USING A SECURITY PROTOCOL TO PROTECT AGAINST FALSE LINKS

Sabina Szymoniak

Czestochowa University of Technology (Poland)

sabina.szymoniak@icis.pcz.pl

ABSTRACT

The article discusses the problem of internet users protection against false URLs. During the coronavirus pandemic, we realized the value and benefits of remote work. Also, training began to be conducted remotely. Available webinars or training made it possible for us to broaden our knowledge. Organizers of such activities can send or share the proposals for participation via social networks or e-mail messages. The development of remote activities contributed to an increase in cybercrimes. A specially crafted script may be hidden under a seemingly safe URL for an online meeting. We propose a new security protocol for users authentication and protection against false links. The protocol consists of two parts: initial and verification. In the initial part, the user must agree on his unique identifier with the trusted Distribution Center. The event organizer must perform a similar action. In the verification part, the meeting participant will verify the correctness of the connection URL. This protocol implements AAA-logic. We examined the security of our protocol using a verification tool. This tool enables verification of security protocols, including various time parameters. The results obtained are promising. Further, we plan to develop the proposed protocol and prepare a system that will use it.

INTRODUCTION

The coronavirus pandemic has shown the value and benefits of remote work. During the binding restrictions, people had to find themselves in the new reality. They had to keep working and performing their official duties. Also, children and other learners moved to the virtual world to continue their education. As time passed, scientific conferences and training began to be conducted remotely too. Thanks to available webinars or training, we can broaden our knowledge. Organizers of such activities very often send and share proposals for participation via social networks or e-mail messages.

Unfortunately, with the development of remote activity, an increase in cybercrimes can also be observed. People who respond to an invitation to a webinar or training may also be exposed to criminals. A specially crafted script may be hidden under a seemingly safe URL for an online meeting. Running this script after clicking on a link can cause enormous damage. The damages include the installation of malware on our device or the hijacking of login data by cybercriminals.

Security is an essential element of everyone's life. There are two solutions to protect against such situations. The first is the detection of false links, using artificial intelligence methods. In this case, there are many methods to accomplish detection which URL contains deceptive content. Yang et al. (Yang et al., 2020) observed that vulnerabilities in recommender systems may encourage deliberate manipulation by malicious users. Also, Lai et al. (Lai et al., 2020) looked at the impact of social recommendation systems on the presence of malicious URLs on social networks. In (Song et al., 2020), the authors presented a novel method of detecting malicious code in JavaScript, based on deep learning. Baccouche et al. (Baccouche et al., 2020) have proposed an LSTM model that will identify mischievous text regardless of the source. Also, in (Ispahany et al., 2020), (Xiao et al., 2020) and (Patel et al., 2020), we can find other methods to the detection of false links.

The second solution is the use of security protocols. These protocols are short algorithms that make it possible to achieve security goals. The two most meaningful goals are mutual user authentication or the exchange of confidential information. Over the past decades, many different security protocols have been developed. It is worth mentioning the Needham Schroeder protocol (Needham et al., 1978), which has been used for many years for user authentication.

With the advent of new security protocols, various methods of protocols verification and tools enabling them appeared. Worth mentioning are methods and tools presented in ((Dolev et al., 1983), (Burrows et al., 1989), (Lowe, 1996), (Paulson, 1999), (Armando A. et al., 2005), (Nigam et al., 2016), (Blanchet B., 2016), (Chadha et al., 2017), (Basin et al., 2018), (Siedlecka-Lamch et al., 2019)). Also, in the case of security protocols, it is worth mentioning the following papers (Steingartner et al., 2019), (Galinec et al., 2019), (Radaković et al., 2018), (Čibej et al., 2019),(Piątkowski, 2020).

In this article, we propose a new security protocol. We can use our protocol for users authentication and protection against false links. The protocol consists of two parts: preparatory and verification. In the preparatory part, the user must agree on his unique identifier with the trusted Distribution Center. Also, the event organizer must perform a similar action. In the verification part, the user (who wants to participate in any event) will verify the correctness of the URL used to make the connection.

