JAM CM Journal of Applied Mathematics and Computational Mechanics 2015, 14(3), 127-139
www.amcm.pcz.pl p-ISSN 2299-9965
DOI: 10.17512/jamcm.2015.3.14 e-ISSN 2353-0588

TIMED MODELS OF SECURITY PROTOCOLS INCLUDING DELAYS IN THE NETWORK

Sabina Szymoniak[1], Mirosław Kurkowski[2], Jacek Piątkowski[1]

[1]Institute of Computer and Information Sciences, Czestochowa University of Technology
Częstochowa, Poland
[2]Institute of Computer Science, Cardinal Stefan Wyszynski University
Warszawa, Poland
sabina.szymoniak@icis.pcz.pl, m.kurkowski@uksw.edu.pl, jacek.piatkowski@icis.pcz.pl

Abstract. A very important part of the network and computer systems is to ensure an appropriate level of information transmission security. Security is based primarily on properly selected communication in security protocol. Unfortunately, the time-dependent protocols are vulnerable to attacks; therefore there is a need to verify these protocols. For verification purposes, protocols can be modeled using timed automata. During the modeling and verification of the protocols, one should also keep in mind delays in the network. So far, delays in the network were not modeled.

Keywords: *computer science, mathematical models*

1. Introduction

Assurance of an appropriate level of information transmission security is a very important part of the network and computer systems. Security is based primarily on properly selected communication in security protocol. Commonly are used encryption and random numbers (*nonce*), which are intended to identify positively consecutive communication sessions. However, when the protocol continues after the break in time, we cannot assume that the key used to encrypt the message has already been broken. There is a possibility to make the so-called re-play attack in which the Intruder uses the old ciphertext. In order to confirm the time to send a message, timestamps are used. This allows to reject these communication sessions which have a very long duration and thus avoiding attacks [1].

One of the protocols that uses the timestamps is the *symmetric Wide Mouth Frog protocol*. This protocol aims to establish a new session key. Session keys are used to encrypt information during a single session. The Wide Mouth Frog protocol has a very simple structure. It occurs in two users (labeled as A and B) and

so-called *trusted server* (labeled as S). The server mediates the exchange of the session key between A and B, and in the mutual authentication of users. Wide Mouth Frog design consists of two steps. In the first step, A sends a request to the server to communicate with B with the corresponding timestamp. The server after verification of the user sends an appropriate message to B, which will help him connect with A. Of course, each message is encrypted with symmetric key shared between the user sending the message and the server [2].

It is worth noting that the very design of correct protocols is a very difficult task. The given protocol may be susceptible to even the simplest attacks. Therefore there is a need to verify the modeling operations and security protocols. In modeling, the user's knowledge about keys or nonces that the user acquires during the protocols execution plays a very important role. Formal verification of protocols uses the formal description of the protocol in the specification language. However, during the verification of time-dependent protocols, one should take into consideration the appropriate conditions of the generation of the new time items or run-time protocol.

The main methods of verification are: *simulations*, and *the formal modeling and verification*. The simulations are based on actual testing systems or the simulations performed on the operation of these systems using virtual machines. The second method involves the construction of special mathematical structures that describe the executions of all actions in the study protocol defined space [1].

The formal methods can be replaced with the *inductive methods* [3], *deductive methods* [4] and *model checking* [5]. The first method uses mathematical induction to prove the fulfillment of certain properties by the protocol. The deductive method requires constructing a deductive system (logic) However, formal mathematical models of appropriate transitions systems are created into the model checking of the study of the protocol executions [6]. So far, there were several established tools for security protocols verification. The proper models are searched by specialized tools like *TPMC* [7], *AVISPA* [8] or *Scyther* [9].

The formal mathematical models can be built using the so-called timed automata. Then, the time needed both to generate and send the whole message of that news as well as network delays can be modeled. These delays are related to the transition time of a message between the sender and the recipient (the network links [6]). So far, during the modeling and verification of security protocols, delays in the network were not modeled. This problem is mainly very important from a practical point of view. People forming network infrastructure need such verification report to check for proper operation and susceptibility to attack before the launch.

The paper will present a new approach to the problem of modeling of executions security protocols in relation to the method described in papers [10, 11]. In our approach, interlaces of protocols executions will be considered.

