HOW TO BE ON TIME WITH SECURITY PROTOCOL?

Sabina Szymoniak

Czestochowa University of Technology (Poland)

sabina.szymoniak@icis.pcz.pl

ABSTRACT

The paper discusses a very important problem which is the verification of security protocols. Security protocols are used to secure users' communication living in the Smart Cities. Users in the cyber world could be exposed to dishonest users' actions. These users are called Intruders. Also, they could fall victim to cybercrime. Due to continuous technological development, the security of protocols should be regularly verified to confirm their correctness. Also, in the case of security protocols, time plays a significant role. It may turn out that a few seconds will allow an Intruder to acquire the appropriate knowledge to execute an attack and stole confidential data. Therefore, it is right to verify security protocols also in terms of the influence of time on their security. For this purpose, we propose a new method for security protocols verification including timed parameters and their influence on security. Our method includes analysis of encryption and decryption times, composing the message time, delays in the network and lifetime, using the specially implemented tool. Thanks to this, we can calculate the correct time protocol execution, indicate time dependencies and check the possibility of Intruder's attack. Our experimental results we present on well-known Needham Schroeder protocol example.

KEYWORDS: timed analysis, security protocols, cybersecurity, verification.

1. INTRODUCTION

The concept of smart cities primarily uses information and communication technologies to increase the interactivity of city infrastructure. The development of technology entails the improvement of everyone's quality of life. In turn, the use of modern IT technologies is associated with the problem of security of users and the entire urban society.

Ensuring security at the appropriate level is based on the security protocol (SP). The security protocol is a sequence of several steps during which authentication information is exchanged between computer network users. Unfortunately, SP's are exposed to attacks and activities of dishonest users, so-called Intruders. An intruder can eavesdrop on the communications of honest users, intercept their messages, and also use the knowledge thus acquired to conduct attacks. The activities of the Intruder entail the need to regularly check the operation and security of protocols.

Over the years, several methods have been developed for verifying security protocols and tools implementing these methods ((Dolev et al., 1983), (Burrows et al., 1989), (Lowe, 1996), (Paulson, 1999), (Armando A., et. al., 2005), (Nigam et al., 2016), (Blanchet B., 2016), (Steingartner et al., 2017), (Chadha et al., 2017), (Basin et al., 2018), (Siedlecka-Lamch O., et al., 2019)). These

methods and tools did not take into account time parameters and their impact on the security of security protocols.

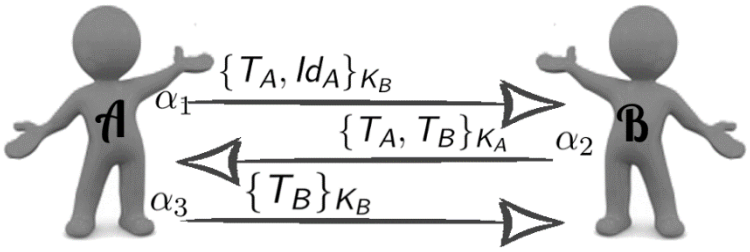
Attempts to demonstrate the influence of time on safety appeared in the works of Jakubowska and Penczka (Jakubowska et al., 2006), (Jakubowska et al., 2007). Unfortunately, this work was not continued. In turn, the model of implementation of security protocols presented in (Kurkowski, 2013) enables the generation of various versions of security protocols. We have extended this model with the mentioned time parameters. Thanks to this, we can check the duration of the session and check how time parameters have an impact on performance security. Also, we check whether the Intruder can attack the protocol for the set time parameters. In our considerations, we took into account constant and random values of time parameters.

The rest of the article is organized as follows. At the second Section we present an example of security protocol, which is Needham Schroeder Public Key protocol. We used this protocol to show the results of our research. The next Section presents our methods and materials. At the fourth Section we present our experimental results of our research. The last Section includes conclusions and plans for the future.