Our protocol aims to implement AAA (Authentication, Authorization, Accounting) logic. This logic relates to a dedicated security framework used to mediate network and application access. Authentication describes how a user is identified to a network or application. Authorization is linked to the policy enforcement process for users. Accounting records session statistics and user information. Thanks to this solution, it is possible to control access to resources, including audit rules. Only authorized users will have access to the network, resources or applications (Galinec et al., 2019), (Steingartner et al., 2021).

The rest of the article is organized as follows. In the second Section, we present Related Works. The next Section presents our new security protocol. In the fourth Section, we present our experimental results. The last Section includes conclusions and plans for the future.

RELATED WORKS

One of the significant security protocols is Needham Schroeder Public key protocol (Needham et al., 1978). This protocol aims to mutual users authentication. In (Lowe, 1996), Gavin Lowe indicated a simple way to attack NSPK. Also, the worth mentioning about Wide Mouthed Frog protocol (Burrows et al., 1989). This protocol aims to the distribution of a new shared symmetric key. WMF is vulnerable to a replay attack.

Next to security protocol is MobInfoSec (Siedlecka-Lamch et al., 2019), (Siedlecka-Lamch, 2020). This protocol enables encryption and sharing of confidential information. It can be used by a group of connected users of mobile devices. Also, we use security protocols during voice calls, video calls, and chat instant messaging. Signal protocol is an example of this usage. This cryptographic protocol can be used to provide end-to-end encryption. Another application of security protocols is in Wireless Sensor Networks (Maheswari et al., 2021).

Also, security protocols can be used in IoT systems and Wireless Sensors Networks. In (Ko et al., 2021) authors proposed a security protocol for securing communication between Unmanned Aerial Vehicles. This protocol provides confidentiality and non-repudiation, which are essential for secure military communication. In (Kwon et al., 2021) suggest a secure and lightweight mutual authentication protocol for Wireless Sensors Networks. This protocol provides mutual authentication and forward secrecy. In (Moreno-Cruz et al., 2020), the authors provide a new security protocol that complies with the Wireless Communication Protocol standard. This protocol takes into account the cross-layer principle to use power consumption to dynamically adapt its operation, which goes from a pure effort to near real-time.

A NEW SECURITY PROTOCOL

The protocol we propose used to establish unique user identifiers and authenticate them during virtual meetings. There are four users in it. The first one is the meeting participant. He is a user who has a link to the event and wants to participate in it. The other user is the meeting organizer, that is, the user who is preparing the event. The third participant is the Distribution Center. It is an application that is responsible for the distribution of unique user identifiers. The last participant in the protocol is the Authentication Center, which is also an application. This application is responsible for verifying participants in virtual events.

The main goals of the proposed protocol are:

- the reconciliation of unique user identifiers,
- the mutual authentication of users,
- the distribution of a symmetric key.

Figure 1. The first part of the proposed security protocol.

$$
\begin{array}{lll}
\alpha_1 & A \rightarrow DC: & \{i(A), T_A\}_{K_A^-} \\
\alpha_2 & DC \rightarrow AC: & \{\#hash(\{UID_A, i(A)\}_{K_A^+}), \\
 & & T_{DC}^{A}\}_{K_{DC-AC}} \\
\alpha_3 & AC \rightarrow A: & \{\#hash(\{UID_A, i(A)\}_{K_A^+}), \\
 & & T_{DC}^{A}\}_{K_A^+} \\
\alpha_4 & A \rightarrow AC: & \{T_{DC}^{A}\}_{K_{AC}^+}
\end{array}
$$

Source: self-elaboration

Our protocol consists of a preparatory and verification part. In the first part, users establish their unique user identifiers from the Distribution Center. During the second part of the protocol, users can verify the identity of the other user. A potential meeting participant can validate the meeting URL and, therefore, the identity of the organizer. On the other hand, the meeting organizer can verify the identity of a potential event participant. After correct verification, participants agree on a symmetric key that will be used for their further communication.