2. Computing structure

To define a model of time-dependent security protocol including delays in the network, corresponding structures, in which an Intruder will also appear, is necessary. Due to their presence, it is possible to consider the real executions of protocol in a real network. Definition sets of the basic structures objects are as follows:

Definition 2.1. Let:

- $\mathbf{P} = \{\mathrm{p}_1, , \mathrm{p}_2, \dots, \mathrm{p}_{\mathrm{n_p}}\}$ *is a set of honest users in computer network,*
- $\mathbf{P}_\iota = \{\iota, \iota(p_1), \iota(p_2), \dots, \iota\left(p_{n_p}\right)$ *is a set representing Intruder* ι *and Intruder who impersonates honest users* $p_i \; for \; 1 \leq i \leq n_p$,
- $\mathbf{I} = \{i_{p_1}, \dots, i_{p_{n_p}}, i_\iota\}$ *be set of network user identifiers,*
- $\mathbf{K} = \bigcup_{i=1}^{n_P}\{k_{p_i}, k_{p_i}^{-1}\} \cup (\bigcup_{i=1}^{n_P}\bigcup_{j=1}^{n_P}\left\{k_{p_i p_j}\right\} \setminus \bigcup_{i=1}^{n_P}\{k_{p_i p_i}\})$ *is a set of users cryptographic keys (symmetric and asymmetric),*
- $\mathbf{N} = \bigcup_{i=1}^{n_P}\{n_{p_i}^1, \dots n_{p_i}^{k_N}\} \cup \{n_{p_\iota}^1, \dots n_{p_\iota}^{k_N}\}$ *is a set of pseudo-random numbers (nonces),*
- $\mathbf{T} = \bigcup_{i=1}^{n_P}\{t_{p_i}^1, \dots t_{p_i}^{n_T}\} \cup \{t_{p_\iota}^1, \dots t_{p_\iota}^{n_T}\}$ *is a set of users timestamps,*
- $\mathbf{\Gamma} = \{l_1, l_2, \dots, l_{n_L}\} \subseteq R_+$ *is a set of positive real numbers representing the tickets' validity periods of time - lifetimes,*
- $\mathbf{\Delta} = \{\delta_1, \delta_2, \dots, \delta_{n_D}\} \subseteq R_*$ *is a set of non-negative real numbers representing delays in the network.*

Numbers n_P, k_N are fixed natural numbers. To simplify the discussion, the assumption was accepted that each participant of the protocol may generate the same number of nonces. Algebra of terms, which allows one to create definitions messages with the relationship between them, is as follows.

Definition 2.2. *The set of the letters* **L** *let be the smallest set satisfying the following conditions:*

- $\mathbf{P} \cup \mathbf{P}_\iota \cup \mathbf{I} \cup \mathbf{K} \cup \mathbf{N} \cup \mathbf{T} \cup \mathbf{\Gamma} \cup \Delta \subseteq \mathbf{L}$,
- *If* $x, y \in \mathbf{L}$, *then concatenation* $x \cdot y \in \mathbf{L}$,
- *If* $x \in \mathbf{L}$ *and* $k \in \mathbf{K}$, *then* $\langle x \rangle_k \in \mathbf{L}$, *where* $\langle x \rangle_k$ *is cryptogram which contains letter x and have been encrypted by key k,*
- *If* $x \in \mathbf{L}$, *then* $h(x) \in \mathbf{L}$, *where* $h(x)$ *is the value of the hash function on the letter x.*

The auxiliary relationship on the set of **L** is as follows.

Definition 2.3. *Relation of direct subletter* $\prec\cdot \subseteq \mathbf{L} \times \mathbf{L}$ *let be the smallest relation satisfying the following conditions:*

1. *If* $x, y \in \mathbf{L}$, *then* $x \prec\cdot \; x \cdot y$ *and* $y \prec\cdot \; x \cdot y$,
2. *If* $x \in \mathbf{L}$ *and* $k \in \mathbf{K}$, *then* $x \prec\cdot \langle x \rangle_k$, *and* $k \prec\cdot \langle x \rangle_k$,
3. *If* $x \in \mathbf{L}$, *then* $x \prec\cdot \; h(x)$.

$\preccurlyeq$ means marked feedback and passers closure of $\prec$·relation. For any set of letters X $\subseteq$ **L**, the sequence of sets $(X^n)_{n\in N}$ will be defined, as well as subsets of **L**, where:

- $X^0 \stackrel{\text{def}}{=} X$,
- $X^{n+1} \stackrel{\text{def}}{=} X^n \cup \{z \in \mathbf{L} \mid (\exists x, y \in X^n, k \in X \cap \mathbf{K})\ z = x \cdot y \vee z = \langle x \rangle_k \vee z = h(x)\}$.

A set X^{n+1} contains letters built inductively because of encryption, concatenation and hash functions that may arise from X^n.

Then set $Comp(X) \stackrel{\text{def}}{=} \bigcup_{n\in N} X^n$ is a set of letters which can be created from the letter of set *X*. The set of all subletters from *X l*etters of the subletter relationship is as follows:

$$Sublet(X) \stackrel{\text{def}}{=} \{l \in \mathbf{L} \mid (\exists x \in X)\ l \preccurlyeq x\}.$$

In further considerations it is necessary to introduce an Intruder which, depending on their powers, will want to deceive others, fair participants, by sending the letters which often will not adhere to the idea of the protocol. The Intruder can compose letters depending on the knowledge. The expression of the Intruder's acquired knowledge depending on already possessed knowledge and information captured from the network is as follows.

