Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(2): 1136–1153.
DOI: 10.3934/mbe.2021061
Received: 28 September 2020
Accepted: 17 December 2020
Published: 11 January 2021

Research article

Security protocols analysis including various time parameters

Sabina Szymoniak*

Department of Computer Science, Czestochowa University of Technology, Dabrowskiego 69, 42-201 Czestochowa, Poland

* **Correspondence:** Email: sabina.szymoniak@icis.pcz.pl.

Abstract: Communication is a key element of everyone's life. Nowadays, we most often use internet connections for communication. We use the network to communicate with our family, make purchases, we operate home appliances, and handle various matters related to public administration. Moreover, the Internet enables us to participate virtually in various official meetings or conferences. There is a risk of losing our sensitive data, as well as their use for malicious purposes. In this situation, each connection should be secured with an appropriate security protocol. These protocols also need to be regularly checked for vulnerability to attacks. In this article, we consider a tool that allows to analyze different executions of security protocols as well as simulate their operation. This research enables the gathering of knowledge about the behaviour of the security protocol, taking into account various time parameters (time of sending the message, time of generation confidential information, encryption and decryption times, delay in the network, lifetime) and various aspects of computer networks. The conducted research and analysis of the obtained results enable the evaluation of the protocol security and its vulnerability to attacks. Our approach also assumes the possibility of setting time limits that should be met during communication and operation of the security protocol. We show obtained results on Andrew and Needham Schroeder Public Key protocols examples.

Keywords: security protocols; timed analysis; formal verification; network application; attacks

1. Introduction

We live in times where it is difficult to imagine simple daily activities without the use of various types of technology. Our refrigerators, ovens, dishwashers and most of all homes are smart. The devices and applications we use suggest and remind us what we should do at the moment. Many issues can also be solved using devices and applications without leaving home (e.g. shopping, payments). A similar situation can also be observed in the case of interpersonal communication. Talking to someone in a place tens of thousands of kilometres away is not a problem. For example, we can use available

instant messengers here. Similar solutions are also possible for public administration. Official matters can also be dealt with via electronic links and dedicated applications without a physical presence in the administrative unit.

To be able to perform such activities, it is necessary to properly secure the operation of devices and applications, as well as secure communication. Security here is a key element that affects our identity and confidentiality of our data. Without properly secured communication, our private information could fall into the wrong hands and be used. Time is an essential element of safety. Every second can contribute to break our passwords and steal confidential information.

Protocols are used to secure operation and communication. Among all protocols, one can distinguish the class of security protocols. These protocols are short programs that are built up of several steps. By performing these steps, users confirm their identity to each other. Therefore, their task is to ensure an appropriate level of security.

One of the first security protocols was the Needham Schroeder [1] protocol presented in 1978. This protocol was designed to provide mutual authentication using a trusted server. The following years favoured the emergence of further protocols securing communication. There were also suggestions that it should be checked whether the protocol used provides security and is not susceptible to attacks by wicked people called Intruders [2]. The aforementioned Needham Schroeder protocol was considered safe until Gavin Lowe showed a straight path to compromising him [3, 4].

The following years also contributed to the development of topics related to the verification of security protocols. Various verification methods have emerged over the years. In [5] Burrows et al. presented a logic which allows to the beliefs of trustworthy parties involved in protocols. In [6] Paulson et al. introduced an inductive method for security protocols verification. In [7] Kurkowski et al. presented a method of modelling the execution of protocols using a network of synchronized automata. In [8] Nguyen et al. introduced a class of security protocol abstractions. The abstractions transformed a term's structure based on its type. In [9, 10] Steingartner et al. presented a method for modelling behaviour of computer systems using coalgebras. In [11] authors introduced a model of hybrid threats, which allows responding to security threats.

Also, various special tools for automatically checking protocol security have been created. The well-known tools are AVISPA et al. [12], Verics et al. [13], Scyther [14, 15], UPPAAL et al. [16], ProVerif [17] and Tamarin et al. [18]. They are effective at finding attacks on protocols and fixing their correctness.

In this paper, we show a tool that also serves to automatically verify security protocols including time parameters. This tool implements the assumptions related to the specification and verification of timed security protocols presented in [19–23].

Thanks to this, it is possible to analyze the duration of protocol executions taking into account delays in the network, as well as to simulate these executions for the values of delay in the networks generated with different probability distributions.

The main contribution of this paper is the security protocols verification tool which takes into account delays in the network. The included time intervals of delays in the network values make it possible to control the values during the tests. The used research methodology and consideration simulations of encryption and decryption times distinguish our tool from other similar tools. Simulations of encryption and decryption times enable to present the operation of the computer network. Also, it is possible to check what possibilities the Intruder will have in various time

situations.

The rest of this paper is organised as follows. In the second section, we present the MobInfoSec protocol as an example of a security protocol. Next Section shows our research methodology. In the fourth section, we present our tool for security protocol verification. The fifth section consists of experimental results for Andrew and Needham Schroeder Public Key protocols. The last section includes our conclusions and plans for the future.

2. MobInfoSec protocol

A protocol is a sequence of steps, which is an exchange of messages. This exchange aims to ensure an appropriate level of security, using cryptographic techniques. The execution of this algorithm type is concurrent (several parties can communicate with each other at the same time). Among the cryptographic protocols, we can distinguish a class of security protocols. These protocols ensure the security of the communicating users and they are based on encrypting and authentication.