Figure 1 shows the syntax of the proposed protocol first part in the *Alice-Bob* notation. By *A*, we marked the user who wants to participate in the event or is its organizer. *DC* stands for Distribution

Center that generates and sends individual user identifiers. We marked the Authentication Center, which confirms the identity of the meeting participants, via *AC*.

The first part of this protocol consists of four steps. In the first step, the user sends a request to Distribution Center to generate an individual user identifier for him. This step should be performed by both the meeting organizer and the participant. The message includes a text user identifier and a timestamp generated by the user. For the meeting participant, the text identifier will contain his name and surname. For the meeting organizer, the text identifier will contain the URL of the event. The user must encrypt this message with his private key.

In the second step, the Distribution Center must generate a numeric, unique user identifier (*UID_A*) and a timestamp generated by Distribution Center that will certify its validity. Then, both numeric and text identifiers are encrypted with the meeting participant's public key. The resulting Distribution Center ciphertext is subjected to a hash function and then added to the message with the generated special timestamp. The resulting message is encrypted with a symmetric key shared between the Distribution Center and Authentication Center and sent to AC.

Figure 2. The second part of the proposed security protocol

$$\begin{array}{lll}
\alpha_1 & A \to AC: & \{\#hash(\{UID_A, i(A)\}_{K_A^+}), \\
 & & T_{DC}^{A}, i(B)\}_{K_{AC}^+} \\
\alpha_2 & AC \to A: & \{\{\#hash(\{UID_B, i(B)\}_{K_A^+}), T_{AC}^{B}\}_{K_A^+}, \\
 & & \{\#hash(\{UID_A, i(A)\}_{K_B^+}), T_{AC}^{A}\}_{K_B^+}\}_{K_A^+} \\
\alpha_3 & A \to B: & \{\#hash(\{UID_A, i(A)\}_{K_B^+}), T_{AC}^{A}\}_{K_B^+} \\
\alpha_4 & B \to A: & \{T_{DC}^{A}, T_{DC}^{B}, K_{AB}\}_{K_A^+}
\end{array}$$

Source: self-elaboration.

After decrypting the message, Authentication Center saves the meeting participant's data in its database. Each such entry will help AC in user authentication. Then the Authentication Center sends to A an identical message as in the previous step. Authentication Center encrypts this message with public key A (step 3).

After decrypting the messages from step three, A saves its data. Then, A constructs a message containing the timestamp generated by DC. A encrypts this message with the Authentication Center public key and forwards it to Authentication Center (fourth step). Thanks to this, A confirms its identity with AC and DC.

Figure 2 shows the syntax of the proposed protocol second part in *Alice-Bob* notation. Compared to the first part of the protocol, a new participant appears here, denoted as B. We can consider this part of the protocol in two ways, depending on the user's role in initiating the protocol. If A is a potential meeting participant, then B is the organizer. In that case, A checks the validity and validity of the URL. Conversely, A, as the meeting organizer, verifies the identity of the event participant.

In the first step of this part of the proposed protocol, A want to verify the identity of user B. He sends a request to the Authentication Center. This message contains A's unique identifier (hashed and encrypted with A public key) and a timestamp generated by Distribution Center for A and text identifier B. The text identifier may be either a URL address or the name and surname of the participant. This situation depends on the role of A in the protocol execution. A message prepared in this way encrypts the AC with the public key and sends it to Authentication Center.

Before the execution of the second step, the Authentication Center checks the identity of A. If such a user is in the AC database, it verifies the identity of B. After both correctly verified, AC constructs a message for A. This message consists of two ciphertexts. The first one contains B's unique identifier and its timestamp. This ciphertext is encrypted with the public key A. The second ciphertext contains A's unique identifier and his timestamp. This ciphertext is encrypted with the public key B. This means that the first part of the complex message is intended for A and only A can decrypt it, while the second part is intended for B, and only B can decrypt it. The entire message is encrypted with A's public key and sent to him.