Definition 2.4. *Let* $X \subseteq \mathbf{L}$ *and* $K \subseteq \mathbf{K}$. *The set of* $\xi_K(X) \subseteq \mathbf{L}$ *let be the smallest set that satisfies the following conditions:*

1. $X \subseteq \xi_K(X)$,
2. *If* $l \cdot m \in \xi_K(X)$, *then* $l \in \xi_K(X)$ *and* $m \in \xi_K(X)$,
3. *If* $\langle l \rangle_k \in \xi_K(X)$ *and* $k \in \xi_K(X) \cup K$, *then* $l \in \xi_K(X)$.

Set of $\xi_K(X)$ contains all the letters which can be obtained from the set of letters *X* by concatenations decomposition and decryption with keys belonging to the set $\xi_K(X)$ or *K*. For further considerations, by a set of $\xi_K(X)$ set of $\xi_\emptyset(X)$ will be denoted.

For the purposes of this article, the protocol will be treated as an algorithm which is an abstract concept. In order to consider the various executions of one protocol and their floats, it is necessary to define formal executions. An execution of the protocol, at a given time, is determined by using parameters therein, such as users, keys, nonces, timestamps but also the time associated with delays occurring in the networks.

For the definition of the protocols executions, two types of functions mapping a set of terms ***T*** in a set of letters **L** are necessary. Two types of functions are: *interpretations* and *partial interpretations*.

Definition 2.5. *Partial interpretation of a set of terms of letters* ***T*** *let be any injective function* $\overline{f}$: ***T*** $\rightarrow$ **L** *satisfying the following conditions:*

1. $\overline{f}(T_P) \subseteq \boldsymbol{P} \cup \mathbf{P}_\iota, \overline{f}(T_I) \subseteq \mathbf{I}, \overline{f}(T_K) \subseteq \mathbf{K}, \overline{f}(T_N) \subseteq \mathbf{N}, \overline{f}(T_L) \subseteq \boldsymbol{\Gamma}, \overline{f}(T_T) \subseteq \mathbf{T}, \overline{f}(T_D) \subseteq \boldsymbol{\Delta}$,

2. $(\forall X, Y \in T)\ \overline{f}(X \cdot Y) = \overline{f}(X) \cdot \overline{f}(Y)$ *(homomorphism),*
3. $(\forall X \in T)\ (\forall k \in T_K)\overline{f}(\langle X\rangle_K) = \langle\overline{f}(X)\rangle_{\overline{f}(k)}$ *(homomorphism),*
4. $(\forall X \in T)\ \overline{f}(h(X)) = h(\overline{f}(X))$ *(homomorphism),*
5. *If* $\overline{f}(P) = p\ for\ p \in \mathbf{P}$, then $\overline{f}(I_P) = i_p, \overline{f}(N_P) \in \{n_p^1, \dots, n_p^{1k_N}\}, \overline{f}(K_P) = k_p, \overline{f}(K_P^{-1}) = K_P^{-1}\ and\ \overline{f}(\tau_P) \in \{t_p^1, \dots, t_p^{n_T}\}$,
6. *If* $\overline{f}(P) = \iota$, then $\overline{f}(I_P) = i_\iota, \overline{f}(K_P) = k_\iota, \overline{f}(K_P^{-1}) = K_\iota^{-1}$,
7. *If* $\overline{f}(P) = \iota(p)$, then $\overline{f}(I_P) = i_\iota, \overline{f}(K_P) = k_\iota, \overline{f}(K_P^{-1}) = K_\iota^{-1}$,
8. $\overline{f}(T_P)\backslash\mathbf{P}_\iota \neq \emptyset$.

Partial interpretations specify all the parameters of the executions of the protocol apart from the aspects of time. Partial interpretation therefore determines the set of protocols executions which differ in execution time, but are realized with the same parameters at different times. Whereas, the interpretation designates a specific execution of the protocol.

Definition 2.6. *For a fixed partial interpretation of the* $\overline{f}$ *interpretation induced by* $\overline{f}$*, a set of terms of letter T and a set of real variables* T_R *will be referred to any injective function* $f: T \cup T_R \to \mathbf{L} \cup R_*$ *,so* $f|_{T\backslash T_T} = \overline{f}$ *and* $f(T_T \cup T_R) \subseteq R_*$

From the above definition, the timestamps and moments of sending messages are finally mapped into positive real numbers.

Honest users execute the protocol according to the schema if it is properly constructed. However, the Intruder may try to transmit wrong, sometimes intercepted or modified data. Depending on the chosen model of the Intruder, the data can be modified in various ways, and the Intruder can construct them with previously acquired knowledge. In steps, in which the Intruder is a sending party, it is possible to compose the Intruder's letters with a given set of knowledge.

Definition 2.7. *Let* $l \in \mathbf{L}$ *be a letter and* $X \subseteq \mathbf{L}$ *be set of letters. Set X will be referred further as a set of letters l generators (which will be determined by* $X \vdash l$*) if the following conditions are satisfying:*

1. $X \subseteq Sublet(\{l\})$,
2. $l \in Comp(X)$,
3. $(\forall m \in X)(m \notin Comp(X\{m\}))$,
4. $(\forall m \in X)(l \notin Comp(X\{m\}))$.