As an example of the security protocol, we show the MobInfoSec protocol. This security system connects a group of users that are using mobile devices. Each user's device in this system has two modules. First of them is Module Secret Protection (SP) which is used for the generation of random number. The second is the Authentication Module (MU) which is used to the transmission of messages. This connection of two modules is called *trusted domain*. The scheme of this protocol is as follows [24, 25]:

$$
\begin{aligned}
\alpha_1 \quad & SP.A \rightarrow MU.A: && \{N_{SP.A}, i(SP.A)\} \\
\alpha_2 \quad & MU.A \rightarrow MU.B_i: && \{N_{SP.A}, i(SP.A)\} \\
\alpha_3 \quad & MU.B_i \rightarrow SP.B_i: && \{N_{SP.A}, i(SP.A)\} \\
\alpha_4 \quad & SP.B_i \rightarrow MU.B_i: && \{\{N_{SP.B_i}, -k_{SP.B_i}, \\
& && h(N_{SP.B_i}, N_{SP.A}, i(SP.A))\}_{-k_{SP.B_i}}\}_{+k_{SP.A}} \\
\alpha_5 \quad & MU.B_i \rightarrow MU.A: && \{\{N_{SP.B_i}, -k_{SP.B_i}, \\
& && h(N_{SP.B_i}, N_{SP.A}, i(SP.A))\}_{-k_{SP.B_i}}\}_{+k_{SP.A}} \\
\alpha_6 \quad & MU.A \rightarrow SP.A: && \{\{N_{SP.B_i}, -k_{SP.B_i}, \\
& && h(N_{SP.B_i}, N_{SP.A}, i(SP.A))\}_{-k_{SP.B_i}}\}_{+k_{SP.A}}
\end{aligned}
$$

where:

- $i(SP.A)$ - the identifier of module SP of user A,
- N_X - nonces (random nubmers),
- $h(X)$ - hash function on message X,
- $-k_{SP.B_i}$ - public key,
- $+k_{SP.A}$ - private key.

In the first step of this protocol module SP of user A generates a nonce $N_{SP.A}$. It composes a message with this nonce and its identifier $i(SP.A)$. Next, module $SP.A$ transmits this message to the MU module. Received message is transmitted to module $MU.B_i$ in second step and to module $SP.B_i$ in third step. Next, user's B_i secret protection module generates nonce $N_{SP.B_i}$. In the fourth step, $SP.B_i$ module creates a message which contains:

- nonce $N_{SP.B_i}$,

- public key $-k_{SP.B_i}$,
- the result of a hash function for $N_{SP.B_i}$, $N_{SP.A}$ and $i(SP.A)$.

Next, this message is encrypted two times: first, by the public key $-k_{SP.B_i}$, and second, by the private key $+k_{SP.A}$. Such composed message goes to the $SP.A$ in a similar way to the previous message:

- from $SP.B_i$ to $MU.B_i$ in fourth step,
- from $MU.B_i$ to $MU.A$ in fifth step,
- from $MU.A$ to $SP.A$ in sixth step.

After that, all SP modules know $N_{SP.A}$ and $N_{SP.B_i}$ (for $i = 1, ..., n$). Next, it is possible to calculate the new symmetric key for theirs trusted channels.

MobInfoSec protocol is secure. So far, no attack on this protocol has been found.

3. Materials and method

In our model, we consider the following time parameters: timestamp, lifetime, time of composing the message, delay in the network as well as step and session times. The timestamp is the unique identifier of the generation of a message, the value of which sets the clock of the unit sending the message. The lifetime is strongly related to the timestamp, which determines for what time, starting from the moment of generating the timestamp, each element contained in a given message can be used. The time of composing the message is related to the time it takes to complete all operations such as encryption and random number generation. The delay in the network is the time of transmission of a message between the sender and the recipient of the computer network links. The step time is the duration of all operations related to its execution, while the session time is the expected time to complete all protocol steps.

In case of step and session times, we distinguished three possible signs: minimal, current and maximal. This distinction takes into account the range of delay in the network values. During our research, we assume a range of values for delays in the network. On this basis, we calculate and set limits for the duration of steps and sessions, and lifetime.

The lifetimes values for steps are calculated according to the following formula:

$$T_k^{out} = \sum_{i=k}^{n} T_i^{max}. \tag{3.1}$$

where:

- k is a step number,
- i is a step counter $i = k + 1...n$,
- n is a number of protocol's steps,
- T_k^{out} is a lifetime in the k-th step,
- T_i^{max} is the maximum step time.

If the step time value exceeds the lifetime time imposed in the step time condition, execution will fail, and all communication should be immediately terminated.

Mentioned parameters allow us to calculate and set values for time conditions that can be imposed on the execution of steps and the entire protocol. Methods for calculating such parameters as the time of composing the message, step time, session time and lifetime are presented in [26, 27].

It is worth mentioning, to accurately illustrate the operation of the protocol in a computer network, we are considering ranges of time parameters.

4. Security protocols verification tool

The security protocols verification tool has been designed to check the vulnerability of the protocol to attacks depending on the selected time parameters. Time parameters are designed to model the real working environment of the protocol, i.e. the computer network. Regardless of the situation in which the protocol is used (logging in to the office, logging in to the bank, logging into a social networking site or messenger), the background of the environment in which we work is essential. The load on a computer network can affect the response speed of the host we want to connect to. Our tool allows analysing confidentiality and authentication.

4.1. Intruder's models

Each real execution of the protocol in the computer network may be exposed to the actions of an intruder who may try to deceive other users. Its operation will prevent the objectives of the protocol from being achieved. intruder's possible behaviours and actions include impersonating another user, improperly structuring the message, and sending the same message multiple times. Our tool includes four Intruder models: Dolev-Yao model, Lazy Intruder, restricted Dolev-Yao, and restricted Lazy Intruder.