In case of incorrect verification of one of the users, the Authentication Center sends ciphertexts filled with zeros. Thus, the user will know that the URL of the meeting is false, and the organizer will know that the user does not exist.

In the third step of the proposed protocol, A sends to B the second ciphertext of the earlier message. After reading it, B learns A's unique identifier and its timestamp. He may proceed to confirm his identity to B.

In the last step, B sends a message to A containing the timestamps of both users. Additionally, it generates a symmetric key and puts it in the message. The generated key will be used in further communication of these users. He encrypts this message with A's public key.

EXPERIMENTAL RESULTS

We then conducted a preliminary security study of our protocol. We used the model and tools described in (Szymoniak, 2010), (Szymoniak, 2021). Referring to the mentioned methodology, we conducted a series of tests related to the analysis of protocol execution times and time simulations of the protocol's operation in a computer network. The analyzes take into account the presence of an Intruder (Dolev et al., 1983). The intruder is a dishonest user who wants to intercept the confidential data of other users. We performed the protocol analysis using a computer with Linux Ubuntu operating system, Intel Core i5 processor and 16 GB RAM. We used an abstract time unit ([tu]) to determine the time.

According to the mentioned methodology of research, we used the following values:

- minimum (D_{min}), current (D) and maximum (D_{max}) delay in the network value,
- minimum (T_s^{min}), current (T_s) and maximum (T_s^{max}) step time,
- message composing time (T_c),
- encryption time (T_e),
- decryption time (T_d),
- time of generating confidential information (T_g),
- minimum (T_{ses}^{min}), current (T_{ses}) and maximum (T_{ses}^{max}) session time.

The distinction between minimum, current and maximum values is significant due to the specificity of the research. Thanks to this, we can determine the range of the tested network delay values and then indicate the specificity of the Intruder's behaviour in various time aspects. Also, we took into account the difference between symmetric and asymmetric encryption and decryption. Symmetric algorithms are faster than asymmetric, so we used different time value for both operations.

For the purposes of time analysis, we made the following assumptions:

- D_{min} = 1 [tu],
- D_{max} = 4 [tu],
- (symmetric) T_e = 4 [tu],
- (symmetric) T_d = 4 [tu],
- (asymmetric) T_e = 6 [tu],
- (asymmetric) T_d = 6 [tu],
- T_g = 1 [tu],
- T_c = 1 [tu].

Table 1. Summary of operations during first part of our protocol.

Step	**G**	**C**	**E**	D	!E	Min	Max
1	+	+	A	+	A	15	18
2	+,+	+	A,S	+	S	19	21
3	-	+	A	+	A	14	17
4	-	+	A	+	A	14	17

Source: self-elaboration

Table 1 shows the summary of operations executed during the first part of our protocol. Column **Step** contains step numbers. Columns **G**, **C** and **D** contain information about the occurrence of operations in the current step: message composing, generating confidential information and delay in the network respectively. Designation **+** means that such operation occurs in the current step, and designation **–** means that such operation does not occur in the current step. In this step, there DC generates confidential information two times, so we included T_g two times in our calculations. Columns **E** and **!E** contains information of algorithms used for encryption or decryption. Please note that in the second step of the first protocol part there DC encrypts two times. It uses both, asymmetric and symmetric encryption. In this same step, the receiver decrypts only symmetric. Columns **Min** and **Max** contain information of minimal and maximal step time.

Next, we calculated lifetimes for steps of our protocol first part:

- L_1= 73 [tu],
- L_2= 55 [tu],
- L_3= 34 [tu],
- L_4= 17 [tu].

Also, we calculated minimal and maximal session time:

- T_s^{min}= 73 [tu],
- T_s^{max}= 61 [tu].

Table 2. Summary of operations during second part of our protocol.

Step	**G**	**C**	**E**	D	!E	Min	Max
1	-	+	A	+	A	15	18
2	-	+	A,A	+	A	20	23
3	-	-	-	+	A	7	10
4	+	+	A	+	A	15	18

Source: self-elaboration

Table 2 shows the summary of operations executed the during second part of our protocol. Please note that there are two asymmetric encryption and only one asymmetric decryption. In the third step, user A sends a message which he received in the earlier step. So he does not generates confidential information and composes and encrypts the message in this step.