2. NEEDHAM SCHROEDER PUBLIC KEY PROTOCOL

As an example we will use the well-known Needham Schroeder protocol, NSPK for short (Needham et al., 1978). NSPK consists of three steps, during which two honest users (signed as *A* and *B*) try to authenticate with each other. For this purpose they exchange messages with timestamps (T_A, T_B), IDs (I_A) encrypted by theirs public keys (K_A, K_B).

Figure 1. Scheme of NSPK protocol.



Source: self-elaboration based on (Needham et al., 1978)

The timed version of NSPK protocol in Alice-Bob notation is presented in Figure 1. In the first step (α_1) *Alice* generate she's timestamp T_A and send it with she's ID to *Bob*. The message is encrypted by *Bob*'s public key K_B. In the next step (α_1) *Bob* generate he's timestamp T_B and send it with *Alice*'s timestamp to *Alice*. This message is encrypted by *Alice's* public key K_A. In the last step (α_3) *Alice* send to *Bob* his timestamp encrypted by *Bob's* public key.

Figure 2. Scheme of attack on NSPK protocol.

$$
\begin{aligned}
&\alpha_1 \quad A \to T \quad : \{T_A, I_A\}_{K_I} \\
&\beta_1 \quad T(A) \to B : \{T_A, I_A\}_{K_B} \\
&\beta_2 \quad B \to T(A) : \{T_A, T_B\}_{K_A} \\
&\alpha_2 \quad T \to A \quad : \{T_A, T_B\}_{K_A} \\
&\alpha_3 \quad A \to T \quad : \{T_B\}_{K_I} \\
&\beta_3 \quad T(A) \to B : \{T_B\}_{K_B}
\end{aligned}
$$

Source: self-elaboration based on (Lowe G., 1996)

The Figure 2 presents a scheme of the attacker's intrusion on NSPK protocol. This attack was described by Gavin Lowe in (Lowe G., 1996).

The Intruder (*Trudy*, *T*) must execute additional steps (signed by *β*), to acquire adequate knowledge to complete the main execution (signed by *α*). Communication is as follows. First, *Alice* begin a communication with *Trudy* (step α_1). Therefore, she prepare message encrypted by *Trudy's* public key according to NSPK's scheme. *Trudy* decrypt this message and encrypt it again with *Bob's* public key. This message is sent to *Bob* (step β_1). Therefore, *Trudy* knows Alice's timestamp and *Bob* thinks that *Alice* try to communicate with him. Next, *Bob* generate his timestamp and send it to *Trudy*, who impersonates *Alice*, with *Alice's* timestamp. Message from step β_2 is encrypted by *Alice's* public key. *Trudy* cannot decrypt this message because she does not know *Alice's* private key. So she send whole message to *Alice* in step α_2. *Alice* decrypt it and send to *Trudy Bob's* timestamp in step α_3. Because this message was encrypted by key K_I, *Trudy* can decrypt it and send to *Bob* his timestamp.

The effect of such protocol execution is following. *Alice* and *Bob* think that they communicated with each other, but in fact all their messages were read by *Trudy*. *Trudy* know *Alice's* and *Bob's* timestamp.

3. METHODS AND MATERIALS

To perform the full specification of the security protocol, we have extended all model definitions from (Kurkowski, 2013) by time parameters. The new model includes definitions of a set of time conditions, a protocol step, as well as the entire protocol in a timed version. In turn, the computational structure defines the current execution of the protocol and its interpretation. In the executions of the protocol, we took into account the Intruder in four models: Dolev-Yao (Dolev et al., 1983), restricted Dolev-Yao, lazy Intruder and restricted lazy Intruder. Interpretation of the protocol makes it possible to generate a set of different protocol executions. Also, the model takes into account changes in participants' knowledge during the protocol.

Also, the computational structure defines a set of time dependencies that allow to calculate the duration of a session and prepare appropriate time conditions. These dependencies are related

to message composition, step and session times, and lifetime. Dependencies include delays in the network.