The most common and at the same time the most favoured Intruder model is the Dolev-Yao [2] model. Here, the intruder controls the network unlimitedly. The intruder has access to all messages that are transmitted. These messages can be intercepted, blocked, and even processed and transmitted by it against the protocol. The intruder will not have access to the information contained in the ciphertext unless he knows the appropriate decryption key.

The Lazy Intruder model is described in Kassem et al. [28] and Mdersheim et al. [29, 30]. The Lazy Intruder can represent malware that can only transmit whole messages but cannot be modified.

We are also introducing an additional restriction to these models, which allows the Intruder to use only those messages that have been sent directly to him (as an honest user). These models are labelled restricted Dolev-Yao constrained model and restricted Lazy Intruder [19].

4.2. Preparation of the protocol

The tool starts by loading the protocol specification, which is written in ProToc [31]. The ProToc language enables the presentation of internal and external actions performed during the protocol. The following specifications can be read from the protocol specification:

- set of protocol participants,
- sender's needed to complete each step,
- objects that must be generated by the sender,
- messages.

For example, for the first step of the MobInfoSec protocol, the listed sets will contain the following elements:

- set of protocol participants: $SP.A$ to $MU.B_i$,
- set of sender's knowledge needed to execute the step: nonce $N_{SP.A}$ and identifier $i(SP.A)$,
- set of generated objects: nonce $N_{SP.A}$,
- set of messages: $\{N_{SP.A}|i(SP.A)\}$.

Then, all structures included in the file are parsed to objects of defined classes. Specification of MobInforSec protocol in ProToc is as follows:

```
u=4;
p=5;
s=6;
protocol;
p_1,p_2;i(p_1),s_1(1);s_1(1);i(p_1)|s_1(1);
p_2,p_3;i(p_1),s_1(1);;i(p_1)|s_1(1);
p_3,p_4;i(p_1),s_1(1);;i(p_1)|s_1(1);
p_4,p_3;s_2(1),s_1(1),k+_2(1),i(p_1),k-_1(1);s_2(1);
    <k-_1(1),<k+_2(1),s_2(1)|k+_2(1)|<k+_2(1),s_2(1)|s_1(1)|i(p_1)>>>;
p_3,p_2;s_2(1),s_1(1),k+_2(1),i(p_1),k-_1(1);;
    <k-_1(1),<k+_2(1),s_2(1)|k+_2(1)|<k+_2(1),s_2(1)|s_1(1)|i(p_1)>>>;
p_2,p_1;s_2(1),s_1(1),k+_2(1),i(p_1),k-_1(1);;
    <k-_1(1),<k+_2(1),s_2(1)|k+_2(1)|<k+_2(1),s_2(1)|s_1(1)|i(p_1)>>>;
```

For timed version* of MobInfoSec protocol we have four users (`u = 4;`), five players† (`p = 5;`) and six steps (`s = 6;`). The first step of the MobInfoSec protocol contain the following elements:

- set of protocol participants: `p_1`, `p_2`,
- set of sender's knowledge needed to execute the step: timestamp `s_1(1)` and identifier `i(p_1)`,
- set of generated objects: nonce `s_1(1)`,
- set of messages: `i(p_1)|s_1(1)`.

In the next stage, the tool generates all potentially possible executions of the tested protocol, including the wicked user, i.e. the Intruder. These executions differ from each other in terms of execution time, as well as the order in which the participants in the protocol appear, including the Intruder and the parameters of his activities.

4.3. Preparation of the knowledge

The next step is to create structures that allow storing the knowledge of participants in the execution of the protocol and calculating and assigning fixed values of time parameters.

The knowledge of each participant must be initiated with so-called *initial knowledge*, which are public and private keys and symmetric keys shared with other participants in the protocol. An

*Timed version means that we exchange nonces by timestamps in protocol structure. Converting nonce to timestamps allows to unambiguously determine when a given message was created.

†Section *players* includes honest users and Intruder.

additional collection here is the public knowledge collection, i.e. objects that are publicly available (user IDs and public keys). The methods of acquiring knowledge by users are presented in [22] and implemented in the tool.

For example, for the first three steps of the MobInfoSec protocol, the knowledge collection of the relevant users (modules) that are recipients in these steps will increase by the nonce $N_{SP.A}$. The $i(SP.A)$ identifier is publicly available information. In turn, in the next steps, where encryption occurs, the knowledge collections of the relevant modules will increase by the entire ciphertext and by those cryptographic objects that will be able to obtain from the ciphertext as a result of its decryption. Because the message is encrypted with the private key of the $SP.A$ module, decryption will not be performed in steps four and five. It will not appear until the sixth step. In this case, the knowledge of the $SP.A$ module will increase by the entire ciphertexts:
$\{\{N_{SP.B_i}, -k_{SP.B_i}, h(N_{SP.B_i}, N_{SP.A}, i(SP.A))\}_{-k_{SP.B_i}}\}_{+k_{SP.A}}$,
and
$\{N_{SP.B_i}, -k_{SP.B_i}, h(N_{SP.B_i}, N_{SP.A}, i(SP.A))\}_{-k_{SP.B_i}}$
and all the elements in the second ciphertext.

4.4. Preparation of the research

Then we can choose one of three types of tests (times analysis, simulations of the delays in the network and simulations of encryption and decryption times).

4.4.1. Timed analysis

The first is the so-called timed analysis. In this part of the test, all protocol executions are tested using fixed time parameter values. We assume several time units for the time of message encryption and decryption, and for the time of generating confidential information, while for the delay in the network parameter we use a range of values. Then, for such selected parameters, the values of the minimum and maximum protocol step time (sum of encryption and decryption times, generation time and delay in the network time), as well as minimum and maximum session time, are calculated. These values allow us to classify the correctness of the given protocol execution. The next stage of the tool's operation is the calculation of lifetimes values for steps according to the formula mentioned in Section 3.