A set *X* is a set of letter *l* generators ($X \vdash l$) if all elements of *X* are subletters of *l*, *l* may be composed of elements of the set *X*, and *X* is the minimal set satisfying this property.

Temporal conditions are defined in set *C*. Induction definition of their interpretation is as follows:

- $f(true) := true$,
- $f(\tau_i + \delta - \tau_j \leq L_f) := f(\tau_i) + f(\delta) - f(L_f) \leq f(L_f)$,
- $f(\mathbf{tc}_1 \wedge \mathbf{tc}_2) := f(\mathbf{tc}_1) \wedge f(\mathbf{tc}_2)$.

A defined set of letters generators and interpretations on the set *T* can be used to define a step in time-dependent protocol and also the whole time-dependent protocol.

Definition 2.8. *Consider the following step:*

$$\alpha = (\alpha^1, \alpha^2) = \big((P, Q, L), (\tau_\alpha, \delta_\alpha, X, G, \boldsymbol{tc})\big)$$

of the given time-dependent protocol Σ and interpretation f on the set $T \cup T_R$ *which satisfies the following* $\bigwedge_{\tau \in G \cap T_T}(f(\tau_\alpha) = f(\tau))$. *This condition indicates that the period of timestamp is identical to the time sending messages with that step protocol. By f-interpretation of step α (denoted by* $f(\alpha)$*) the following tuple will be denoted:*

- $\Big(\big(f(P), f(Q), f(L)\big), \big(f(\tau_\alpha), f(\delta_\alpha), f(X), f(G), f(\mathbf{tc})\big)\Big), if\ f(P) \in \mathbf{P}$,
- $\big((f(P), f(Q), f(L)), (f(\tau_\alpha), f(\delta_\alpha), \{X \vdash f(L)\}, \emptyset, f(\mathbf{tc}))\big),\ if\ f(P) \in \mathbf{P}_\iota$

In a situation where the Intruder is a sending party, there is the assumption that they can compose the letter *f(L)* of each set which generates the letter *f(L)*. It is also assumed that the Intruder previously may have generated its own set of nonces, keys, and timestamps. The above assumption is not a constraint considerations because the Intruder does not need to perform their activities, including generating confidential information in accordance with the idea of the protocol and they may use sensitive data repeatedly in various applications (sessions) of the protocol.

To determine the execution of the protocol and the knowledge of users, including the Intruder, the secondary concepts and notations must be introduced which will be helpful in defining and recording further structure and dependencies.

Let $f(\alpha_i) = \big((p, q, l), (t, d, X, G, \mathbf{tc})\big)$ for some $p, q \in \mathbf{P} \cup \mathbf{P}_\iota, \boldsymbol{X} \in 2^L_{fin}, G \in \in 2^{\mathbf{K} \cup \mathbf{N}}_{fin}, and\ l \in \mathbf{L}$ be:

Let:

- $Send^{f(\alpha_i)} = p$ - sender in step $f(\alpha_i)$,
- $Lett^{f(\alpha_i)} = l$ - letter in step $f(\alpha_i)$,
- $Gen^{f(\alpha_i)} = G$ - a set of new confidential data generated in step $f(\alpha_i)$,
- $Resp^{f(\alpha_i)} = q$ - recipient of letter in $f(\alpha_i)$,
- $Part^{f(\alpha_i)} = \{Send^{f(\alpha_i)}, Resp^{f(\alpha_i)}\}$,
- $Time^{f(\alpha_i)} = t$,
- $Delay^{f(\alpha_i)} = d$ - delay in the network in step $f(\alpha_i)$,
- $TConstr^{f(\alpha_i)} = \mathbf{tc}$ - time for step $f(\alpha_i)$.

If the sender is honest, that meaning: $Send^{f(\alpha_i)} \in \mathbf{P}$, then let $Comp^{f(\alpha_i)} = X$ be a set of letters, all of which compose the letter $Lett^{f(\alpha_i)}$. However, if the sender is an Intruder, that meaning: $Send^{f(\alpha_i)} \in \mathbf{P}_\iota$, then let $Comp^{f(\alpha_i)} = \bigcup X \subseteq L \mid X \vdash Lett^{f(\alpha_i)}\}$ be the sum of all the sets of letters generators $Lett^{f(\alpha_i)}$.

Similarly, for the partial interpretation $\overline{f}$ of interpretation *f*, a similar notation is used: $Send^{\overline{f}(\alpha_i)}, Lett^{\overline{f}(\alpha_i)}, Gen^{\overline{f}(\alpha_i)}, Resp^{\overline{f}(\alpha_i)}$ respectively

for $\overline{f}(P), \overline{f}(L), \overline{f}(G), \overline{f}(Q)$. These structures allow the formal definition of both the individual steps and of the whole protocol.