For the needs of our approach, we introduced the following delays in the network values for the protocol's step: minimum (D_{min}), current (D_s) and maximum (D_{max}). Such distinction determines the range of tested values delays in the network. It is necessary to enable correct protocol execution to honest users regardless of network conditions. Also, for step time and the session time we consider similar distinction. The step time (minimal T_s^{min}, current T_s, maximal T_s^{max}) is the sum of message composition's time (T_c), encryption time by the sender (T_e), delay in the network and decryption of the message's time by the recipient (T_d). The session time (minimal T_{ses}^{min}, current T_{ses}, maximal T_{ses}^{max}) consists of all steps' times. The values of step and session times depend on used delays in the network values.

To check the influence of time parameter values on the protocol's users and its security, lifetime was established. Lifetime cannot be exceeded in any of the executed steps. If this value will be exceeded, users should know that they are communicating with the Intruder. Therefore, communication should be immediately terminated. We calculate lifetime value in one step as a sum of maximal step times of this step and next steps.

For research, we created a tool which allows verifying the timed security protocols. In the beginning, the tool loads the protocol's specification from the file. Then all potential executions of the tested protocol are combinatorically generated. In the next step, using the SAT-solver, we checked whether the generated executions are possible in reality. It is possible that during one execution the Intruder will not be able to acquire the appropriate knowledge to complete it.

Then we conducted two types of research on the loaded protocol. The first of these is the so-called time analysis. This analysis enables the determination of limits for delays in the network and lifetime for which the protocol remains secure. The second type of research is simulations. In this case, we can simulate delays in the network values and encryption and decryption times to provide a real representation of the computer network.

4. EXPERIMENTAL RESULTS

Our tests were carried out using a computer unit with the Linux Ubuntu operating system with Intel Core i7 processor, and 16 GB RAM. Also, we used an abstract time unit ([tu]) to determine the time.

The experimental results will be presented on the example of Needham Schroeder Public Key protocol. According to NSPK protocol structure, we assumed that encryption and decryption times were equal 5 time units ([tu]), time of composing the message for the first and second step were equal 2 [tu], time of composing the message for the third step were equal 1 [tu]. Also, we choose the range of delays in the network values from 1 to 10 [tu] and set constant value for the current delay in the network D=1 [tu].

Next, we calculated lifetimes for steps:

- L_1=59 [tu],
- L_2=39 [tu],
- L_3=19 [tu].

Also, we calculated minimal and maximal session time:

- T_s^{min}=32 [tu],
- T_s^{max}=65 [tu].

These values were necessary to enable and set time conditions.

Table 1. Summary of NSPK protocol's executions.

No.	Send. - Rec.	Parameters	No.	Send. - Rec.	Parameters
1	*A→B*		10	*B→I(A)*	T_a, K_a
2	*B→A*		11	*I→A*	T_i, K_i
3	*I→B*	T_i, K_i	12	*I→A*	T_b, K_i
4	*I→B*	T_a, K_i	13	*I(B)→A*	T_i, K_b
5	*I(A)→B*	T_i, K_a	14	*I(B)→A*	T_b, K_b
6	*I(A)→B*	T_a, K_a	15	*A→I*	T_i, K_i
7	*B→I*	T_i, K_i	16	*A→I*	T_b, K_i
8	*B→I*	T_a, K_i	17	*A→I(B)*	T_i, K_b
9	*B→I(A)*	T_a, K_a	18	*A→I(B)*	T_b, K_b

Source: self-elaboration

For the NSPK protocol, eighteen executions have been generated. A list of these executions was presented in Table 1. Column *Send. - Rec.* relate to protocol's participants: *A*, *B* is the honest users, *I*, *I(A)*, *I(B)* is the Intruder. *I* means Intruder who occur as a regular user, *I(A)* means intruder who impersonates user *A*, and *I(B)* means Intruder who impersonates user *B.* Column *Parameters* includes cryptographic objects, which are used by Intruder during execution. Column *No.* contains an ordinal number which was assigned in order to simplify the reference to execution. For example, execution no. 9 take place between honest user *B* and Intruder who impersonates user *A*. In this case, the Intruder uses the timestamp of user *A* and also his public key.