Next, we calculated lifetimes for steps of our protocol second part:

- L_1= 69 [tu],
- L_2= 51 [tu],
- L_3= 28 [tu],
- L_4= 18 [tu].

Also, we calculated minimal and maximal session time:

- T_s^{min} = 57 [tu],
- T_s^{max}= 69 [tu].

These values were necessary to enable and set time conditions.

For our protocol, the tool described in (Szymoniak, 2021) and (Szymoniak et al., 2021) generated seventy-two different executions. These executions were generated combinatorically. The difference between particular executions is the order users appear and the objects used by the Intruder. The generated executions can be divided into three types. The first is fair execution, in which only honest users are present. The second type of executions are executions with an Intruder who does not impersonate any user. The third type is executions, in which the intruder impersonates one of the honest users.

Next, we performed simulations of our protocol executions. For this part of the research, we used the randomly generated current delay in the network values. These values were randomly generated according to normal, uniform, Cauchy's and exponential probability distributions. We choose these probability distributions to model the real work of a computer network. Also, mentioned tool allows the generation of values out the accepted range <D_{min}, D_{max}>.

We will present the simulations of our protocol on normal and uniform probability distributions example. All executions were tested in thousand series. The simulations assume that execution can end with one of four statuses. The first status is *correct*. Here execution ends between T_s^{min} and T_s^{max}. The second status is !*min*. Here execution ends below T_s^{min} while the time conditions are met. The third status is !*max*. Here execution ends above T_s^{max} while the time conditions are met. The last status is *error*. Here execution ends with the failure to meet the time conditions.

For research with the first part of our protocol and normal probability distribution, executions ended with *correct*, *!max* end *error* statuses. We observed that only twenty executions were possible. This set consisted of six executions of the first type (numbered 1-6), fourteen executions of the second type (numbered 7-20). There were no executions of the third type, so the tool did not find an attack on our protocol.

Table 3 presents simulations results for these executions ended in the correct session time. We used the current delay in the network value generated according to a normal probability distribution. We present the average session time and the average delay in the network value. Please note that the session time depends on protocol execution. If Intruder does not need to get knowledge from an additional step, the session time is lower, and the execution can end with the correct session time.

Table 3. Executions ended in the correct session time.

No.	Average session time [tu]	**Average delay in the network [tu]**
1	61.61	2.90
2	61.94	5.64
3	61.96	4.06
4	62.43	4.05
5	62.52	4.27
6	62.91	4.87
7	63.04	3.55
8	63.51	4.35
9	64.22	3.17
10	64.42	3.51
11	64.52	4.96
12	64.72	4.34
13	65.79	4.25
14	65.82	5.63
15	65.97	5.22
16	66.62	5.88
17	66.72	4.61
18	66.77	5.43
19	67.01	5.61
20	67.05	15.61

Source: self-elaboration

Table 4 presents simulations results for possible executions ended above the correct session time. Also, here we checked the first part of our protocol with the current delay in the network value generated according to a normal probability distribution. Obtained results showed that additional steps executed by Intruder can increase the session time while the time conditions are met.

Table 4. Executions ended above correct session time.

No.	Average session time [tu]	**Average delay in the network [tu]**
1	82.03	10.20
2	82.04	10.20
3	82.05	10.21
4	82.05	10.01
5	82.06	10.61
6	82.06	10.21
7	82.07	10.01
8	82.08	10.61
9	82.11	10.02
10	82.13	10.22
11	82.14	17.47
12	82.14	10.02
13	82.15	10.62
14	82.17	10.03
15	82.22	10.24
16	82.23	10.04
17	82.23	10.04
18	82.25	10.05
19	82.28	14.35
20	82.30	10.45

Source: self-elaboration

Next, we perform research with a uniform probability distribution for the second part of our protocol. Similarly, for this stage of research, executions ended with *correct*, *!max* end *error* statuses. Also, we observed that only twenty executions were possible. This set consisted of six executions of the first type (numbered 1-6), fourteen executions of the second type (numbered 7-20).