Definition 2.9. Let *f* be interpretation satisfying the following conditions:

- $f(\alpha_1), f(\alpha_2), \dots, f(\alpha_n)$ are *f*-interpretation of protocols steps,
- $\bigwedge_{i=1}^{n} TConst^{f(\alpha_i)} \equiv true$.

By *f*-execution of protocol Σ, a course $f(\Sigma) = f(\alpha_1), f(\alpha_2), \dots, f(\alpha_n)$ will be understood.

Example 2.1. *Consider the Wide Mouth Frog (WMF) protocol and interpretation f where* $f(A) = a, f(B) = b, f(S) = s, f(L_f) = l, and\ f(\tau_i) = t_i$. *From definition 2.5 of this interpretation, the results are as follows:* $f(I_A) = i_a, f(\tau_A) = t_a, f(\tau_S) = t_s, f(K_{AS}) = k_{as}$, $f(K_{AB}) = k_{ab}$ *and* $f(K_{BS}) = k_{bs}$. *The next steps of f-interpretation of WMF protocol are following:*

1. $f(\alpha_1) = \Big(\big(a, s, i_a, \langle t_a, i_b, k_{ab}\rangle_{k_{as}}\big), \big(t_1, d_1, \{i_a, i_b, t_a, k_{ab}, k_{as}\}, \{t_a, k_{ab,}\}, t_1 + + d_1 - t_a \le l\big)\Big)$,
2. $f(\alpha_2) = \Big(\big(s, b, \langle t_s, i_a, k_{ab}\rangle_{k_{bs}}\big), (t_2, d_2, \{i_a, t_s, k_{ab}, k_{as}\}, \{t_s\}, t_2 + d_2 - t_a \le l \wedge t_2 + + d_2 - t_s \le l)\Big)$.

Any interpretation corresponds to a specific execution protocol. Due to consideration of the different interlaces of the various executions of the examined protocol, it is necessary to introduce an ancillary structures that allow one to analyze the set of interpretation and also interlace of executions of steps which make up the specific executions. First of all, these concepts allow one to express the knowledge of users participating in the many interlaces of executions of the examined protocol.

For the considered set of interpretations *F* set $Comp_F^p$ ($Comp_F^\iota$) of letters will be defined which the user $p \in \bigcup_{f\in F} f(T_P)\backslash \mathrm{P}_\iota$ (Intruder ι) needs in order to create all the letters that they must create and send in all the steps of executions with interpretations from the set $f \in F$.

Definition 2.10. *For honest user p, it is set:* $Comp_F^p = \bigcup_{i=1}^{n} \bigcup\{Comp^{f(\alpha_i)} \mid f \in F \wedge Send^{f(\alpha_i)} = p\}$. *For Intruder* ι, *it is:* $Comp_F^\iota = \bigcup_{i=1}^{n} \bigcup\{Comp^{f(\alpha_i)} \mid f \in F \wedge Send^{f(\alpha_i)} \in \mathbf{P}_\iota\}$.

Let's consider any finite set of protocols interpretations consisting of k steps $\mathbf{r} = (f^1(\alpha_{i_1}), f^2(\alpha_{i_2}), \dots, f^k(\alpha_{i_k}),)$. For any $p \in \bigcup_{i=1}^{k} f^i(T_p)$ the sequence of the user's knowledge sets $\left(\kappa_p^j\right)_{j=1,\dots,k}$ will be defined in the following steps of protocols executions.

Definition 2.11. *For honest user* $p \in \bigcup_{f\in F} f(T_P)\backslash \mathbf{P}_\iota$ *,their knowledge in j-step is defined inductively:*

$$\kappa_p^0 = \mathbf{I} \cup \left\{k_p^{-1}\right\} \cup \left\{k_q \mid q \in \mathbf{P}\right\} \cup \{k_\iota\},$$

$$\kappa_p^{j+1} = \begin{cases} \kappa_p^j & if\ p \notin Part^{f^{j+1}(a_{i_{j+1}})} \\ \kappa_p^j \cup Gen^{f^{j+1}(a_{i_{j+1}})} & if\ p = Send^{f^{j+1}(a_{i_{j+1}})} \\ Comp_F^p \cap \xi_{\{k_p^{-1}\}}(\kappa_p^j \cup \{Lett^{f^{j+1}(a_{i_{j+1}})}\}) & if\ p = Send^{f^{j+1}(a_{i_{j+1}})}. \end{cases}$$

The user's knowledge, who does not participate in the protocol step, does not change. If the user is the initiator of step, their knowledge will be increased by the generated new set of confidential information. However, if the user is the recipient of the letter in the step, their knowledge will be increased by all the letters which can be isolated from the letter obtained in the step. In order to reduce the model, knowledge is limited only to the above mentioned set $Comp_F^p$ - which is the set of letters which the user needs to compose a list of all versions of a set of steps *F* when they are a sending party.