Table 2. Timed analysis of attacking execution in [tu].

α step	β step	T_e	T_c	D	T_d	T_s	T_{ses}	Result
α_1		4	2	1	4	11	11	ok
	β_1	4	2	1	4	11	22	ok
	β_2	4	2	1	0	7	29	ok
α_2		0	2	1	4	(7) 25	36	ok
α_3		4	1	1	4	10	46	ok
	β_3	4	1	1	4	(10) 27	56	$T_3>L_3$

Source: self-elaboration

Let's analyze the attacking execution. We analysed an interlacing of 7 (α-execution) and 11 (*β*-execution) executions, which reflects attack included in Figure 2. The timed analysis was presented in Table 2. This Table consist of nine columns. First of them is α-step which is related to the steps of basic execution. The second column is called *β*-step. Here will be assigned steps from additional execution. Next six columns are connected to time parameters. These are encryption time ($T_{e)}$), composition time (T_c), delay in the network value (D), decryption time (T_d), current step time (T_s) and current session time (T_{ses}). The last column is called Result. Here we assign our comment about current step.

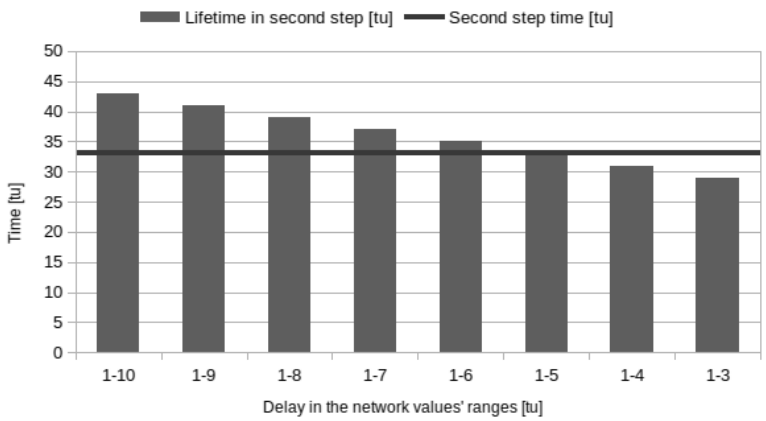
Therefore, each step consists of encryption time, composition time, delay in the network value and decryption time. Because Intruder did not have enough knowledge to execute α_2-step, he must establish *β*-execution and execute additional step from it. Steps times from additional steps were added to basic step time and also to the session time. Therefore, α_2-step includes β_1 and β_2 times. Also, β_3-step includes α_2 and α_3 times.

In the step α_2 and β_2 there were no included encryption time. The Intruder did not perform such operations, because he has not appropriate knowledge to decrypt the message from β_2, therefore he sends whole this message to *Bob* in the step α_2.

Please note that if Intruder will not end *β*-execution, the α-execution will be ended in the correct time, including additional steps' times and Intruder will know *A's* and *B's* timestamps. Therefore, the attack on this protocol is possible for assumed values of the time parameters.

Next, we calculated how changes in the delay in the network range would affect protocol security. We increased the maximum delay value in the network by 1 [tu] and checked how long the duration of the second step would be. Thanks to this, we will be able to determine what limit should be set for this step.

Figure 3. Changes in the delay in the network range.



Source: self-elaboration

Our results were shown in Figure 3. The setting the upper limit of delay in the network values to 4 [tu] protects the protocol. In such a situation, a lifetime set in the second step will end the communication and the attack on NSPK protocol is not possible.

