array}{l} M^{S'}_{c^\theta} = M^S_{c^\theta} \cup \{e^\theta\}, \\ M^{S'}_{c'} = M^S_{c'} \text{ when } c' \neq c^\theta \end{array} \right\}, \text{otherwise } M^{S'} = M^S. \end{array} \tag{3}$$

For each states S, S' of DP P

- $S \to S'$ means that S is related with S' by a transition relation,
- $S \xrightarrow{\alpha_{P_i}} S'$ means that $S \to S'$, and (3) holds,
- $S \Rightarrow S'$ means that either $S = S'$, or there is a sequence $S_0, \ldots, S_n$ of states of P, such that

$$S_0 = S, \quad S_n = S', \quad \forall\, i = 0, \ldots, n-1 \quad S_i \to S_{i+1}.$$

A state S of P is said to be **reachable**, if $\odot \Rightarrow S$.

The set of reachable states of P is denoted by Σ_P.

2.5 Schemes of distributed processes

2.5.1 A concept of a scheme of a distributed process

Let P be a DP of the form $\prod_{i\in I}$, and $\forall\, i \in I$ SP P_i has the form

$$\alpha_1. \ \ldots \ \alpha_n.P'_i. \tag{4}$$

The sequence of actions $\alpha_1 \ldots \alpha_n$ and SP P'_i will be called a **prefix** and a **postfix** of SP P_i, respectively.

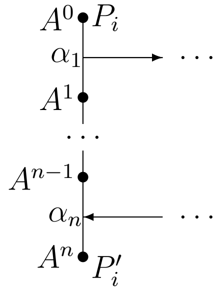
If

- each external actions in the prefix of P_i is a sending (receiving) a message to (from) a certain SP $P_j \in P_I$, and
- the action of SP P_j corresponding to the receiving (sending) this message is in the prefix of P_j,

then these dependencies between actions can be expressed as a **scheme** of DP P, which has the following form:

- each SP $P_i \in P_I$ is represented in this scheme by a **thread**, i.e. by a vertical line, on which there are marked points corresponding to nodes of the graph P_i belonging to the prefix of P_i (the upper point of the thread corresponds to P_i^0), and
  - near each such point it might be specified an identifier of the corresponding node,
  - near the upper point of the thread a name of SP P_i is specified,
  - if $P'_i \neq \mathbf{0}$, then the postfix name P'_i is specified near the bottom point of the thread,
  - the segments connecting the neighboring points of the thread correspond to edges of P_i related to the prefix of P_i, there are the specified labels of the corresponding edges beside these segments,
- for each segment O of the thread connecting neighboring points, if the corresponding action is sending a message, then there is an arrow in the scheme, such that
  - the start of this arrow lies on the segment O, and
  - the end of this arrow lies on the segment O', the label of which is an action of the corresponding SP $P_j \in P_I$ to receive this message.

For example if $P_i = \alpha_1. \ \ldots \ \alpha_n.P'_i$, where α_1 is a sending, and α_n is a receiving, and $A^0, \ldots, A^n$ are identifiers of the corresponding nodes of P_i, then a thread corresponding to P_i has the following form:



(5)

2.5.2 Examples of schemes of distributed processes

1. First example is a DP consisting of two SPs named A and B, which is a model for transmitting one message x from A to B through a hidden channel c_{AB} (only A and B know the name of this channel).

   This DP works as follows:

   - A sends B the message x through channel c_{AB},
   - B receives a message from channel c_{AB}, writes this message to the variable y, and then it behaves in the same way as the SP P.

   SPs A and B are defined as follows:

$$A = (\bar{c}_{AB}!x).\mathbf{0}, \quad B = (\bar{c}_{AB}?\hat{y}).P.$$

   The scheme of the DP (A, B) has the following form:

(6)

2. Second example is a DP consisting of two SPs named A and B, which is a model of transmission an encrypted message $k_{AB}(x)$ from A to B through the 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 the key k_{AB}.

   This DP works as follows:

   - A sends B an encrypted message $k_{AB}(x)$ to channel $\circ$,

- B receives the message $k_{AB}(x)$ from channel $\circ$, decrypts it, writes the extracted message x into the variable y, and then behaves in the same way as SP P.

SPs A and B are defined as follows:

$$A = (!\bar{k}_{AB}(x)).\mathbf{0}, \quad B = (?\bar{k}_{AB}(\hat{y})).P.$$

A scheme of DP (A, B) has the following form:

(7)

3. Third example is a DP consisting of three SPs named A, B, and T, which is a model for transmission one message x from A to B through a hidden channel c_{AB}, using a **trusted intermediary** T, where A and T (B and T) communicate through a hidden channel c_{AT} (c_{BT}), and only A and T (B and T) know the name of this channel.

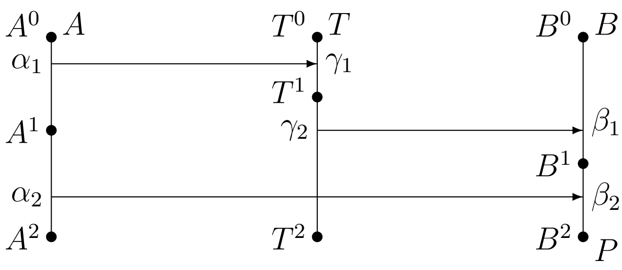
   This DP works as follows:

   - A sends T channel name c_{AB} (only A knows name c_{AB} at first) through channel c_{AT},
   - T sends B received channel name c_{AB} through channel c_{BT},
   - A sends B message x through channel c_{AB},
   - B receives a message from channel c_{AB} and writes it to variable y and then it behaves in the same way as SP P.

   SPs A, B and T are defined as follows:

$$\begin{array}{lll} A = \alpha_1.\alpha_2.\mathbf{0}, & \text{where} \quad \alpha_1 = \bar{c}_{AT}!\bar{c}_{AB}, & \alpha_2 = \bar{c}_{AB}!x, \\ T = \gamma_1.\gamma_2.\mathbf{0}, & \text{where} \quad \gamma_1 = \bar{c}_{AT}?\hat{u}, & \gamma_2 = \bar{c}_{BT}!u, \\ B = \beta_1.\beta_2.P, & \text{where} \quad \beta_1 = \bar{c}_{BT}?\hat{v}, & \beta_2 = v?\hat{y}. \end{array} \tag{8}$$

   A scheme of DP (A, B, T) has the following form:



(9)

4. Fourth example is a DP (called a **Wide-Mouth Frog (WMF)** protocol), consisting of three SPs named A, B and T (where T is a trusted intermediary). This DP is a model of a transmission of encrypted message $k_{AB}(x)$ from A to B through open channel $\circ$ with use of T, with whom A and B communicate through open channel $\circ$. SP A

   - creates the secret key k_{AB},
   - sends B this key in an encrypted form using T, and then
   - sends B encrypted message $k_{AB}(x)$.

   It is assumed that A and T (B and T) have a shared secret key k_{AT} (k_{BT}), on which they can encrypt and decrypt messages using a symmetric encryption system, and only A and T (B and T) know secret key k_{AT} (k_{BT}).

   This DP works as follows.

   - A creates a secret key k_{AB} (at first only A knows this key) and sends T encrypted message $k_{AT}(k_{AB})$ through $\circ$, then A sends B encrypted message $k_{AB}(x)$ through $\circ$,
   - T receives a message from A, decrypts this message, then encrypts the extracted key k_{AB} with the key k_{BT}, and sends B encrypted message $k_{BT}(k_{AB})$ through $\circ$,
   - B extracts key k_{AB} from the message received from T, and then uses this key to extract message x from the message received from A, writes x to variable y, and then behaves in the same way as SP P.