Table 5 presents simulations results for these executions that ended in the correct session time. We used the current delay in the network value generated according to a uniform probability distribution. We observed a lower current delay in the network values than in the case of a normal probability distribution. These values affect session times and involve the Intruder to execute additional steps.

Table 5. Executions ended in the correct session time.

No.	Average session time [tu]	**Average delay in the network [tu]**
1	61.8	1.76
2	63.8	2.16
3	65.4	2.48
4	65.6	2.52
5	66.1	2.62
6	66.5	2.7
7	66.6	2.72
8	66.9	2.78
9	67.1	2.82
10	67.3	2.86
11	67.3	2.86
12	67.6	2.92
13	67.7	2.94

14	67.7	2.94
15	67.7	2.94
16	67.8	2.96
17	68	3
18	68	3
19	68.1	3.02
20	68.2	3.04

Source: self-elaboration

Table 6 presents simulations results for possible executions ended above the correct session time. We used the current delay in the network value generated according to a uniform probability distribution. We observed that session times were greater than maximal session time less than 1 [tu].

Table 6. Executions ended above correct session time.

No.	Average session time [tu]	**Average delay in the network [tu]**
1	69.1	5.94
2	69.1	7.2
3	69.2	8.76
4	69.2	6.74
5	69.2	5.5
6	69.3	6.22
7	69.3	6.74
8	69.4	6.56
9	69.4	6.42
10	69.4	6.2
11	69.5	7.04
12	69.5	6.38
13	69.5	7.3
14	69.5	7.22
15	69.6	7
16	69.6	7.66
17	69.6	5.74
18	69.6	6.94
19	69.6	7.68
20	69.7	7.1

Source: self-elaboration

CONCLUSION

This paper discussed the new security protocol for protection against false links. Security protocols are widely used for users’ protection in the virtual world. Their use gained importance during the COVID-19 pandemic, where many aspects of our lives had to move to the Internet. In the era of the development of cyber attacks, there was also a need to strengthen the security of user communication. This is where security protocols come in to help.

We presented the new security protocol which aims to users authentication and protection against false links. The proposed protocol consists of two parts: preparatory and verification. In the first part, the user must agree on his unique identifier with the trusted Distribution Center. Also, the event organizer must perform a similar action. In the second part, the user (who wants to participate in any event) will verify the correctness of the URL used to make the connection.

We used the tool described in (Szymoniak, 2021) and (Szymoniak et al., 2021) to check our protocol. We made timed simulations using the current delay in the network value generated according to selected probability distributions. We observed time influence on our protocol. Badly selected time dependencies could be met even the current session time is greater than maximal session time. Also, we observed that any execution with the Intruder who impersonates an honest user was not possible. This suggests that our protocol is secure.

In further work, we will focus on developing a URL verification method. As part of this method, we plan to build a method, which will indicate whether a given URL is false or good. Distribution Center will use this method. Ultimately, we plan to prepare a tool that will use the protocol and related methods to protect users from making a false link.

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.

KEYWORDS: Security protocols, false URL detection, cybersecurity, neural network.

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.

Baccouche, A., Ahmed, S., Sierra-Sosa, D. & Elmaghraby, A. (2020). Malicious Text Identification: Deep Learning from Public Comments and Emails. Information, 11, 312.

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.

Čibej, U., Fürst, L. & Mihelič, J. (2019). A Symmetry-Breaking Node Equivalence for Pruning the Search Space in Backtracking Algorithms. Symmetry, 11, 1300.

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

Galinec, D., Steingartner, W. & Zebić, V. (2019). Cyber Rapid Response Team: An Option within Hybrid Threats," 2019 IEEE 15th International Scientific Conference on Informatics, Poprad, Slovakia.