In the next step, we performed simulations of Needham Schroeder Public Key protocol's executions. We used the randomly generated the current delay in the network values. We used normal, uniform, Cauchy's, Poisson's and exponential probability distributions to generate these values. The tool also allows random selection of values out of the accepted range to model the real work of a computer network.

We made the following assumptions:

- encryption and decryption times were equal 2 [tu],
- time of composing the message for all steps was equal 1 [tu],
- the range of delays in the network values from 1 to 10 [tu].

Next, we calculated new lifetimes for steps, minimal and maximal session times:

- L_1=44 [tu],
- L_2=29 [tu],
- L_3=14 [tu],
- T_s^{min}=17 [tu],
- T_s^{max}=44 [tu].

Table 3. NSPK executions ended in the correct session time.

No.	Session time [tu]			**Average delay in the network [tu]**
	min	**avg**	**max**	
1	18.4	30.98	43.3	5.67
2	19.1	30.83	43	5.61
3	17.4	29.24	41.8	5.41
4	40.7	42.62	43.9	3.04
7	18	29.73	41.3	5.58
8	39.6	42.31	43.6	3.1
11	17	29.72	42.2	5.57
12	39.8	42.14	43.8	3.1
15	17.4	29.77	41.7	5.71
16	38.3	41.92	43.8	2.89

Source: self-elaboration

We carried out 18,000 test series for each probability distribution. In Table 3, we presented minimal, average and maximal values of session time for several executions of Needham Schroeder Public Key protocol including delay in the network values generated according to the uniform probability distribution. Also, we presented the average delay in the network values. These sessions ended in the correct session time.

Also, we assumed two specific situations. First of them was when the current session time was lower then minimal session time (T_s^{min}). This means that Intruder may send cryptograms which were in his knowledge set. The Intruder did not encrypt or decrypt messages and also he did not generate any object, so these times were not added to session time. Session time (T_s) was lower then minimal session time (T_s^{min}).

Table 4. NSPK executions ended below the minimal session time.

No.	Session time [tu]			**Average delay in the network [tu]**
	min	**avg**	**max**	
3	16.01	16.31	16.98	1.1
7	16.02	16.3	16.97	1.2
10	16.03	16.49	16.49	1.09
11	16.04	16.3	16.99	1.1
15	16.02	16.31	16.98	1.1
18	16.03	16.51	16.99	1.09

Source: self-elaboration

In Table 4, we presented minimal, average and maximal values of session time for several executions of Needham Schroeder Public Key protocol including delay in the network values generated according to the exponential probability distribution. Also, we presented the average delay in the network values in each execution. These sessions ended in below the minimal session time. In these exeutions Intruder used his cryptographic objects and also resent whole ciphertext received from honest users. Please note that, the average delay in the network values were between 1.09 [tu] and 1.2 [tu].

The second specific situation was when the current session time was upper then maximal session time. This means that Intruder must execute additional steps to get knowledge. Additional steps' times affect the current step and session time. In this case, the execution ended incorrectly (exceeding the maximum session time), while the time conditions imposed on each step have been preserved.

Table 5. NSPK executions ended upper then the maximal session time.

No.	Session time [tu]			**Average delay in the network [tu]**
	min	**avg**	**max**	
1	44.02	50.40	76.81	12.98
2	44.01	51.81	77.61	12.74
3	44.02	51.75	70.47	17.84
4	48.08	75.26	105.99	8.81
7	44.05	52.15	71.12	21.66
8	46.05	68.58	90.61	7.19
11	44.03	52.19	75.57	13.06
12	47.81	73.72	110.78	8.37
15	44.16	52.38	72.02	13.13
16	46.48	66.0	95.99	7.1

Source: self-elaboration

In Table 5, we presented minimal, average and maximal values of session time for several executions of Needham Schroeder Public Key protocol including delay in the network values generated according to the normal probability distribution. Also, we presented the average delay in the network values in each execution. These sessions ended in below the maximal session time. Please note that, the average delay in the network values were greater then in case of exponential probability distributions. Also, we observed values out of adopted range.