   SPs A, B and T are defined as follows:

   $$\begin{array}{llll} A = \alpha_1.\alpha_2.\mathbf{0}, & \text{where} & \alpha_1 = !\bar{k}_{AT}(\bar{k}_{AB}), & \alpha_2 = !\bar{k}_{AB}(x), \\ T = \gamma_1.\gamma_2.\mathbf{0}, & \text{where} & \gamma_1 = ?\bar{k}_{AT}(\hat{u}), & \gamma_2 = !\bar{k}_{BT}(u), \\ B = \beta_1.\beta_2.P, & \text{where} & \beta_1 = ?\bar{k}_{BT}(\hat{v}), & \beta_2 = ?v(\hat{y}). \end{array} \tag{10}$$

   A scheme of DP (A, B, T) has the same form (9), as the scheme of the previous DP.

2.6 Transition graphs of distributed processes

2.6.1 A concept of a transition graph of a distributed process

Let P be a DP of the form $\prod_{i\in I} P_i$.

A **transition graph (TG)** of DP P is a graph G_P such that

- a set of nodes of G_P is the Cartesian product of the sets of nodes of graphs from P_I, i.e. each node of G_P is a family of nodes

$$V = \{v_i \mid i \in I\}, \text{ where } \forall\, i \in I \;\; v_i \in P_i,$$

- each edge of G_P has the form

$$\{v_i \mid i \in I\} \xrightarrow{\alpha_{P_i}} \{v'_i \mid i \in I\}, \tag{11}$$

  where P_i has the edge $v_i \xrightarrow{\alpha} v'_i$ and $\forall\, i' \in I \setminus \{i\} \quad v_{i'} = v'_{i'}$.

The node $\{P_i^0 \mid i \in I\} \in G_P$ is said to be an **initial** node of G_P, and is denoted by G_P^0. An edge $V \xrightarrow{\alpha_{P_i}} V'$ is said to be a **realizable** edge, if $\exists\, S, S' \in \Sigma_P$: $V = V^S$ and $V' = V^{S'}$.

It is not difficult to prove that if $\forall\, i \in I$ P_i is acyclic, then G_P is acyclic.

For each DP P the graph G_{P^*} can be considered as a completion of the graph G_P with cyclic edges corresponding to the actions of P_*.

If DP P has the form $(P_1, \ldots, P_n)$, then the following conventions will be used in a graphical representation of G_P:

- each node $V = \{v_i \mid i = 1, \ldots, n\}$ of G_P is represented by an oval, there is a list $v_1 \ldots v_n$ of components of V inside this oval,

- an initial node G_P^0 is represented by a double oval.

2.6.2 Examples of transition graphs of distributed processes

In this section we outline some examples of TGs for DPs described by schemes from section 2.5.

1. A TG for a DP described by scheme (6):

$$\begin{tikzcd} A^0B^0 \arrow[r, "(\bar{c}_{AB}?\hat{y})_B"] \arrow[d, "(\bar{c}_{AB}!x)_A"'] & A^0B^1 \arrow[d, "(\bar{c}_{AB}!x)_A"] \arrow[r] & \cdots \\ A^1B^0 \arrow[r, "(\bar{c}_{AB}?\hat{y})_B"] & A^1B^1 \arrow[r] & \cdots \end{tikzcd} \tag{12}$$

   where the slanted arrows denote

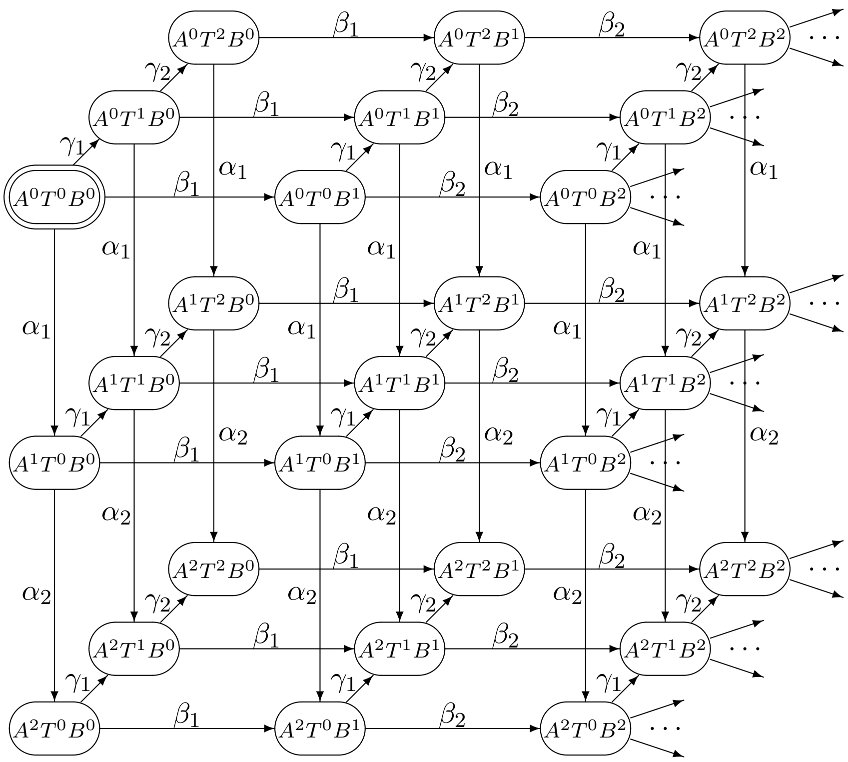
   - edges of G_P outgoing from the corresponding nodes,
   - and parts of G_P reachable after passing through these edges,

which are not represented in this picture, this convention will be used in the following TG examples as well.

2. A TG for a DP described by scheme (7):

$$\begin{tikzcd} A^0B^0 \arrow[r, "?\bar{k}_{AB}(\hat{y})"] \arrow[d, "!\bar{k}_{AB}(x)"'] & A^1B^0 \arrow[d, "!\bar{k}_{AB}(x)"] \arrow[r] & \cdots \\ A^1B^0 \arrow[r, "?\bar{k}_{AB}(\hat{y})"] & A^1B^1 \arrow[r] & \cdots \end{tikzcd} \tag{13}$$

3. A TG for a DP described by scheme (9):



(14)

2.7 Values of expressions and formulas in states of distributed processes

2.7.1 A concept of a value of an expression and a formula in a state of a distributed process

Let there are given the DP $P = \prod_{i\in I} P_i$, the state $S \in \Sigma_P$, the expression $E \in Expr$, and the formula $\beta \in Fm$.

The notation E^S denotes a subset of the set Tm, called a **value of the expression E in the state S**, and defined as follows:

- if $E \subseteq Tm$, then $E^S = E^{\theta^S}$,
- if $E = X_P$, then $E^S = (X_P^S)^{\theta^S}$,
- if $E = M_c$, then $E^S = M^S_{c^{\theta^S}}$,
- if $E = k^{-1}(E')$, then $E^S = \{e \in Tm \mid \exists\, e' \in (E')^S : k^{\theta^S}(e) \subseteq e'\}$,
- $(E \cap E')^S = E^S \cap (E')^S$, $(E \cup E')^S = E^S \cup (E')^S$, $(\neg E)^S = Tm \setminus E^S$.