According to the described assumptions, we also assume that a correctly completed execution should obtain a time between the minimum and maximum session time. The Intruder's activity involves the ability to send entire ciphertexts without the need to encrypt them. Therefore, encryption times will not be added to the time of steps and sessions, i.e. such protocol execution may end before the minimum session time.

Due to the Intruder's activity and additional steps taken to acquire knowledge, some executions may end above the maximum session time. This situation is only possible if the time conditions imposed on individual steps of the protocol are observed.

During the analysis of execution times, the current value of the delay in the network is the lower limit of the accepted range of values for this parameter. The result of this type of research will be information on whether the protocol is secure for such time parameters. We can, therefore, conclude that with a minimal delay in the network, an Intruder can launch an attack.

4.4.2. Simulations of the delays in the network

Simulations of the delays in the network are the second type of research. These studies will aim to show the impact of the delay value in the network on the time of execution and the capabilities of the Intruder. The purpose of introducing timed simulations to our research is to present a real computer network operation.

In this case, we also accept several time units for the time of message encryption and decryption, and for the time of generating confidential information, and for the delay in the network parameter we use a range of values. The values for the minimum and maximum protocol step time and lifetime values are calculated in the same way.

The difference between execution time analysis and delays in the network simulations will be based on the current value of the delay in the network (for the step currently being analyzed). In the case of simulations, current values of delays in the network will be randomized according to the selected probability distributions. Normal, uniform, Cauchy's, and exponential distributions were selected for the study. Probability distributions were selected so that it was possible to present different loads on the computer network. Also, it was assumed that it is possible to randomly draw values outside the accepted range to fully model the operation of a computer network.

During this part of the research, each of the generated executions is subjected to a thousand tests for the selected probability distribution. Thanks to this, we can determine the impact of an Intruder's work on the computer network.

4.4.3. Simulations of encryption and decryption times

The third type is simulations of encryption and decryption times. In this case, we also take several time units for the time of generating confidential information, and for the delay in the network parameter, we use a range of values. The values for the minimum and maximum protocol step time and lifetime values are calculated in the same way.

The difference between delays in the network simulations and encryption and decryption time simulations is related to two aspects. First of all, we consider the constant delay in the network value (the lower limit of the accepted range) for the current value of this parameter. Secondly, the encryption and decryption times are randomized according to a uniform probability distribution. This allows us to model the speed of computer units on which the participants of the protocol work.

Also, during this part of the research, each of the generated executions is subjected to a thousand tests. Thanks to this, we can determine the impact of the power of computer units on the activities of an Intruder with minimal load on the computer network.

4.5. Tool advantages

The presented tool allows checking how the selected security protocol will behave in various situations. Also, we can show his safety and sensitivity to the action of an Intruder, taking into account various values of time parameters.

The main advantage of this tool is taking into account delays in the network. This feature distinguishes them from other similar tools. Additionally, time intervals have been taken into account for this parameter. Time intervals make it possible to control the values during the tests. An important aspect of the tool are simulations of delay in the network values and simulations of encryption and

decryption times. Thanks to them, it is possible to present the operation of the computer network and to check what possibilities the Intruder will have in various time situations.

5. Experimental results

To show how our tool works, we have conducted many tests on various security protocols. The tests were carried out using a computer unit with the Linux Ubuntu operating system, Intel Core i7 processor, and 16 GB RAM. Also, we used an abstract time unit ([tu]) to determine the time. This time unit could be considered as any period. Also, we assumed that the Intruder could impersonate only honest users.

It is possible to test any security protocol written in the ProToc language [31] with our tool. The verification of each protocol is provided in three stages:

- timed analysis,
- simulations of the delays in the network,
- simulations of encryption and decryption times.

Our tool returns one of four statuses for the tested executions. The first is status *correct*. It ends with those executions whose session time was in the range of $< T_{ses}^{min}, T_{ses}^{max} >$. Status *!min* is associated with executions whose session time was less than T_{ses}^{min}. Status *!max* is associated with executions whose session time was greater than T_{ses}^{min}. Completion of execution with one of these three statuses was possible while maintaining the time conditions imposed on the protocol steps. The last status (*error*) marked executions that ended in an error as a result of not meeting the time conditions.

As mentioned, the MobInfoSec protocol is secure. No attack on this protocol has been found so far. We checked this protocol, but during our investigations, only honest executions ended with status *correct*. By *honest executions*, we denote executions between honest users without Intruder. We will present our results on Andrew and Needham Schroeder Public Key protocols examples. The complexity of these protocols allows for an accurate presentation of the obtained results.

5.1. Timed analysis

Timed analysis of executions will be shown on Andrew protocol[‡] example [32].

Encryption or decryption time depends directly on the data size and the processing power of the system where it is being executed. So, assumptions should be related to those parameters. For our research we determined the following assumptions:

- time of generating confidential information: $T_g = 1[tu]$,
- time of composing the message $T_c = 1[tu]$,
- encryption time $T_e = 3[tu]$,
- decryption time $T_d = 3[tu]$,

[‡]The syntax of the timed version of Andrew protocol in Common Language is as follows:

$\alpha_1 \quad A \to B: \quad i(A), \{T_A\}_{K_{AB}}$,
$\alpha_2 \quad B \to A: \quad \{T_A, T_B\}_{K_{AB}}$,
$\alpha_3 \quad A \to B: \quad \{T_B\}_{K_{AB}}$,
$\alpha_4 \quad B \to A: \quad \{K'_{AB}, T_B\}_{K_{AB}}$.