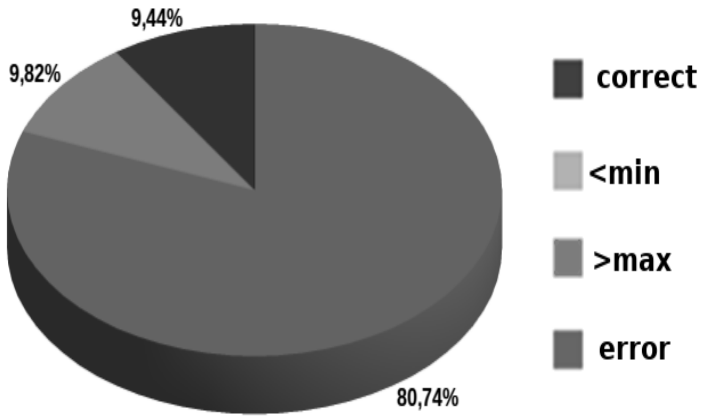
To present summary of our research we included following designations on the charts:

- *correct*, which is designated to the session ended between minimal session time (T_s^{min}) and maximal session time (T_s^{max}),
- *<min*, which is designated the session that were ended lower then minimal session time (T_s^{min}),
- *>max*, which is designated to the session that were ended upper then maximal session time (T_s^{max}),
- *error*, which is designated the session that were ended because one of the time conditions was not met.

We presented the percentage summary of the number of NSPK protocol executions ended with assumed statuses. Please note that there was no situation in which session ended upper then maximal session time (T_s^{max}).

On Figure 4, we presented a summary of the results for the Needham Schroeder Public Key protocol using a normal probability distribution.

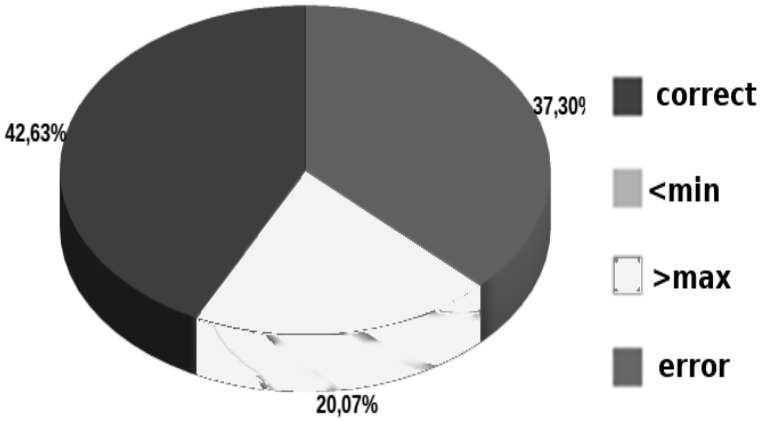
Figure 4. Summary of the results for the NSPK protocol using normal probability distribution.



Source: self-elaboration

On Figure 5, we presented a summary of the results for the Needham Schroeder Public Key protocol using a uniform probability distribution. Please note that there was no situation in which session ended upper then minimal session time (T_s^{min}).

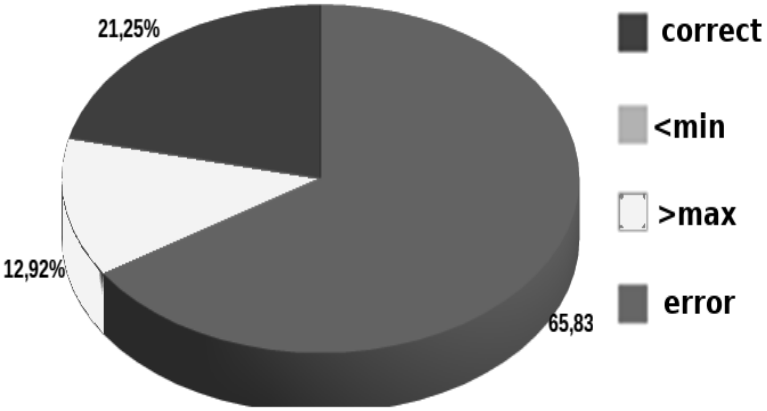
Figure 5. Summary of the results for the NSPK protocol using uniform probability distribution.