The notation $S \models \beta$ denotes the statement β **holds in** S, which is true iff one of the following cases holds:

-
  - $\beta = (E = E')$, $(E \subseteq E')$, or $(E \supseteq E')$, where $E, E' \in Expr$, and
  - $E^S = (E')^S$, $E^S \subseteq (E')^S$, or $E^S \supseteq (E')^S$, respectively,
-
  - $\beta = (x \perp P_i)$, where $x \in Var$, $i \in I$, and
  - $\forall\, e \in (X_{P_i}^S)^{\theta^S} \;\; x \notin Var_e$,
-
  - $\beta = (x \perp C)$, where $x \in Var$, $C \subseteq Channels$, and
  - $\forall\, c \in C$, $\forall\, e \in M_c^S \;\; x \notin Var_e$,
-
  - $\beta = (k \perp_{\mathbf{K}} P_i)$, where $k \in Keys$, $i \in I$, and
  - $\forall\, e \in (X_{P_i}^S)^{\theta^S}$ each occurrence of k in e is hidden,
-
  - $\beta = (k \perp_{\mathbf{K}} C)$, where $k \in Keys$, $C \subseteq Channels$, and
  - $\forall\, c \in C$, $\forall\, e \in M_c^S$ each occurrence of k in e is hidden,
- $\beta = \{\beta_i \mid i \in I\}$ is a family of EFs, $\forall\, i \in I \;\; S \models \beta_i$.

2.7.2 Theorems on preserving values of formulas under transitions

Below we prove theorems that some formulas have the same values in states related by a transition relation.

Theorem 1.
Let $P = \prod_{i\in I} P_i$ be a DP and $S, S' \in \Sigma_P$ be states such that

$$\exists\, i \in I : \; S \stackrel{\alpha_{P_i}}{\longrightarrow} S'.$$

Then the implication $S \models \beta \;\; \Rightarrow \;\; S' \models \beta$ holds, where β is a formula of one of the following forms:

1. $\beta = \{x \perp P_i, x \perp Channels\}$, where $x \in X_P$,
2. $\beta = \{k \perp_{\mathbf{K}} P_i, k \perp_{\mathbf{K}} Channels\}$, where $k \in X^{\mathbf{K}}_{P_i}$.

Proof.

1. Let $\beta = \{x \perp P_i, x \perp Channels\}$, where $x \in X_P$.
   $S \models \beta$ means that

$$\left.\begin{array}{l} \forall\, y \in X^S_{P_i} \quad x \not\in Var_{y^{\theta^S}} \\ \forall\, c \in Channels, \forall\, e \in M^S_c \quad x \not\in Var_e. \end{array}\right\} \tag{15}$$

   It is required to prove that (15) implies $S' \models \beta$, i.e.

$$\left.\begin{array}{l} \forall\, y \in X^{S'}_{P_i} \quad x \not\in Var_{y^{\theta^{S'}}} \\ \forall\, c \in Channels, \forall\, e \in M^{S'}_c \quad x \not\in Var_e. \end{array}\right\} \tag{16}$$

   If first statement in (16) is wrong, then first statement in (15) implies that $X^S_{P_i} \neq X^{S'}_{P_i}$. This is only possible if

$$\begin{array}{l} \alpha \text{ is of the form } c?e,\ X^{S'}_{P_i} = X^S_{P_i} \cup Var_e, \\ e^{\theta^{S'}} \in M^S_{c^{\theta^S}}, \text{ and } \exists\, y \in Var_e : x \in y^{\theta^{S'}} \;\; (\Rightarrow \;\; x \in Var_{e^{\theta^{S'}}}). \end{array} \tag{17}$$

   (17) contradicts second statement in (15).

   If second statement in (16) is wrong, then second statement in (15) implies that $\exists\, c \in Channels : M^S_c \neq M^{S'}_c$. This is only possible if

$$\left.\begin{array}{l} \alpha \text{ has the form } c'!e, \text{ where } (c')^{\theta^S} = c, \text{ and } e \in Tm(X^S_{P_i}), \\ M^{S'}_c = M^S_c \cup \{e^{\theta^S}\}, \text{ and } x \in Var_{e^{\theta^S}}. \end{array}\right\} \tag{18}$$

Denote by symbols X and θ the set $X^S_{P_i}$ and the binding θ^S, respectively. From (18) it follows that $e \in Tm(X)$ and $x \in Var_{e^\theta}$.

From $x \in Var_{e^\theta}$ it follows that $\exists\, y \in Var_e : x \in Var_{y^\theta}$.

From $e \in Tm(X)$ and $y \in Var_e$ it follows that $y \in X$, so $y^\theta \in X^\theta$.

Thus, we get the statements

$$y^\theta \in X^\theta, \quad x \in Var_{y^\theta}$$

that contradict first statement in (15).

2. Let $\beta = \{k \perp_{\mathbf{K}} P_i, k \perp_{\mathbf{K}} Channels\}$, where $k \in X^{\mathbf{K}}_{P_i}$.

   $S \models \beta$ means that

$$\left.\begin{array}{l} \forall\, x \in X^S_{P_i} \text{ each occurrence of } k \text{ in } x^{\theta^S} \text{ is hidden,} \\ \forall\, c \in Channels,\ \forall\, e \in M^S_c \text{ each occurrence of } k \text{ in } e \text{ is hidden.} \end{array}\right\} \quad (19)$$

   It is required to prove that (19) implies $S' \models \beta$, i.e.

$$\left.\begin{array}{l} \forall\, x \in X^{S'}_{P_i} \text{ each occurrence of } k \text{ in } x^{\theta^{S'}} \text{ is hidden,} \\ \forall\, c \in Channels,\ \forall\, e \in M^{S'}_c \text{ each occurrence of } k \text{ in } e \text{ is hidden.} \end{array}\right\} \quad (20)$$

   If first statement in (20) is wrong, then first statement in (19) implies that $X^S_{P_i} \neq X^{S'}_{P_i}$. This is possible in the following two cases:

$$\text{(a)} \left\{\begin{array}{l} \alpha \text{ is of the form } c?e,\ X^{S'}_{P_i} = X^S_{P_i} \cup Var_e, \\ e^{\theta^{S'}} \in M^S_{c^{\theta^S}}, \text{ and } \exists\, y \in Var_e: \\ \qquad\qquad \exists \text{ unhidden occurrence of } k \text{ in } y^{\theta^{S'}}, \end{array}\right.$$

$$\text{(b)} \left\{\begin{array}{l} \alpha \text{ is of the form } e := e',\ X^{S'}_{P_i} = X^S_{P_i} \cup Var_e, \\ e^{\theta^{S'}} = (e')^{\theta^S}, \text{ and } \exists\, y \in Var_e: \\ \qquad\qquad \exists \text{ unhidden occurrence of } k \text{ in } y^{\theta^{S'}}. \end{array}\right.$$

   In case 2(b)i $\exists$ unhidden occurrence of k in $e^{\theta^{S'}}$, that contradicts second statement in (19).

   In case 2(b)ii the following is true:

$$\exists\, e' \in Tm(X^S_{P_i}) : \exists \text{ unhidden occurrence of } k \text{ in } (e')^{\theta^S}. \qquad (21)$$

However, according to first statement in (19), $\forall\, x \in X^S_{P_i}$ each occurrence of k in x^{θ^S} is hidden, whence by induction on the structure of e' it is easy to prove that (21) is false.