Example 2.2. *Let's consider f-execution of the WMF protocol given in Example 2.1 and following sequence of protocols* $\mathbf{r} = (f(\alpha_1), f(\alpha_2), f(\alpha_3))$*. The sequences of sets* $\left(\kappa_a^i\right)_{i=0,1,2,3}$ *and* $\left(\kappa_b^i\right)_{i=0,1,2,3}$ *are as follows:*

- $\kappa_a^0 = \{i_a, i_b, i_s, k_{as}\}$ *– initial knowledge of user a,*
- $\kappa_a^1 = \kappa_a^0 \cup \{t_a, k_{ab}\}$ *– knowledge after execution of the first step by user a (initial knowledge increased by timestamp and key* k_{ab} *which were generated during the first step by a),*
- $\kappa_a^2 = \kappa_a^1$,
- $\kappa_b^0 = \{i_a, i_b, i_s, k_{bs}\}$,
- $\kappa_b^1 = \kappa_b^0$,
- $\kappa_b^2 = \kappa_b^1 \cup \{t_s, k_{ab}\}$.

Intruder's knowledge depends on the considered model.

Definition 2.12. *Induction definition of Intruder's knowledge limited by Dolev-Yao model is as follows:*

$$\kappa_\iota^0 = \mathbf{I} \cup \{k_\iota^{-1}, k_\iota\} \cup \left\{k_q \mid q \in \mathbf{P}\right\} \cup \left\{n_\iota^1, \dots, n_\iota^{k_N}\right\} \cup \{t_\iota^1, \dots, t_\iota^{n_T}\},$$

$$\kappa_\iota^{j+1} = \begin{cases} \kappa_\iota^j & if\ Resp^{f^{j+1}(a_{i_{j+1}})} \notin \mathbf{P}_\iota \\ Comp_F^\iota \cap \xi_{\{k_\iota^{-1}\}}(\kappa_\iota^j \cup \{Lett^{f^{j+1}(a_{i_{j+1}})}\}) & if\ Resp^{f^{j+1}(a_{i_{j+1}})} \in \mathbf{P}_\iota. \end{cases}$$

If the intruder is not the recipient of steps executions, their knowledge will not change. On the other hand, when the Intruder is the recipient, they will expand

their knowledge by the original message and by all that can be taken from this letter at the current state of their knowledge (of course, in order to reduce the size of the model, this knowledge is limited to set $Comp_F^L$).

For simplicity, it is assumed that an Intruder cannot generate nonces according to the protocols idea. They have previously prepared the appropriate number of nonces that can be used in each protocols steps executed. The Intruder may also use the nonces many times in any step of the executions.

Conditions, guaranteeing that the sequence of step interpretation is the calculation of the protocol in the computing structure, are as:

Definition 2.13. *Calculation of the protocol Σ is a finite interpretation of the protocols steps: $\boldsymbol{r} = (f^1(\alpha_{i_1}), f^2(\alpha_{i_2}), \dots, f^k(\alpha_{i_k}),)$, which are the following conditions:*

1. $(\forall k \in \mathbf{N}_+)\left[i_k > 1 \Longrightarrow (\exists j < k)\left(f^j = f^k \wedge i_j = i_k - 1\right)\right]$,
2. $(\forall k, j \in \mathbf{N}_+)\left[k \neq j \Longrightarrow Gen^{f^k(\alpha_{i_k})} \cap Gen^{f^j(\alpha_{i_j})} = \emptyset\right]$,
3. $(\forall j \in \mathbf{N}_+)\left[Lett^{f^j(\alpha_{i_j})} \in Comp(\kappa^{j-1}_{Send^{f^j(\alpha_{i_j})}} \cup Gen^{f^j(\alpha_{i_j})})\right]$,
4. $\forall_{n=1,\dots k-1}(Time^{f^n(\alpha_{i_n})} < Time^{f^{n+1}(\alpha_{i_{n+1}})})$,
5. $\forall_{n=1,\dots k}\, TConstr^{f^n(\alpha_{i_n})} = true$.

According to the first condition, each step of the protocol (except for the first step with the partial interpretation $\overline{f}$ over the step before) is its predecessor at the same interpretation. The second condition decrees that nonces sets, generated by honest users, are disjoint.

The third condition guarantees that letter $Lett^{f^j(\alpha_{i_j})}$ (sent in each step) can be sent by the user $Send^{f^j(\alpha_{i_j})}$ only when the user has the appropriate knowledge to compose this letter. The fourth condition ensures appropriate relationship of time between successive steps during the executions. The last condition says that all the time conditions of each step and used interpretation must be satisfied.

3. Timed automata modeling executions of the protocol

For the full modeling of the security protocols, it is necessary to prepare a synchronized network of timed automata which must include two types of this automata. The first is *automata of executions* that will represent the next steps of protocols executions. The second type is *automata of knowledge* whose task is to present knowledge of the protocols participants. Automata of executions will be the timed automata since they have to take into account the time dependencies. In contrast, automata of knowledge do not contain time dependencies; however, due to the definition of timed automata, they can be considered as a product of timed automa-

ta network. With the proper synchronization between automata of executions and automata of knowledge, it will be possible to complete the modeling of protocols executions. Automata of knowledge will not be included in this work.