Source: self-elaboration

On the Figure 6, we presented summary of the results for the Needham Schroeder Public Key protocol using Poisson's probability distribution. Please note that there were no situation in which session ended lower then minimal session time *(T_s^{min}).* and there were a lot of error situations.

Figure 6. Summary of the results for the NSPK using Poisson's probability distribution.

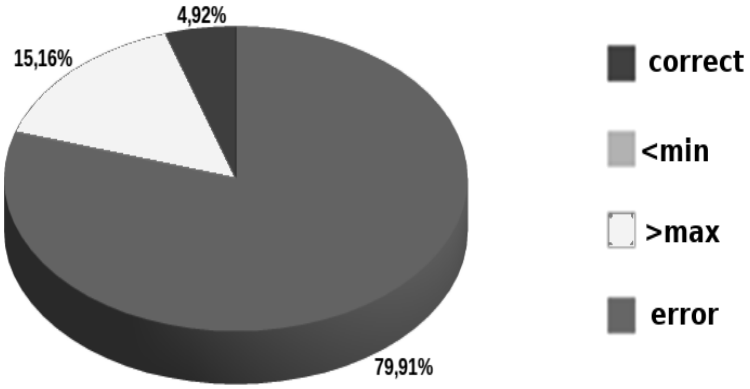


Source: self-elaboration

On Figure 7, we presented a summary of the results for the Needham Schroeder Public Key protocol using Poisson's probability distribution. Please note that there was no situation in which session ended lower then minimal session time *(T_s^{min})* and there were a lot of error situations.

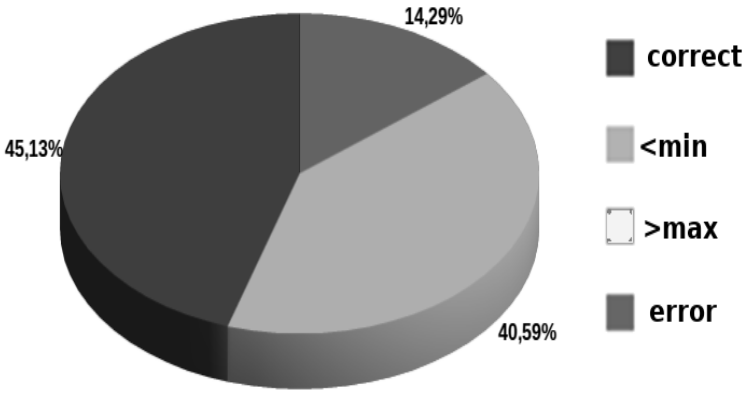
On Figure 8, we presented a summary of the results for the Needham Schroeder Public Key protocol using an exponential probability distribution. Please note that there was no situation in which session ended upper then maximal session time.

Figure 7. Summary of the results for the NSPK using Cauchy's probability distribution.



Source: self-elaboration

Figure 8. Summary of the results for the NSPK using exponential probability distribution.



Source: self-elaboration

The obtained results showed various aspects of computer network operation. We observed that usage of uniform probability distribution shows the natural operation of the network. The usage of normal probability distribution reflects the real operation of the network. The usage of Cauchy and Poisson probability distributions suggest a busy network that very often has problems. The analyze of the results obtained for the exponential probability distribution it should be stated that this distribution illustrates the fast network.

Also, we try to check what is the influence of encryption algorithms' speed and computing power. For this purpose, we perform simulations Needham Schroeder Public Key protocol's executions using randomly generated values of encryption and decryption time, according to a uniform probability distribution.