If second statement in (20) is wrong, then second statement in (19) implies that $\exists\, c \in Channels : M^S_c \neq M^{S'}_c$. This is only possible if

$$\begin{array}{l} \alpha \text{ has the form } c'!e, \text{ where } (c')^{\theta^S} = c, \text{ and } e \in Tm(X^S_{P_i}), \\ M^{S'}_c = M^S_c \cup \{e^{\theta^S}\}, \text{ and } \exists \text{ an unhidden occurrence of } k \text{ in } e^{\theta^S}. \end{array} \quad (22)$$

As in previous case, we prove by induction on the structure of e that each occurrence of k in e^{θ^S} is hidden (for the base of induction we use first statement in (19)) that contradicts the last statement in (22). ■

Theorem 2.
Let $P = \prod_{i\in I} P_i$ be a DP, and $S, S' \in \Sigma_P$ be states such that

$$\exists\, i \in I : \; S \stackrel{\alpha_{P_i}}{\longrightarrow} S'.$$

Then the implication $S \models \beta \;\Rightarrow\; S' \models \beta$ holds, where β is a formula of one of the following two forms:

$$\{c \perp P_i, c \perp Channels, M_c = E\}, \text{ where } c \in X^{\mathbf{C}}_P, \quad (23)$$

$$\left\{\begin{array}{l} k \perp_{\mathbf{K}} P_i, \; k \perp_{\mathbf{K}} Channels, \\ k^{-1}(M_c) \subseteq E \quad (\forall\, c \in Channels), \\ k^{-1}(X_{P_i}) \subseteq E \end{array}\right\}, \text{where } k \in X^{\mathbf{K}}_P, E \subseteq Tm(X_P). \quad (24)$$

Proof.

1. Let β has the form (23).

   According to theorem 1, if first two EFs occurred in β hold in S, then these EFs hold in S' as well.

   Thus, to prove $S' \models \beta$ it suffices to prove the implication

   $$S \models \{c \perp P_i, c \perp Channels, M_c = E\} \;\;\Rightarrow\;\; S' \models M_c = E. \quad (25)$$

   If the conclusion of implication (25) does not hold, then the sets M^S_c and $M^{S'}_c$ are different. This is possible only if α is of the form $c'!e$, where $c = (c')^{\theta^S}$ and $c' \in X^S_{P_i}$. However, $S \models c \perp P_i$ implies that $c \not\in Var_{(c')^{\theta^S}}$, i. e. $c \not\in \{c\}$, which is impossible.

2. Let β has the form (24).

   According to theorem 1, if first two EFs occurred in β hold in S, then these EFs hold in S' as well.

   Thus, to prove $S' \models \beta$ it suffices to prove the implication

$$\left.\begin{array}{l} S \models k^{-1}(M_c) \subseteq E \quad (\forall\, c \in Channels) \\ S \models k^{-1}(X_{P_i}) \subseteq E \\ S \models \{k \perp_{\mathbf{K}} P_i, k \perp_{\mathbf{K}} Channels\} \end{array}\right\} \Rightarrow \\ \Rightarrow \left\{\begin{array}{l} S' \models k^{-1}(M_c) \subseteq E \quad (\forall\, c \in Channels) \\ S' \models k^{-1}(X_{P_i}) \subseteq E \end{array}\right. \tag{26}$$

   (a) If first statement in the conclusion of implication (26) is wrong, then $\exists\, c \in Channels : S' \not\models k^{-1}(M_c) \subseteq E$.

   From first statement in the premise of implication (26) it follows that this is possible only if

$$\begin{array}{l} \alpha \text{ has the form } c'!e', \text{ where } c = (c')^{\theta^S}, e' \in Tm(X^S_{P_i}), \\ M^{S'}_c = M^S_c \cup \{(e')^{\theta^S}\}, \text{ with } \exists\, k(e) \subseteq (e')^{\theta^S}\!:\ e \notin E. \end{array}$$

   The term e' does not contain k, because $e' \in Tm(X^S_{P_i})$, and if e' contains k, then $k \in X^S_{P_i}$, which contradicts the assumption $S \models k \perp_{\mathbf{K}} P_i$ in the premise of implication (26).

   Thus, $\exists\, x \in Var_{e'} \subseteq X^S_{P_i}$, $\exists\, k(e) \subseteq x^{\theta^S}$, and $e \notin E$.

   However, this contradicts the statement $S \models k^{-1}(X_{P_i}) \subseteq E$ in the premise of implication (26).

   (b) If second statement in the conclusion of implication (26) does not hold, then from second statement in the premise of implication (26) it follows that $X^S_{P_i} \neq X^{S'}_{P_i}$, and

$$\exists\, x \in X^{S'}_{P_i} : \exists\, e \notin E : k(e) \subseteq x^{\theta^{S'}}. \tag{27}$$

   This is possible in two cases:

   i. $\alpha = c?e'$, in this case

$$X^{S'}_{P_i} = X^S_{P_i} \cup Var_{e'}, \quad x \in Var_{e'}, \quad (e')^{\theta^{S'}} \in M^S_{c^{\theta^S}}. \tag{28}$$

   Let $c' = c^{\theta^S}$.

According to first statement in the premise of implication (26), $S \models k^{-1}(M_{c'}) \subseteq E$, so the following implication holds:

$$k(e) \subseteq \tilde{e} \in M^S_{c'} \;\Rightarrow\; e \in E. \tag{29}$$

The premise of implication (29) holds when $\tilde{e} = (e')^{\theta^{S'}}$, this follows from the last statement in (30) and from

$$k(e) \subseteq x^{\theta^{S'}}, \;\; x \in Var_{e'}, \;\; x^{\theta^{S'}} \subseteq (e')^{\theta^{S'}} \in M^S_{c^{\theta^S}}.$$

Thus, the conclusion of implication (29) holds, which contradicts the statement $e \notin E$ in (27).

ii. $\alpha = (e' := e'')$, in this case

$$\begin{array}{l} X^{S'}_{P_i} = X^S_{P_i} \cup Var_{e'}, \;\; x \in Var_{e'}, \\ e'' \in Tm(X^S_{P_i}), \;\; (e')^{\theta^{S'}} = (e'')^{\theta^S}. \end{array} \tag{30}$$

According to second statement in the premise of implication (26), $S \models k^{-1}(X_{P_i}) \subseteq E$, so the following implication holds:

$$k(e) \subseteq \tilde{e} \in (X^S_{P_i})^{\theta^S} \;\Rightarrow\; e \in E. \tag{31}$$

Since $x \in Var_{e'}$, then

$$x^{\theta^{S'}} \subseteq (e')^{\theta^{S'}} = (e'')^{\theta^S}.$$

The last statements and (27) imply the statements

$$k(e) \subseteq (e'')^{\theta^S} \in (Tm(X^S_{P_i}))^{\theta^S}. \tag{32}$$

The term e'' does not contain k, because $e'' \in Tm(X^S_{P_i})$, and if e'' contains k, then $k \in X^S_{P_i}$, which contradicts the assumption $S \models k \perp_{\mathbf{K}} P_i$ in the premise of implication (26).
Hence, based on (32), we obtain

$$\exists\, y \in Var_{e''} \subseteq X^S_{P_i} : k(e) \subseteq y^{\theta^S}. \tag{33}$$

From (33) it follows that if we define $\tilde{e}$ as the term y^{θ^S}, then the premise of implication (31) will be true.
Consequently, a conclusion of this implication will also be true, i.e. the statement $e \in E$ is true, which contradicts the assumption $e \notin E$ in (27). ■

Theorem 3.

Formula (24) in theorem 2 can be replaced by a formula β of the form

$$\left\{\begin{array}{l} k \perp_{\mathbf{K}} P_i,\ k \perp_{\mathbf{K}} Channels, \\ k^{-1}(M_{c_0}) = E \\ k^{-1}(M_c) \subseteq E \quad (\forall\, c \in Channels), \\ k^{-1}(X_{P_i}) \subseteq E \end{array}\right\} \tag{34}$$

where $c_0 \in X_P^{\mathbf{C}}$, $k \in X_P^{\mathbf{K}}, E \subseteq Tm(X_P)$.

Proof.

If $S \models \beta$, then $S \models \beta'$, where β' is obtained from β by removing the formula $k^{-1}(M_{c_0}) = E$.