Let *C* be set of conditions using integers as *X*.

Definition 3.1. *Timed automaton (abbreviated TA) is following five* $\mathcal{A} = (A, L, l^0, E, \mathcal{X})$*, where:*

- A *is a finite set of labels, a* $A \cap R_* = \emptyset$*,*
- L *is a finite set of states,*
- $l^0 \in L$ *is an initial state,*
- $\mathcal{X}$ *is a finite set of clocks,*
- $E \subseteq L \times C \times 2^X \times L$ *is a relation of transition.*

Each element e from set E which represents the transition from the location l to location l' will be denoted by $l \xrightarrow{a,\mathbf{cc},X} l'$*. In this assay, a is executed set;* $X \subseteq \mathcal{X}$ *is a set of reset clocks (zeroed) during executing of action a;* $\mathbf{cc} \in C$ *are time conditions that have been imposed on executing of action a and transition of e.*

The clocks allow one to express the time dependencies between successive steps of executions and timestamps validity periods.

Along with the relation of transition, the so-called transitions system is related to; which is responsible for executing actions (transitions between states) with a time parameters. In this system, the transition is defined as the action and time successors. The time successors only change the valuation of clocks (the passage of time). On the other hand, actions successors, which are associated with the execution of action, may be taken when the time conditions **cc** are satisfied at an appropriate valuation of clocks.

According to the methodology proposed in [4], product of network of timed automata can be built. The runs of the product of timed automata's network will be considering as suitable actions sequences. From all of the runs in the product of timed automata's will be constructed the tree. All of transitions from automata's components, labeled with the same actions, they will be synchronized with each other. Other transitions can be executed in any order but the sequence must result from the automata's components.

Automata modeling executions of the protocols must be synchronized with the automata modeling the users' knowledge. With the proper synchronization, the correct executions tree of the examined protocol can be obtained.

Let's consider protocol $\Sigma = (\alpha_1, \dots, \alpha_n)$ and its partial interpretation $\overline{f}$. All of the executions $f(\Sigma)$ where f is related to $\overline{f}$ are modeled by following automata of executions $A_{\overline{f}} = (\Sigma_{\overline{f}}, Q_{\overline{f}}, s_0^{\overline{f}}, \mathcal{X}_{\overline{f}}, \delta_{\overline{f}})$ where:

- $\Sigma_{\overline{f}} = \left\{ \mathrm{k}_{\overline{f},i} \,\middle|\, 1 \le i \le n \wedge Send^{f(\alpha_i)} \in \mathbf{P} \right\} \cup \bigcup_{i=1}^{n} \cup \{\{k_{\overline{f}}^X\} | \, X \subseteq L \wedge Send^{f(\alpha_i)} \in \mathbf{P_t} \wedge X \vdash Lett^{f(\alpha_i)}\}$,
- $Q_{\overline{f}} = \{s_0^{\overline{f}}, s_1^{\overline{f}}, s_2^{\overline{f}}, \dots, s_{0n}^{\overline{f}}\}$ is a set of state where $s_0^{\overline{f}}$ is the initial state,

- $\mathcal{X}_{\overline{f}} = \bigcup_{i=1}^{n}\{z_t \mid t \in (\mathbf{T} \cap Gen^{f(\alpha_i)})\}$,

  $\delta_{\overline{f}} = \{\left(s_{i-1}^{\overline{f}}, k_{\overline{f},i}, Z\left(TConstr^{\overline{f}(\alpha_i)}\right), \{z_t \mid t \in \left(\mathbf{T} \cap Gen^{f(\alpha_{i-1})}\right)\}, s_i^{\overline{f}}\right) \mid$
- $1 \le i \le n \wedge \mathrm{k}_{\overline{f},i} \in \Sigma_{\overline{f}}\} \cup \{(s_{i-1}^{\overline{f}}, k_{\overline{f},i}^{X}, Z\left(TConstr^{\overline{f}(\alpha_i)}\right), \{z_t \mid$
  $t \in \left(\mathbf{T} \cap Gen^{f(\alpha_{i-1})}\right)\}, s_i^{\overline{f}}) \mid 1 \le i \le n \wedge k_{\overline{f},i}^{X} \in \Sigma_{\overline{f}}\}$

Defined inductively time conditions $Z\left(TConstr^{\overline{f}(\alpha_i)}\right)$ are as follows:

1. $Z(true) = true$,
2. $Z(\tau_i + \delta - \tau_p \le l) = z_{t_p} \le l$,
3. $Z(TConstr_1 \wedge TConstr_2) = Z(TConstr_1) \wedge Z(TConstr_2)$.