In this notation A, B mean honest users, $i(A)$ means user's identifier, T_A and T_B mean timestamps of users marked by a subscript, and K_{AB} means symmetric key shared between users marked by subscript.

Table 1. Timed analysis of attacking execution (Andrew protocol).

Step	Progress of the step	Step time	Comment
α_1	α_1 (6)	6	ok
β_1	β_1 (4)	4	ok
β_2	β_2 (6)	6	ok
α_2	β_1 (4), β_2 (6), α_2 (4)	14	ok
α_3	α_3 (5)	5	ok
β_3	α_2 (4), α_3 (5), β_3 (4)	13	ok
β_4	β_4 (6)	6	ok
α_4	β_3 (4), β_4 (6), α_4 (4)	14	T^4_{out}

- minimal delay in the network $D_{min} = 1[tu]$,
- maximal delay in the network $D_{max} = 3[tu]$.

Next, we calculated the values of the minimum and maximum session time and lifetime values, according to mentioned methodology:

- $T^{min}_{ses} = 35[tu]$,
- $T^{max}_{ses} = 43[tu]$,
- $T^1_{out} = 43[tu]$,
- $T^2_{out} = 32[tu]$,
- $T^3_{out} = 21[tu]$,
- $T^4_{out} = 11[tu]$.

Let's examine one of the executions that attack Andrew's protocol (1)[§]. This execution maps to a *Man in the Middle* attack.

$$\begin{array}{rlcl} \alpha_1 & A \to I(B) & : & A, \{T_A\}_{K_{AB}}, \\ \beta_1 & I(A) \to B & : & A, \{T_A\}_{K_{AB}}, \\ \beta_2 & B \to I(A) & : & \{T_A, T_B\}_{K_{AB}}, \\ \alpha_2 & I(B) \to A & : & \{T_A, T_B\}_{K_{AB}}, \\ \alpha_3 & A \to I(B) & : & \{T_B\}_{K_{AB}}, \\ \beta_3 & I(A) \to B & : & \{T_B\}_{K_{AB}}, \\ \beta_4 & B \to I(A) & : & \{K'_{AB}, T_B\}_{K_{AB}}, \\ \alpha_4 & i(B) \to A & : & \{K'_{AB}, T_B\}_{K_{AB}}. \end{array} \tag{5.1}$$

According to the principles of the *Man in the Middle* attack, the Intruder mediates between two honest users (A, B) and sends the entire ciphertexts that it receives from them. The Intruder does not encrypt or decrypt in any of the steps. The times of these operations are not added to the duration of the steps and sessions.

Table 1 presents the analysis of the times of one of the attacking executions generated for the Andrew protocol. The interlacing of two executions has been taken into account here (according to (1)). Execution steps have been marked in the *Step* column. The column *Progress of the step* contains information about which steps should be included in the duration of the current step. The total text

[§]In this notation *I(A)*, *I(B)* mean Intruder who impersonates user A and B, resepectively.

duration of this step has been included in the *Step time* column. The column *Comment* contains a comment about the course of the given step.

In the first step of this execution (α_1), we only include operations from this step. Thus, we include the times: nonce generation (1), message composition (1), encryption by the sender (3) and delay in the network (1). However, we do not add decryption time to the recipient because the recipient is an Intruder who does not have the K_{AB} decryption key. The duration of this step is 6 [tu]. Similarly, we should consider the next steps (β_1 and β_2), bearing in mind that the Intruder in step β_1 sends the finished message, so the generation, composing and encryption times will not be added here.

It is worth paying attention to the step α_2. To complete this step, we must first complete the β_1 and β_2 steps. Therefore, the duration of these steps will be added to the total duration of the α_2 step.

The first seven steps of this execution will run smoothly. The time conditions imposed will be retained here. The problem will appear when executing the α_4 step. In this case, we need to add the duration of the steps β_2 (without α_2 and α_3) and β_3. The total duration of this step is 14 [tu]. Lifetime T^4_{out} will be exceeded, which means that communication should be immediately interrupted. An Intruder cannot launch a *Man in the Middle* attack on such timed parameters.

Let's analyse the Man in the Middle attack on Needham Schroeder Public Key (NSPK) protocol[¶] [1].

We calculated following lifetimes for this protocol:

- $T^{out}_1 = 29$ [tu],
- $T^{out}_2 = 19$ [tu],
- $T^{out}_3 = 9$ [tu].

Also, we calculated minimal and maximal session times:

- $T^{min}_s = 23$ [tu],
- $T^{max}_s = 29$ [tu].

The Man in the Middle attack, which is an interlacing of the α and β executions, is as follows.

$$\begin{array}{llll} \alpha_1 & A, I(B) & : & \{I_A, T_A\}_{K_B}, \\ \beta_1 & I(A), B & : & \{I_A, T_A\}_{K_B}, \\ \beta_2 & B, I(A) & : & \{T_A, T_B\}_{K_A}, \\ \alpha_2 & I(B), A & : & \{T_A, T_B\}_{K_A}, \\ \alpha_3 & A, I(B) & : & \{T_B\}_{K_B}, \\ \beta_3 & I(A), B & : & \{T_B\}_{K_B}. \end{array}$$

As with the Andrew protocol, the Intruder sends all encrypted messages. It does not perform encryption and decryption operations. These times are not added to the total step time.

Table 2 presents a timed analysis of Man in the Middle execution for the NSPK protocol. Please note that the Intruder executed additional steps and acquired the necessary knowledge before lifetimes. A Man in the Middle attack is possible on such timed parameters.