According to theorem 2, the statement $S \models \beta'$ implies the statement $S' \models \beta'$. In particular, $S' \models k^{-1}(M_{c_0}) \subseteq E$.

(3) implies the inclusion $M_{c_0}^S \subseteq M_{c_0}^{S'}$, from which we obtain the statements

$$E = (k^{-1}(M_{c_0}))^S \subseteq (k^{-1}(M_{c_0}))^{S'} \subseteq E,$$

therefore, $S' \models k^{-1}(M_{c_0}) = E$. Thus, $S' \models \beta$. ■

2.8 Marking of a transition graph

2.8.1 A concept of a marking of a transition graph

Let P be a DP of the form $\prod_{i\in I} P_i$.

A **marking** of the TG G_P is a pair

$$(G, \{\beta_V \in Fm \mid V \in G\}) \tag{35}$$

where G is a subset of the set of nodes of G_P, such that

- $G_P^0 \in G$, and
- $\forall\, V \in G$, if G_P has an edge of the form $V' \to V$, then $V' \in G$.

Marking (35) is said to be **correct**, if

- $G_P^0 \models \beta_{G_P^0}$, and
- $\forall\, S, S' \in \Sigma_P$, if $S \to S'$ and $V^S, V^{S'} \in G$, then the following implication holds:
$$S \models \beta_{V^S} \quad \Rightarrow \quad S' \models \beta_{V^{S'}}.$$

It was noted in section 2.6.1 that for each DP P the graph G_{P^*} is a completion of the graph G_P with cyclic edges corresponding to actions of the adversary P_*. Therefore, for each DP P, any marking of the TG G_P can also be considered as a marking of the corresponding TG G_{P^*}.

Below, a marking of any TG G_P is said to be correct, if it is a correct marking (in the sense of the above definition) of the corresponding TG G_{P^*}.

2.8.2 Examples of correct markings of transition graphs

In this section we present examples of correct markings for TGs from section 2.6.2. The correctness of all the markings listed below can be justified the theorems from section 2.7.2.

Below we denote nodes of TGs by lists of nodes of corresponding SPs.

1. For TG (12) one of correct markings has the form

$$G = \{A^0B^0, A^1B^0, A^1B^1\}$$

and

$$\beta_{A^0B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} M_{c_{AB}} = \emptyset \\ c_{AB} \perp P_* \\ c_{AB} \perp Channels \end{array} \right\}, \ \beta_{A^1B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} M_{c_{AB}} = \{x\} \\ c_{AB} \perp P_* \\ c_{AB} \perp Channels \end{array} \right\}$$
$$\beta_{A^1B^1} \stackrel{\text{def}}{=} \{x = y\}$$

2. For TG (13) one of correct markings has the form

$$G = \{A^0B^0, A^1B^0, A^1B^1\}$$

and

$$\beta_{A^0B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} k_{AB}^{-1}(M_\circ) = \emptyset \\ k_{AB} \perp_{\mathbf{K}} P_* \\ k_{AB} \perp_{\mathbf{K}} Channels \end{array} \right\}, \ \beta_{A^1B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} {k_{AB}}^{-1}(M_\circ) = \{x\} \\ k_{AB} \perp_{\mathbf{K}} P_* \\ k_{AB} \perp_{\mathbf{K}} Channels \end{array} \right\},$$
$$\beta_{A^1B^1} \stackrel{\text{def}}{=} \{x = y\}$$

3. For TG (14), where actions $\alpha_i, \beta_i, \gamma_i$ $(i = 1, 2)$ are defined according to (8), one of correct markings has the form

$$G = \left\{ \begin{array}{l} A^0T^0B^0, A^1T^0B^0, A^2T^0B^0, A^1T^1B^0, A^2T^1B^0, \\ A^1T^2B^0, A^2T^2B^0, A^1T^2B^1, A^2T^2B^1, A^2T^2B^2 \end{array} \right\} \quad (36)$$

and

- $\beta_{A^0T^0B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} M_{c_{AT}} = M_{c_{BT}} = M_{c_{AB}} = \emptyset \\ \{c_{AT}, c_{BT}, c_{AB}\} \perp P_* \\ \{c_{AT}, c_{BT}, c_{AB}\} \perp Channels \end{array} \right\},$
- $\beta_{A^1T^0B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} M_{c_{AT}} = \{c_{AB}\} \\ M_{c_{BT}} = M_{c_{AB}} = \emptyset \\ \{c_{AT}, c_{BT}, c_{AB}\} \perp P_* \\ \{c_{AT}, c_{BT}\} \perp Channels \end{array} \right\},$
- $\beta_{A^2T^0B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} M_{c_{AT}} = \{c_{AB}\} \\ M_{c_{BT}} = \emptyset \\ M_{c_{AB}} = \{x\} \\ \{c_{AT}, c_{BT}, c_{AB}\} \perp P_* \\ \{c_{AT}, c_{BT}\} \perp Channels \end{array} \right\},$
- $\beta_{A^1T^1B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = c_{AB} \\ M_{c_{AT}} = \{c_{AB}\} \\ M_{c_{BT}} = M_{c_{AB}} = \emptyset \\ \{c_{AT}, c_{BT}, c_{AB}\} \perp P_* \\ \{c_{AT}, c_{BT}\} \perp Channels \end{array} \right\},$
- $\beta_{A^2T^1B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = c_{AB} \\ M_{c_{AT}} = \{c_{AB}\} \\ M_{c_{BT}} = \emptyset \\ M_{c_{AB}} = \{x\} \\ \{c_{AT}, c_{BT}, c_{AB}\} \perp P_* \\ \{c_{AT}, c_{BT}\} \perp Channels \end{array} \right\},$
- $\beta_{A^1T^2B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = c_{AB} \\ M_{c_{AT}} = \{c_{AB}\} \\ M_{c_{BT}} = \{u\} \\ M_{c_{AB}} = \emptyset \\ \{c_{AT}, c_{BT}, c_{AB}\} \perp P_* \\ \{c_{AT}, c_{BT}\} \perp Channels \end{array} \right\},$
- $\beta_{A^2T^2B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = c_{AB} \\ M_{c_{AT}} = \{c_{AB}\} \\ M_{c_{BT}} = \{u\} \\ M_{c_{AB}} = \{x\} \\ \{c_{AT}, c_{BT}, c_{AB}\} \perp P_* \\ \{c_{AT}, c_{BT}\} \perp Channels \end{array} \right\},$
- $\beta_{A^1T^2B^1} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = c_{AB} \\ v = u \\ M_{c_{AT}} = \{c_{AB}\} \\ M_{c_{BT}} = \{u\} \\ M_{c_{AB}} = \emptyset \\ \{c_{AT}, c_{BT}, c_{AB}\} \perp P_* \\ \{c_{AT}, c_{BT}\} \perp Channels \end{array} \right\},$
- $\beta_{A^2T^2B^1} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = c_{AB} \\ v = u \\ M_{c_{AT}} = \{c_{AB}\} \\ M_{c_{BT}} = \{u\} \\ M_{c_{AB}} = \{x\} \\ \{c_{AT}, c_{BT}, c_{AB}\} \perp P_* \\ \{c_{AT}, c_{BT}\} \perp Channels \end{array} \right\},$

- $\beta_{A^2T^2B^2} \stackrel{\text{def}}{=} \{y = x\}$

4. For TG (14), where actions $\alpha_i, \beta_i, \gamma_i$ $(i = 1, 2)$ are defined according to (10), one of correct markings has the form:

   - G has the same form, as in (36), and
   - – $\beta_{A^0T^0B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} k_{AT}^{-1}(M_\circ) = k_{BT}^{-1}(M_\circ) = k_{AB}^{-1}(M_\circ) = \emptyset \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} P_* \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} Channels \end{array} \right\}$,

     – $\beta_{A^1T^0B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} k_{AT}^{-1}(M_\circ) = \{k_{AB}\} \\ k_{BT}^{-1}(M_\circ) = k_{AB}^{-1}(M_\circ) = \emptyset \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} P_* \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} Channels \end{array} \right\}$,

     – $\beta_{A^2T^0B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} k_{AT}^{-1}(M_\circ) = \{k_{AB}\} \\ k_{BT}^{-1}(M_\circ) = \emptyset \\ k_{AB}^{-1}(M_\circ) = \{x\} \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} P_* \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} Channels \end{array} \right\}$,

     – $\beta_{A^1T^1B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = k_{AB} \\ k_{AT}^{-1}(M_\circ) = \{k_{AB}\} \\ k_{BT}^{-1}(M_\circ) = k_{AB}^{-1}(M_\circ) = \emptyset \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} P_* \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} Channels \end{array} \right\}$,

     – $\beta_{A^2T^1B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = k_{AB} \\ k_{AT}^{-1}(M_\circ) = \{k_{AB}\} \\ k_{BT}^{-1}(M_\circ) = \emptyset \\ k_{AB}^{-1}(M_\circ) = \{x\} \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} P_* \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} Channels \end{array} \right\}$,

     – $\beta_{A^1T^2B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = k_{AB} \\ k_{AT}^{-1}(M_\circ) = \{k_{AB}\} \\ k_{BT}^{-1}(M_\circ) = \{u\} \\ k_{AB}^{-1}(M_\circ) = \emptyset \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} P_* \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} Channels \end{array} \right\}$,

     – $\beta_{A^2T^2B^0} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = k_{AB} \\ k_{AT}^{-1}(M_\circ) = \{k_{AB}\} \\ k_{BT}^{-1}(M_\circ) = \{u\} \\ k_{AB}^{-1}(M_\circ) = \{x\} \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} P_* \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} Channels \end{array} \right\}$,

$$- \ \beta_{A^1T^2B^1} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = k_{AB} \\ v = u \\ k_{AT}^{-1}(M_\circ) = \{k_{AB}\} \\ k_{BT}^{-1}(M_\circ) = \{u\} \\ k_{AB}^{-1}(M_\circ) = \emptyset \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} P_* \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} Channels \end{array} \right\},$$

$$- \ \beta_{A^2T^2B^1} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} u = k_{AB} \\ v = u \\ k_{AT}^{-1}(M_\circ) = \{k_{AB}\} \\ k_{BT}^{-1}(M_\circ) = \{u\} \\ k_{AB}^{-1}(M_\circ) = \{x\} \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} P_* \\ \{k_{AT}, k_{BT}, k_{AB}\} \perp_{\mathbf{K}} Channels \end{array} \right\},$$

$$- \ \beta_{A^2T^2B^2} \stackrel{\text{def}}{=} \{y = x\}.$$

2.8.3 Application of markings of transition graphs in the problems of verification of distributed processes

An **execution** of a DP P is a sequence $S_0, \ldots, S_n$ of states from Σ_P, such as $S_0 = \odot$, and either $n = 0$, or $\forall\, i = 0, \ldots, n-1 \quad S_i \to S_{i+1}$.

It is not difficult to see that each such sequence $S_0, \ldots, S_n$ corresponds to a path $G_P^0 = V^{S_0} \to \ldots \to V^{S_n}$ in TG G_P.

Some correctness properties of DPs have the following form: in each state S of an arbitrary execution of a DP P, the following implication holds:

$$S \models \beta \ \Rightarrow \ S \models \beta', \text{ where } \beta, \beta' \in Fm \text{ are given formulas.} \tag{37}$$

For example, for the DP P^*, where P is any of the DPs presented in section 2.5.2, one of the correctness properties has the following form: for arbitrary execution $S_0, \ldots, S_n$ of this DP,

- if
  - $S_n \models (v_B^{S_n} = B^1)$ (for first and second DPs in section 2.5.2), or
  - $S_n \models (v_B^{S_n} = B^2)$ (for third and fourth DPs in section 2.5.2),

  i.e. if B executed an action of receiving the message sent by A and wrote the received message in variable y,
- then $S_n \models (x = y)$, i.e. the received message is the same as the message x that A sent B.

Properties of the form (37) can be verified using a marking of TG G_P of an analyzed DP P as follows:

- a correct marking $(G, \{\beta_V \in Fm \mid V \in G\})$ of G_P is being built, and
- for each node $V \in G$, such that β_V implies β, the implication $\beta_V \Rightarrow \beta'$ is being checked.

To check the above statements, there is no need to fully build the TG G_P of the analyzed DP P. It is convenient to build the TG together with the construction of its marking as follows: if a formula β_V is built to mark the node V of G_P, and β_V implies an unrealizability of some edge outgoing from V, then this edge is discarded. For example, this can happen if

- a label of an edge outgoing from V is of the form $(c?\hat{y})_B$, and
- β_V contains the conjunctive term $M_c = \emptyset$.

As a result of such a construction with discarding unrealizable edges, a fragment of the TG G_P will be obtained. We shall call such fragment a **reduced TG**.

It is not difficult to see that the reduced TG preserves all the properties of the TG G_P. In particular, for the solution of the verification problem described above for a property of the form (37), the corresponding reduced TG can be used instead of the TG G_P.

2.8.4 Reduction of transition graphs

1. The edge $A^0B^0 \overset{(\bar{c}_{AB}?\hat{y})_B}{\longrightarrow} A^0B^1$ in TG (12) is unrealizable.

   The reduced TG (12) has the form

$$A^0B^0 \xrightarrow{(\bar{c}_{AB}!x)_A} A^1B^0 \xrightarrow{(\bar{c}_{AB}?\hat{y})_B} A^1B^1 \cdots \tag{38}$$

2. The edge $A^0B^0 \overset{?\bar{k}_{AB}(\hat{y})}{\longrightarrow} A^0B^1$ in TG (13) is unrealizable.

   The reduced TG (13) has the form

$$A^0B^0 \xrightarrow{!\bar{k}_{AB}(x)} A^1B^0 \xrightarrow{?\bar{k}_{AB}(\hat{y})} A^1B^1 \cdots \tag{39}$$

3. The following edges in TG (14) (where actions $\alpha_i, \beta_i, \gamma_i$ $(i = 1, 2)$ are

defined according to (8) or according to (10)) are unrealizable:

$$
\begin{aligned}
A^0T^0B^0 &\xrightarrow{\beta_1} A^0T^0B^1 \\
A^0T^0B^0 &\xrightarrow{\gamma_1} A^0T^1B^0 \\
A^1T^0B^0 &\xrightarrow{\beta_1} A^1T^0B^1 \\
A^1T^1B^0 &\xrightarrow{\beta_1} A^1T^1B^1 \\
A^1T^2B^1 &\xrightarrow{\beta_2} A^1T^2B^2 \\
A^2T^0B^0 &\xrightarrow{\beta_1} A^2T^0B^1 \\
A^2T^1B^0 &\xrightarrow{\beta_1} A^2T^1B^1
\end{aligned}
$$

The reduced TG (14) has the form

$$
\begin{tikzcd}
A^0T^0B^0 \arrow[r, "\alpha_1"] & A^1T^0B^0 \arrow[r, "\alpha_2"] \arrow[d, "\gamma_1"'] & A^2T^0B^0 \arrow[d, "\gamma_1"] & \\
& A^1T^1B^0 \arrow[r, "\alpha_2"] \arrow[d, "\gamma_2"'] & A^2T^1B^0 \arrow[d, "\gamma_2"] & \\
& A^1T^2B^0 \arrow[r, "\alpha_2"] \arrow[d, "\beta_1"'] & A^2T^2B^0 \arrow[d, "\beta_1"] & \\
& A^1T^2B^1 \arrow[r, "\alpha_2"] & A^2T^2B^1 \arrow[r, "\beta_2"] & A^2T^2B^2 \ \cdots
\end{tikzcd}
$$

Note that in all reduced TGs there is a single node S such that

- $S \models (v_B^S = B^1)$ (for first and second DPs in section 2.5.2), and
- $S \models (v_B^S = B^2)$ (for third and fourth DPs in section 2.5.2).