Each state $s_i^{\overline{f}}$ of automata is reachable after execution of one of the steps of $f(\alpha_i)$ in execution $f(\Sigma)$ for some f associated with $\overline{f}$. In addition, the state $s_i^{\overline{f}}$ can be reachable only when $TConstr^{f(\alpha_i)} = true$ and all clocks $\{z_t \mid t \in \left(\mathbf{T} \cap Gen^{f(\alpha_{i-1})}\right)\}$ are reset.

If the sender of this step is honest, there will be only one way to execute this step. The sender must have adequate knowledge to compose a letter sent in this step. However, if the sender is the Intruder, they have a number of opportunities to execute this step determined by sets of letters generators. Each of these cases is denoted with another label $k_{\overline{f},i}^{X}$ for different sets of generators of the letter *X*.

Example 3.1. *Automaton of executions for WMF protocol for partial interpretation* $\overline{f}$ *where* $\overline{f}(A) = a, \overline{f}(B) = b, \overline{f}(S) = s, \overline{f}(L_F) = l$, *are as follows:*

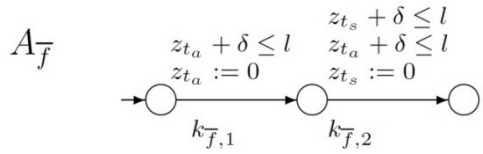

Fig. 1. Automaton of executions for the WMF protocol

Figure 1 presents the automaton of executions for the Wide Mouth Frog protocol. This automaton models a transmission of messages executions of the WMF protocol according to the *f*. This automaton cannot be accepted as the full model of execution because it does not model the user's knowledge which is necessary to execute the following steps.

4. Conclusions

The security of information transmission is a very important part of the network and computer systems. The assurance an adequate level of security is associated

with selecting appropriate communications protocol that will protect all communications. However, computer networks and protocols themselves are exposed to unauthorized people called *the Intruders*. Therefore, it was necessary to verify the security protocols.

This paper presented a formal model of security protocols executions which takes network delays into account. In this case, the examination of the protocols is necessary from practical point of view. While creating network infrastructure, such verification to validate actions and susceptibility for attacks of this infrastructure should be made before launching the whole infrastructure.

A synchronized network of timed automata have been applied to present a formal model of security protocols. This model adequately reproduces the real executions of the protocols in networks. As a result, it is possible to verify the actions of modeled protocols. Also a tool for automatic verification of protocols is being created.

In the next studies we will review the timed property of time-dependent protocols that take into account the delay in networks.

Acknowledgement

The first author acknowledged a support from the grant BS/MN-1-112/303/15/P.

References

[1] Kurkowski M., Penczek W., Applying timed automata to model checking of security protocols, [in:] Handbook of Finite State Based Models and Applications, ed. J. Wang, Chapman and Hall/CRC Press, 2013, 223-254.

[2] Kurkowski M., Grosser A., Piątkowski J., Szymoniak S., ProToc - an universal language for security protocols specification, Advances in Intelligent Systems and Computing 2015, 342, 237-248.

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

[4] Burrows M., Abadi M., Needham R., A Logic of Authentication, Proceedings of the Royal Society of London A, 1989, vol. 426, 233-271. A preliminary version appeared as Research Report 39, DEC Systems Research Center, Palo Alto, February 1989.

[5] Dolev D., Yao A., On the security of public key protocols, IEEE Transactions on Information Theory 1983, 29(2), 198-208.

[6] Kurkowski M., Formalne metody weryfikacji własności protokołów zabezpieczających w sieciach komputerowych, Wyd. Exit, Warszawa 2013.

[7] Benerecetti M., Cuomo N., Peron A., TPMC: A model checker for time-sensitive security protocols, Journal of Computers, North America, 2009, 4, 5, 366-377.

[8] Armando A., Basin D., Boichut Y., Chevalier Y., Compagna L., Cuellar J., Hankes Drielsma P., Heám P.C., Kouchnarenko O., Mantovani J., Mödersheim S., von Oheimband D., Rusinowitch M., Santiago J., Turuani M., Viganó L., Vigneron L., The AVISPA tool for the automated validation of internet security protocols and applications, Proc. of 17th International Conference on Computer Aided Verification (CAV'05), vol. 3576 of LNCS, Springer-Verlag, 2005, 281-285.

[9] Cremers C., Feltz M., Operational Semantics and Verification of Security Protocols, Information Security and Cryptography series, Springer, 2012.

[10] Jakubowska G., Penczek W., Modeling and checking timed authentication security protocols, Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS\&P'06), Informatik-Berichte Humboldt University, 2006, 206(2), 280-291.

[11] Jakubowska G., Penczek W., Is your security protocol on time? Proc. of the IPM Int. Symp. on Fundamentals of Software Engineering (FSEN'07), LNCS, vol. 4767, Springer-Verlag, 2007, 65-80.

[12] Penczek W., Półrola A., Advances in verification of time petri nets and timed automata: A temporal logic approach, Studies in Computational Intelligence 20, Springer-Verlag 2006.