[¶]Syntax of NSPK timed version in Common Language is as follows:
$\alpha_1 \quad A \to B: \quad \{T_A, I_A\}_{K_B}$;
$\alpha_2 \quad B \to A: \quad \{T_A, T_B\}_{K_A}$;
$\alpha_3 \quad A \to B: \quad \{T_B\}_{K_B}$.

Table 2. Timed analysis of attacking execution (NSPK protocol).

Step	Progress of the step	Step time	Comment
α_1	α_1 (5)	5	ok
β_1	β_1 (4)	4	ok
β_2	β_2 (5)	5	ok
α_2	β_1 (4), β_2 (5), α_2 (4)	13	ok
α_3	α_3 (4)	5	ok
β_3	α_3 (4), β_3 (4)	8	ok

The presented examples showed how many additional steps an Intruder can execute if too large time limits for the response are set in the computer network. First of all, it is necessary to analyse the set time limits to prevent such situations. Then, set the constraint values so that honest users have enough time to execute their operations and that an Intruder cannot acquire additional knowledge.

5.2. Delays in the network simulations

The results of delays in the network simulations will be presented on the example of the NSPK protocol.

For this stage of our research we modified the following assumptions:

- minimal delay in the network $D_{min} = 1[tu]$,
- maximal delay in the network $D_{max} = 10[tu]$.

Other assumptions remained the same as in the case of time analysis.

Next, we calculated the values of the minimum and maximum session time and lifetime values, according to NSPK protocol structure:

- $T_{ses}^{min} = 25[tu]$,
- $T_{ses}^{max} = 52[tu]$,
- $T_{out}^{1} = 52[tu]$,
- $T_{out}^{2} = 34[tu]$,
- $T_{out}^{3} = 16[tu]$.

For NSPK protocol we generated eighteen different executions. Each of them was tested in 1000 test with different delay in the network values.

5.2.1. Uniform probability distribution

First, we used uniform probability distribution to generate delay in the network values. For this probability distribution we obtained following results:

- 6842 sessions ended in correct time,
- 4142 sessions ended upper then T_{ses}^{max},
- 7016 sessions ended with error.

This number of errors results from unfulfilled time conditions imposed on individual steps. An Intruder does not always have the right set of knowledge to carry out an execution. For this reason,

Table 3. Summary of times for uniform probability distribution.

Status	Average D [tu]	Minimal T_{ses}[tu]	Average T_{ses} [tu]	Maximal T_{ses}[tu]
correct	5.36	25	38.5	52
!max	5.45	52.1	62.76	87

Table 4. Summary of times for normal probability distribution.

Status	Average D [tu]	Minimal T_{ses}[tu]	Average T_{ses} [tu]	Maximal T_{ses}[tu]
correct	5.08	17	20.4	22.82
!min	7.82	23	41.38	50
!max	10.05	52.1	62.39	118.2

Intruder must take additional steps to acquire knowledge. The duration of these steps affects the fulfilment of time conditions and the time of the entire session. For all tests, the minimum delay in the network was 1 [tu], the average was 5,47 [tu] and the maximum was 10 [tu].

The Table 3 presents a summary of the times for these executions in which the imposed time conditions have been preserved. We have highlighted the average delay in the network values and the minimum, average and maximum values of the session time. The summary takes into account the division into statuses with which sessions ended.

5.2.2. Normal probability distribution

Next, we used normal probability distribution to generate delay in the network values. For this probability distribution we obtained following results:

- 1698 sessions ended in correct time,
- 18 sessions ended below then T_{ses}^{min},
- 1710 sessions ended upper then T_{ses}^{max},
- 14,574 sessions ended with error.

For all tests, the minimum delay in the network was 1.03 [tu], the average was 12.43 [tu] and the maximum was 37 [tu]. In case of this stage of research we allow to draw numbers out of range of $< D_{min}, D_{max} >$. Delays in the network values were greater than D_{max}.

Similarly, as Table 3, the Table 4 presents a summary of the times for these executions in which the imposed time conditions have been preserved. Please note that even value of delay in the network is huge, steps may be correctly executed.

5.2.3. Cauchy's probability distribution

Next, we used Cauchy's probability distribution to generate delay in the network values. The obtained results were following:

- 782 sessions ended in correct time,
- 7 sessions ended below then T_{ses}^{min},
- 982 sessions ended upper then T_{ses}^{max},
- 16,229 sessions ended with error.

Table 5. Summary of times for encryption and decryption simulations.

Status	Minimal T_e[tu]	Average T_e [tu]	Maximal T_e[tu]
correct	1.33	5.51	10
!min	2.67	6.43	9.67
!max	1.67	5.66	10

For all tests, the minimum delay in the network was 1 [tu], the average was 60.57 [tu] and the maximum was 17,389.9 [tu].

5.2.4. Exponential probability distribution

The last examined probability distribution was exponential probability distribution. We obtained following results:

- 1005 sessions ended in correct time,
- 3995 sessions ended below then T_{ses}^{min},
- 4000 sessions ended with error.

For all tests, the minimum delay in the network was 1 [tu], the average was 1.1 [tu] and the maximum was 1.65 [tu]. The delays in the network values oscillated around 1.1 [tu]. For this reason, there were no sessions whose times would exceed the max. Only those executions in which the Intruder was unable to acquire adequate knowledge to carry out the steps ended with an error.

5.3. Encryption and decryption time simulations

Then, we performed simulations of encryption and decryption times. For this stage we determined the following assumptions:

- time of generating confidential information: $T_g = 1[tu]$,
- time of composing the message $T_c = 1[tu]$,
- minimal encryption and decryption time $T_e^{min} = T_d^{min} = 1[tu]$,
- maximal encryption and decryption time $T_e^{max} = T_d^{max} = 10[tu]$,
- minimal delay in the network $D_{min} = 1[tu]$,
- maximal delay in the network $D_{max} = 5[tu]$.
- current delay in the network $D = 5[tu]$.