There are correct markings of these reduced TGs presented in section 2.8.2 such that $\beta_{V^S} = (x = y)$. As stated above, this statement is a justification of the property $S \models (x = y)$

Thus, by building a suitable marking, we verified the following property of all four considered DPs: if B executed the action of receiving the message sent by A and wrote the received message in the variable y, then the received message is the same as the message that A sent B.

3 An example of a cryptographic protocol verification

3.1 Description of a cryptographic protocoll

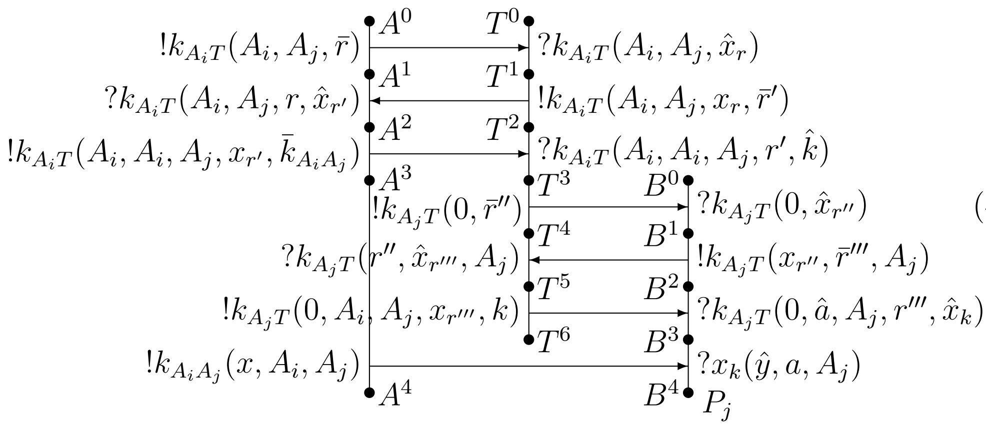
In this section we consider an example of a cryptographic protocol for transmitting encrypted messages between multiple agents through the open channel $\circ$. The participants of this protocol are

- agents from the set $\mathbf{A} = \{A_1, \ldots, A_n\}$, and
- a trusted intermediary T, with use of which agents from the set $\mathbf{A}$ send messages to each other.

Each agent $A_i \in \mathbf{A}$ uses the key k_{A_iT} to communicate with T, which is known only to agent A_i and T. A session of a transmission of an encrypted message x from agent $A_i \in \mathbf{A}$ to agent $A_j \in \mathbf{A}$ is a modification of the Wide Mouth Frog protocol. This session is denoted by the notation $A_i \stackrel{x}{\to} A_j$, and is consisting of the following actions:

- an exchange messages between A_i and T, resulting in T finds out
  - the sender's name A_i, the recipient's name A_j, and
  - the key $k_{A_iA_j}$, on which the message x from A_i to A_j will be encrypted,
- an exchange messages between T and A_j, resulting in A_j finds out
  - the sender's name A_i of the message that A_j will receive from A_i,
  - the key $k_{A_iA_j}$ on which this message will be encrypted,
- sending the encrypted message $k_{A_iA_j}(x, \ldots)$ from A_i to A_j.

This session is represented by the following scheme:



(40)

We denote

- by the notations A_{ij},T_{ij} and B_j the SPs corresponding to the left, middle and right threads of this diagram, these SPs describe the work of the sender (A_i), a trusted intermediary (T) and the recipient (A_j) respectively in this session, and
- by the symbol T the SP $\sum_{i,j=1}^{n} T_{ij}$, which denotes the work of a trusted intermediary in an arbitrary session of this protocol.

Let a finite set of sessions be given:

$$A_{i_1} \stackrel{x_1}{\to} A_{j_1},\ \ldots,\ A_{i_m} \stackrel{x_m}{\to} A_{j_m}. \tag{41}$$

One of cryptographic protocols designed to implement this set of sessions is represented by a DP

$$P = (A_{i_1 j_1}(x_1/x), \ldots, A_{i_m j_m}(x_m/x), T^\infty, B_1^\infty, \ldots, B_n^\infty) \tag{42}$$

This DP consists of SPs of the following families: A, T, B_1, …, B_n.

$\forall\, i \geq 1$ we denote those variables of the i–th copy of the SP B_j in B_j^∞, which are obtained by renaming the corresponding variables of B_j, by $x^{(i)}$, where x is the corresponding variable of B_j.

A property of this protocol that must be verified is the following:

$$\begin{array}{l} \forall\, S \in \Sigma_P,\ \forall\, j = 1, \ldots, n,\ \forall\, i \geq 1, \text{ if } S \models (v_{B_j}^{(i)} = B^4), \\ \text{then } M_\circ^S \text{ has a pair of messages of the form} \\ \bar{k}_{A_i A_j}(x, A_i, A_j, r) \text{ and } \bar{k}_{A_i T}(A_i, A_j, r) \end{array} \tag{43}$$

which means the following: a session from (41) of the form $A_i \stackrel{x}{\to} A_j$ was executed correctly.

3.2 Verification of the protocol

Let $S \in \Sigma_P$, where P is a DP of the form (42).

Using theorem 3 from section 2.7.2, it is not so difficult to prove that

$$\forall\, i, j = 1, \ldots, n \quad \{k_{A_i T}, k_{A_i A_j}\} \perp_{\mathbf{K}} P_*, \quad \{k_{A_i T}, k_{A_i A_j}\} \perp_{\mathbf{K}} M_\circ^S \tag{44}$$

Let $\tilde{M}_\circ^S$ be the set of messages in $M_\circ^S$ of the form $k_{A_i T}(\ldots)$ and $k_{A_i A_j}(\ldots)$. Using (40) and (44), it is not so difficult to prove that every message in $\tilde{M}_\circ^S$ has one of the following seven forms:

$$k_{A_i T}(A_i, A_j, r), \tag{45}$$

$$k_{A_iT}(A_i, A_j, r, r'), \tag{46}$$
$$k_{A_iT}(A_i, A_i, A_j, r, k_{A_iA_j}), \tag{47}$$
$$k_{A_jT}(0, r), \tag{48}$$
$$k_{A_jT}(r, r', A_j), \tag{49}$$
$$k_{A_jT}(0, A_i, A_j, r, k), \tag{50}$$
$$k_{A_iA_j}(x, A_i, A_j, r). \tag{51}$$

Let

- $\tilde{M}^S_{45}$, ..., $\tilde{M}^S_{51}$ be subsets of $\tilde{M}^S_\circ$, consisting of messages of the form (45), ..., (51) respectively,
- $\rho_{45,46}$ be a set of pairs of the form $((45), (46))$, in each of which the third component (r) listed in (45) is the same as the third component (r) listed in (46),
- $\rho_{46,47}$, $\rho_{48,49}$, $\rho_{49,50}$, be similar sets of pairs of the form $((46), (47))$, $((48), (49))$, $((49), (50))$.