Ispahany, J., & Islam, R. (2020). Detecting Malicious URLs of COVID-19 Pandemic using ML technologies.

Ko, Y., Kim, J., Duguma, D.G., Astillo, P.V., You, I. & Pau, G. (2021). Drone Secure Communication Protocol for Future Sensitive Applications in Military Zone. *Sensors*.

Kwon, D.K., Yu, S.J., Lee, J.Y., Son, S.H. & Park, Y.H. (2021). WSN-SLAP: Secure and Lightweight Mutual Authentication Protocol for Wireless Sensor Networks. *Sensors*.

Lai, C.-M., Shiu, H.-J. & Chapman, J. (2020) Quantifiable Interactivity of Malicious URLs and the Social Media Ecosystem. Electronics, 9.

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.

Maheswari, M., & Karthika, R.A. (2021). A Novel QoS Based Secure Unequal Clustering Protocol with Intrusion Detection System in Wireless Sensor Networks. *Wireless Pers Commun*.

Moreno-Cruz, F., Toral-López, V., Escobar-Molero, A., Ruíz, V.U., Rivadeneyra, A. & Morales, D.P. (2020). *treNch*: Ultra-Low Power Wireless Communication Protocol for IoT and Energy Harvesting. *Sensors*, *20*, 6156.

Patel, A. & Tailor, J. (2020). A malicious activity monitoring mechanism to detect and prevent ransomware, Computer Fraud & Security, Volume 2020, Issue 1, 2020, Pages 14-19.

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

Piątkowski, J. (2020). The Conditional Multiway Mapped Tree: Modeling and Analysis of Hierarchical Data Dependencies. IEEE Access 8: 74083-74092.

Radaković, D. & Herceg, D. (2018) Towards a completely extensible dynamic geometry software with metadata, Computer Languages, Systems & Structures, Volume 52, pp 1-20.

Siedlecka-Lamch, O. (2020). Probabilistic and timed analysis of security protocols, In proceeding of the 13th International Conference on Computational Intelligence in Security for Information Systems CISIS 2020, 16-18 September 2020, Burgos, Spain; paper 24.

Siedlecka-Lamch, O., Szymoniak, S. & Kurkowski, M. (2019) A fast method for security protocols verification, Computer Information Systems and Industrial Management, Springer.

Song, X., Chen, C., Cui, B. & Fu, J. (2020). Malicious JavaScript Detection Based on Bidirectional LSTM Model. Appl. Sci., 10, 3440.

Steingartner, W., Novitzka, V. & Schreiner, W. (2019). Coalgebraic operational semantics for an imperative language, Computing and Informatics, 38 (5), pp. 1181-1209.

Steingartner, W. & Galinec, D. & Kozina, A. (2021). Threat Defense: Cyber Deception Approach and Education for Resilience in Hybrid Threats Model, *Symmetry* 13, no. 4: 597.

Szymoniak, S. (2021). Security protocols analysis including various time parameters, Mathematical Biosciences and Engineering, 18(2): 1136-1153.

Szymoniak, S. (2020). How to be on time with security protocol?, Mario Arias Oliva, Jorge Pelegrín Borondo, Kiyoshi Murata, Ana María Lara Palma, Societal Challenges in the Smart Society ETHICOMP Book Series, Universidad de La Rioja, p. 225-237.

Szymoniak, S., Siedlecka-Lamch, O., Zbrzezny, A.M., Zbrzezny, A. & Kurkowski, M. (2021). SAT and SMT-Based Verification of Security Protocols Including Time Aspects. *Sensors*, *21*, 3055.

Xiao, D. & Jiang M. (2020). Malicious Mail Filtering and Tracing System Based on KNN and Improved LSTM Algorithm," 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Calgary, AB, Canada, 2020, pp. 222-229.

Yang, Z., Sun, Q., Zhang, Y. & Wang, W. (2020). Identification of Malicious Injection Attacks in Dense Rating and Co-Visitation Behaviors, in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 537-552, 2021, doi: 10.1109/TIFS.2020.3016827.