Again, we carried out 18,000 test series. We observed that for encryption and decryption times close to 1 [tu], the attacking execution ended correctly. This means that if Intruder has great computing power, his encryption and decryption times could be short and he can successfully perform an attack on the protocol.

5. CONCLUSION

This paper discussed the problem of security protocols' verification. SPs are widely used in smart cities to secure users communication. For this reason, security protocols verification is important to check if they provide an appropriate level of security.

We presented a new approach to this issue. In our research, we take into account the following time parameters: encryption and decryption times, composing the message time, delays in the network and lifetime. These parameters were used to calculate the correct protocol's execution time and designate time dependencies. The imposed dependencies should protect prevent loss of confidential information. We researched by timed analysis and simulation of delays in the network and simulations of encryption and decryption times.

We observed that time has a huge impact on protocols' security. Badly selected time dependencies could allow Intruder to perform additional actions to steal the data and threaten the security of the smart city. During our research, we analyzed how delays in the network range affect Intruder's capabilities. We took into account constant and random values of time parameters. Observed results showed that if delays in the network range will be to extensive, it will not be secure for honest users because Intruder could have enough time to compromise the protocol.

Also, we observed how the selected probability distributions illustrated the operation of a computer network. Our next research will be focused on further parameters, which may have an impact on communication security in smart cities.

ACKNOWLEDGEMENTS

The project financed under the program of the Polish Minister of Science and Higher Education under the name "Regional Initiative of Excellence" in the years 2019 - 2022 project number 020/RID/2018/19, the amount of financing 12,000,000.00 PLN.

REFERENCES

Armando A., et. al. (2005). The AVISPA tool for the automated validation of internet security protocols and applications, In: Proc. of 17th Int. Conf. on Computer Aided Verification (CAV'05), vol. 3576 of LNCS, pp. 281–285, Springer

Basin D., Cremers C., Meadows C. (2018). Model Checking Security Protocols, in Handbook of Model Checking, Springer International Publishing

Blanchet B. (2016). Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif, Foundations and Trends in Privacy and Security, vol. 1(1-2) pp.1–135

Burrows M., Abadi M., Needham R. (1989). A Logic of Authentication, In: Proceedings of the Royal Society of London A, vol. 426

Chadha R., Sistla P, Viswanathan M. (2017). Verification of randomized security protocols, Logic in Computer Science

Dolev D., Yao A. (1983). On the security of public key protocols. In: IEEE Transactions on Information Theory, 29(2)

Jakubowska G., Penczek W. (2006). Modeling and Checking Timed Authentication Security Protocols, Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'06), Informatik-Berichte 206(2)

Jakubowska G., Penczek W. (2007). Is your security protocol on time?, In Proc. Of FSEN'07, volume 4767 of LNCS, Springer-Verlag

Kurkowski M. (2013). Formalne metody weryfikacji wlasności protokolow zabezpieczajacych w sieciach komputerowych, in polish, Exit, Warsaw

Lowe G. (1996). 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, pages 147–166, London, UK, 1996. Springer-Verlag.

Needham R. M., Schroeder M. D. (1978). Using encryption for authentication in large networks of computers. Commun. ACM, 21(12)

Nigam V., et. al (2016). Towards the Automated Verification of Cyber-Physical Security Protocols: Bounding the Number of Timed Intruders, Computer Security – ESORICS 2016", Springer International Publishing

Paulson L. (1999). Inductive Analysis of the Internet Protocol TLS, ACM Transactions on Information and System Security (TISSEC), vol 2 (3)

Siedlecka-Lamch O., et. al (2019) A fast method for security protocols verification, Computer Information Systems and Industrial Management, Springer

Steingartner W., Novitzka V. (2017). Coalgebras for modelling observable behaviour of programs, In: Journal of applied mathematics and computational mechanics. 16(2)