We assumed a range of delay in the network values, but each step used constant value of this parameter. Current encryption and decryption times were generated according to the uniform probability distribution. Each execution of NSPK protocol was tested in the 1000 series.

We observed:

- 6596 sessions ended in correct time,
- 580 sessions ended below then T_{ses}^{min},
- 2348 sessions ended upper then T_{ses}^{max},
- 8476 sessions ended with error.

A summary of the encryption and decryption times obtained during the simulation was presented in Table 5. The minimum, average and maximum times of these operations are shown, divided into statuses obtained by the session.

5.4. Summary and discussion

After gathering knowledge about the operation of the protocol in various situations, we can modify the time conditions imposed on each step to make the protocol more secure.

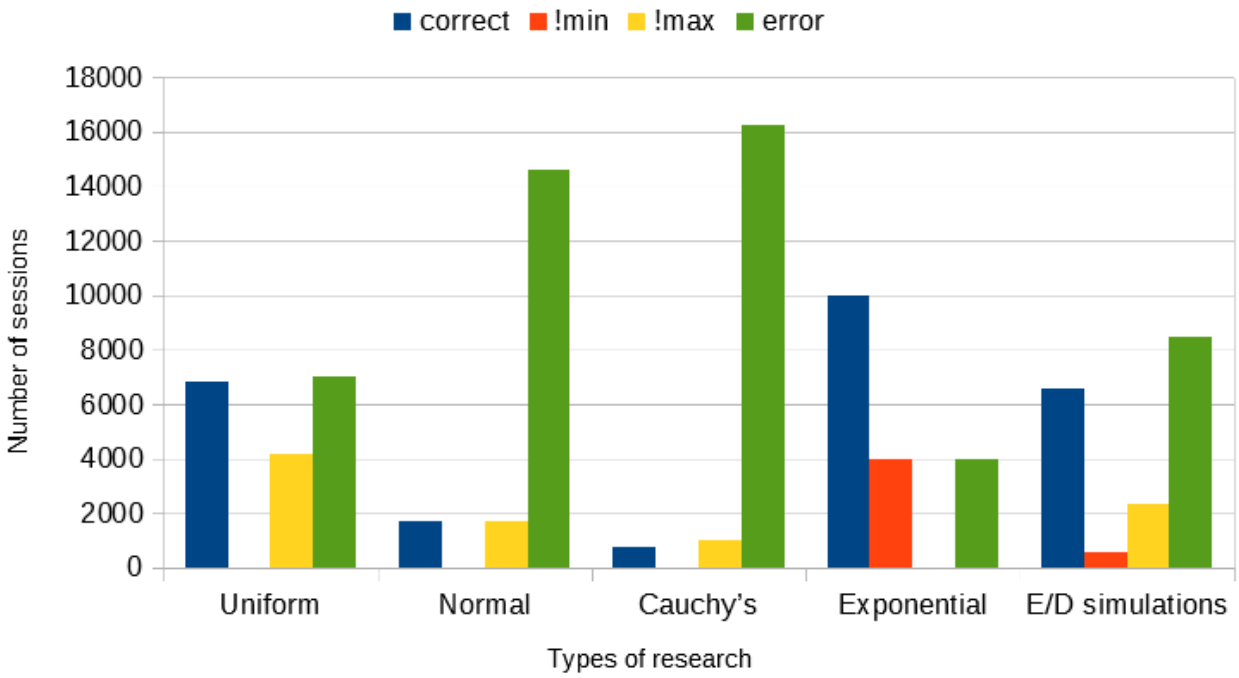

Figure 1. Summary of execution numbers.

Figure 1 presents a summary of the number of sessions of the NSPK protocol, divided into statuses obtained at individual stages of the study. Please note that the most errors occurred in the case of tests with normal and Cauchy distribution. This is because these distributions were designed to model loaded and problematic computer networks.

In turn, the uniform distribution modelled the normal operation of the computer network. In such a network, sometimes errors occur. The exponential distribution modelled a very fast computer network. In this case, sessions are carried out without problems. The resulting errors were related to the Intruder's inability to acquire knowledge.

In the case of encryption and decryption simulations, Please note that the duration of these activities also significantly affected the duration of the session. This means that if the Intruder has a very fast computer unit, Intruder may have enough time to launch the attack. It should be remembered here that all sessions were carried out with the maximum delay in the network.

6. Conclusions

In this article, we have presented a new tool for simulating the operation of security protocols and verifying their correctness. We showed how we prepare the test protocol and what time parameters we accept. Our tool enables modelling of user behaviour. We take into account their knowledge, which changes during the implementation of the tested protocols. We have also built time conditions that take into account the times of generating confidential information, the time of composing the message, the time of encrypting the message by the sender, transmission time, the time of decrypting the message by the recipient and lifetime.

Our research has shown that time has a significant impact on the correct executions of protocols and thus on the security of sensitive data. The results presented in Section 5 showed how many additional steps an Intruder can execute if too large time limits for the response are set in the computer network. In such a situation, it is necessary to analyse set time limits to prevent them. Then, the administrator should set the constraint values so that honest users have enough time to execute their operations and that an Intruder cannot acquire additional knowledge. The obtained experimental results are promising and, we would like to continue our work.

In future work, we would like to implement further time parameters for our research, use different times for encryption and decryption, take into account the higher speed of these operations for an Intruder, and lower for honest users, and combine the implemented types of simulations.

Conflict of interest

The authors declare no conflict of interest.