Define a binary relation ρ on $\tilde{M}^S_\circ$ as the least transitive relation containing $\rho_{45,46}$, $\rho_{46,47}$, $\rho_{48,49}$, $\rho_{49,50}$, and satisfying the following conditions:

- if ρ contains pairs of the form
$$((45), (47)) \quad \text{and} \quad ((48), (50)) \tag{52}$$
and the last component in message (47) of the first pair is the same as the last component in message (50) of the second pair, then ρ contains the pair ((47), (48)) whose components are the corresponding messages from (52), and
- ρ contains each pair of the form ((50), (51)), in which the keys k and $k_{A_iA_j}$ are equal.

Below the notations $\exists_1$ and $\exists_{\leq 1}$ are read as “there is only one” and “there is at most one”, respectively.

With use of theorem 3, it is not so difficult to prove that

$$\left\{ \begin{array}{l} \forall\, e \in \tilde{M}^S_{47}\ \exists_1\, e' \in \tilde{M}^S_{45} : (e', e) \in \rho, \\ \forall\, e \in \tilde{M}^S_{50}\ \exists_1\, e' \in \tilde{M}^S_{48} : (e', e) \in \rho, \\ \forall\, e \in \tilde{M}^S_{50}\ \exists_1\, e' \in \tilde{M}^S_{45} : (e', e) \in \rho, \\ \forall\, e \in \tilde{M}^S_{51}\ \exists_{\leq 1}\, e' \in \tilde{M}^S_{45} : (e', e) \in \rho. \end{array} \right. \tag{53}$$

(53) and theorem 3 imply the following statement $\forall\, S \in \Sigma_P$, $\forall\, i \geq 1$, if $S \models (v^{(i)}_{B_j} = B^4)$, then $M^S_\circ$ contains a pair of messages of the form (43), i.e. the integrity property of the analyzed protocol is true: if agent A_j performed the action of receiving a message sent by agent A_i and wrote the received message to variable y_{B_j}, then the received message is the same as the message x that A_i sent A_j in the same session.

4 Conclusion

In the present work, a new model of cryptographic protocols was built, and examples of its use for solving problems of verification of protocol integrity properties are shown.

The objectives for further development of this model and verification methods based on it are the following:

1. an automation of synthesis of suitable markings in transition graphs of the analyzed protocols,
2. development of the language of specification of properties of cryptographic protocols, which allow to express e.g.
   - properties of confidentiality (secrecy) of transmitted messages, i.e. the adversary's inability to extract any new information about the content of messages intercepted by him,
   - matching properties in authentication protocols, or zero knowledge properties,
   - non-traceability properties in electronic payments,
   - properties of correctness of the votes' counting in voting protocols,
3. construction of automated synthesis methods of cryptographic protocols by describing the properties which the cryptographic protocols must satisfy, etc.

References

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

[2] Needham R., Schroeder M., Using Encryption for Authentication in large networks of computers, Communications of the ACM, 21(12), (1978) 993-999.

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

[4] Cervesato I., Jaggard A.D., Scedrov A., Tsay J.-K., Walstad C., Breaking and fixing public-key Kerberos, Information and Computation Volume 206, Issues 2-4, (2008), Pages 402-424.

[5] Lowe G., Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR, In Proceedings of TACAS, (1996) 147-166, Springer Verlag.

[6] Kerberos: The Network Authentication Protocol. MIT Kerberos. 10 September 2015. Retrieved 31 October 2015.
`http://web.mit.edu/kerberos/`

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

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

[9] J. D. Guttman and F. J. Thayer. Authentication Tests and the Structure of Bundles. Theoretical Computer Science, June, 2002.

[10] Joshua D. Guttman. State and Progress in Strand Spaces: Proving Fair Exchange. Journal of Automated Reasoning, 48(2): 159–195, 2012.

[11] M. Abadi. Security Protocols and Their Properties. In NATO Science Series: Volume for the 20th International Summer School on Foundations of Secure Computation, pp. 39-60, Marktoberdorf, Germany, 1999.

[12] M. Abadi and B. Blanchet. Secrecy Types for Asymmetric Communication. In Conference on Foundations of Software Science and Computation Structures (FOSSACS), LNCS 2030, pp. 25-41, 2001.

[13] M. Abadi and R. Needham. Prudent Engineering Practice for Cryptographic Protocols. In IEEE Transactions on Software Engineering, 22(1):6-15, 1996.

[14] M. Abadi and B. Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. In Journal of the ACM, 52(1), pp. 102-146, 2005.

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

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

[17] J. Zhou, D. Gollmann A Fair Non-repudiation Protocol. In IEEE Symposium on Research on Security and Privacy, pp. 55-61, 1996.

[18] M. Abadi, N. Glew, B. Horne, B. Pinkas Certified E-mail with a Light On-line Trusted Third Party: Design and Implementation. In 11th Int. World Wid Web Conference, pp. 387-396, 2002.

[19] M. Abadi, B. Blanchet Computer-assisted Verification for Certified E-mail. In Science of Computer Programming, 58(1-2):3-27, 2005.

[20] M. Abadi. Secrecy by Typing in Security Protocols. In Journal of the ACM, 46(5), pp. 749-786, 1999.

[21] S. Kremer, M. Ryan. Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In 14th European Symposium on Programming (ESOP), pp. 186-200, 2005.

[22] B. Blanchet. Automatic Proof of Strong Secrecy for Security Protocols. In IEEE Symposium on Security and Privacy, pp. 86-100, 2004.

[23] B. Blanchet, M. Abadi, C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In 20th IEEE Symposium on Logic in Computer Science (LICS), pp. 331-340, 2005.

[24] 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), 2000.

[25] W. Aiello, S. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. Keromytis, O. Reingold. Just Fast Keying: Key Agreement in a Hostile Internet. In ACM Transactions on Information and System Security, 7(2):242-273, 2004.

[26] M. Abadi, B. Blanchet, C. Fournet. Just Fast Keying in the Pi Calculus. In ACM Transactions on Information and System Security, 10(3), 2007.

[27] A. Gordon and A. Jeffrey. Authenticity Authenticity by Typing for Security Protocols. In Journal of Computer Security, 11(4), pp. 451-521, 2003.

[28] Duncan, Richard. An Overview of Different Authentication Methods and Protocols. SANS Institute. Retrieved 31 October 2015

[29] Proceedings of Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA '06) Information and Computation Volume 206, Issue 2, (2008).

[30] Veronique Cortier, Steve Kremer. Formal Models and Techniques for Analyzing Security Protocols. Now Publishers Inc., Hanover, United States (2014).

[31] Syverson P., van Oorschot P.C., On Unifying some Cryptographic Protocol Logics, Proceedings of the 1994 IEEE Computer Security Foundations Workshop VII, (1994) 14-29, IEEE Computer Society Press.

[32] 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.

[33] Paulson L., Proving Properties of Security Protocols by Induction, Proceedings of the IEEE Computer Security Foundations Workshop X, (1997) 70-83, IEEE Computer Society Press.

[34] Brackin S., A State-Based HOL Theory of Protocol Failure, (1997), ATR 98007, Arca Systems, Inc., http://www.arca.com/paper.htm.

[35] Mark D. Ryan and Ben Smyth, Applied pi calculus, in: Formal Models and Techniques for Analyzing Security Protocols, Edited by Veronique Cortier, 2011 IOS Press, p. 112-142.

[36] Abadi M., Gordon A., A Calculus for Cryptographic Protocols: The Spi Calculus, Proceedings of the Fourth ACM Conference on Computers and Communications Security, (1997) 36-47, ACM Press.

[37] 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