References

1. R. M. Needham, M. D. Schroeder, Using encryption for authentication in large networks of computers, *Commun. ACM*, **21** (1978), 993–999.
2. D. Dolev, A. C. Yao, On the security of public key protocols, *IEEE Trans. Inf. Theory*, **29** (1983), 198–208.
3. G. Lowe, An attack on the needham-schroeder public-key authentication protocol, *Inf. Process. Lett.*, **56** (1995), 131–133.
4. G. Lowe, *Breaking and fixing the needham-schroeder public-key protocol using FDR*, in Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems, TACAs '96, Springer-Verlag, London, UK, 1996.
5. M. Burrows, M. Abadi, R. Needham, A logic of authentication, *ACM Trans. Comput. Syst.*, **8** (1990), 18–36.
6. L. C. Paulson, Inductive analysis of the internet protocol TLS, *ACM Trans. Inf. Syst. Secur.*, **2** (1999), 332–351.
7. M. Kurkowski, W. Penczek, Verifying security protocols modelled by networks of automata, *Fundam. Inf.*, **79** (2007), 453–471.

8. B. Nguyen, C. Sprenger, C. Cremers, Abstractions for security protocol verification, *J. Comput. Secur.*, **26** (2018), 459–508.
9. W. Steingartner, V. Novitzka, Coalgebras for modelling observable behaviour of programs, *J. Appl. Math. Comput. Mech.*, **16** (2017), 145–157.
10. J. Perh, Z. Bilanov, W. Steingartner, J. Piatko, *Benchmark of software developed in different component models*, in 2019 IEEE 13th International Symposium on Applied Computational Intelligence and Informatics (SACI), 2019.
11. D. Galinec, W. Steingartner, V. Zebic, *Cyber rapid response team: An option within hybrid threats*, IEEE 15th International Scientific Conference on Informatics, 2019.
12. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, et al., *The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
13. M. Kacprzak, W. NabiaÅek, A. Niewiadomski, W. Penczek, A. PÅrola, M. Szreter, et al., Verics 2007-a model checker for knowledge and real-time, *Fundam. Inf.*, **85** (2008), 313–328.
14. C. Cremers, *The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols*, International Conference on Computer Aided Verification, 2008.
15. A. Casimiro, R. d Lemos, C. Gacek, *Operational Semantics and Verification of Security Protocols*, Information Security and Cryptography, Springer, 2012.
16. A. David, K. G. Larsen, A. Legay, M. Mikuionis, D. B. Poulsen, Uppaal smc tutorial, *Int. J. Software Tools Technol. Trans*, **17** (2015), 397–415.
17. B. Blanchet, Modeling and verifying security protocols with the applied pi calculus and proverif, *Found. Trends Priv. Secur.*, **1** (2016), 1–135.
18. V. Cortier, S. Delaune, J. Dreier, *Automatic generation of sources lemmas in tamarin: Towards automatic proofs of security protocols*, European Symposium on Research in Computer Security, 2020.
19. M. Kurkowski, *Formalne metody weryfikacji własności protokołów zabezpieczajacych w sieciach komputerowych*, Informatyka-Akademicka Oficyna Wydawnicza EXIT, Akademicka Oficyna Wydawnicza Exit, 2013.
20. M. Kurkowski, O. Siedlecka-Lamch, P. Dudek, *Using backward induction techniques in (timed) security protocols verification*, in Proceedings of 12th International Conference CISIM 2013, Krakow, Poland, 2013.
21. O. Siedlecka-Lamch, M. Kurkowski, J. Piatkowski, *Probabilistic model checking of security protocols without perfect cryptography assumption*, International Conference on Computer Networks, 2016.
22. S. Szymoniak, M. Kurkowski, J. Piatkowski, Timed models of security protocols including delays in the network, *J. Appl. Math. Comput. Mech.*, **14** (2015), 127–139.
23. R. Corin, S. Etalle, P. H. Hartel, A. Mader, Timed Analysis of Security Protocols, *J. Comput. Secur.* **15** (2017), 619–645.

24. O. Siedlecka-Lamch, I. El Fray, M. Kurkowski, J. Pejas, *Verification of mutual authentication protocol for mobinfosec system*, IFIP International Conference on Computer Information Systems and Industrial Management, 2015.
25. O. Siedlecka-Lamch, S. Szymoniak, M. Kurkowski, A fast method for security protocols verification, IFIP International Conference on Computer Information Systems and Industrial Management, 2019,
26. S. Szymoniak, *The impact of time parameters on the security protocols correctness*, International Conference on Computer Networks, 2018.
27. S. Szymoniak, O. Siedlecka-Lamch, M. Kurkowski, *On some time aspects in security protocols analysis*, International Conference on Computer Networks, 2018.
28. A. Kassem, P. Lafourcade, Y. Lakhnech, S. Mödersheim, *Multiple independent lazy intruders*, in 1st Workshop on Hot Issues in Security Principles and Trust (HotSpot 2013), 2013.
29. D. Basin, S. Mdersheim L. Vigan, OFMC: A symbolic model-checker for security protocols, *Int. J. Inf. Secur.*, **4** (2005), 181–208.
30. S. Mdersheim, F. Nielson, H. R. Nielson, *Lazy mobile intruders*, International Conference on Principles of Security and Trust Springer, 2013.
31. A. Grosser, M. Kurkowski, J. Piatkowski, S. Szymoniak, *ProToc–An Universal Language for Security Protocols Specifications*, Soft Computing in Computer and Information Science, 2015.
32. M. Satyanarayanan, Integrating security in a large distributed system, *ACM Trans. Comput. Syst.*, **7** (1989), 247–280.

AIMS Press