sensors

MDPI

Article

SAT and SMT-Based Verification of Security Protocols Including Time Aspects †

Sabina Szymoniak [1,*], Olga Siedlecka-Lamch [1], Agnieszka M. Zbrzezny [2] and Andrzej Zbrzezny [3] and Miroslaw Kurkowski [4]

[1] Department of Computer Science, Czestochowa University of Technology, Dabrowskiego 73, 42-200 Czestochowa, Poland; olga.siedlecka@icis.pcz.pl
[2] Faculty of Mathematics and Computer Science, University of Warmia and Mazury, Sloneczna 54, 10-710 Olsztyn, Poland; agnieszka.zbrzezny@matman.uwm.edu.pl
[3] Department of Mathematics and Computer Science, Jan Dlugosz University in Czestochowa, Armii Krajowej 13/15, 42-200 Czestochowa, Poland; a.zbrzezny@ujd.edu.pl
[4] Institute of Computer Science, Cardinal St. Wyszynski University, Woycickiego 1/3, 01-938 Warsaw, Poland; m.kurkowski@uksw.edu.pl
* Correspondence: sabina.szymoniak@icis.pcz.pl;
† This paper is an extended version of our paper published in "Network's Delays in Timed Analysis of Security Protocols". In Proceedings of the 39th International Conference on Information Systems Architecture and Technology—ISAT 2018, Nysa, Poland, 1–18 September 2018.

Abstract: For many years various types of devices equipped with sensors have guaranteed proper work in a huge amount of machines and systems. For the proper operation of sensors, devices, and complex systems, we need secure communication. Security protocols (SP) in this case, guarantee the achievement of security goals. However, the design of SP is not an easy process. Sometimes SP cannot realise their security goals because of errors in their constructions and need to be investigated and verified in the case of their correctness. Now SP uses often time primitives due to the necessity of security dependence on the passing of time. In this work, we propose and investigate the SAT-and SMT-based formal verification methods of SP used in communication between devices equipped with sensors. For this, we use a formal model based on networks of communicating timed automata. Using this, we show how the security property of SP dedicated to the sensors world can be verified. In our work, we investigate such timed properties as delays in the network and lifetimes. The delay in the network is the lower time constraint related to sending the message. Lifetime is an upper constraint related to the validity of the timestamps generated for the transmitted messages.

Keywords: security protocols; modelling; verification; sensor network protocols; time analysis

Citation: Szymoniak, S.; Siedlecka-Lamch, O.; Zbrzezny, A.M.; Zbrzezny, A.; Kurkowski, M. SAT and SMT-Based Verification of Security Protocols Including Time Aspects. *Sensors* **2021**, *21*, 3055. https://doi.org/10.3390/s21093055

Academic Editor: Fatos Xhafa

Received: 10 April 2021
Accepted: 23 April 2021
Published: 27 April 2021

1. Introduction

Contemporary fast life requires constant communication and information exchange. Modern communication takes place between people or institutions and increasingly between various types of devices equipped with sensors. Often, such devices are autonomous and equipped with software that uses artificial intelligence algorithms. There are many examples of such devices around us, ranging from cleaning robots to autonomous cars or even aircraft. Regardless of the type of participants, communication must remain secure, and the protocols used for this purpose should be verified and, if it is necessary, improved.

Security protocols (SP) are algorithms that ensure crucial security objectives in the communication process. Often for this purpose, cryptographic algorithms and techniques are applied. During communication, the most important data security goals are authorisation, information security and integrity, and cryptographic key distribution. For many years, many such protocols used timestamps. In addition, security protocols can be used in IoT systems [1–5]. It is necessary for the appropriate time-dependent management of

cryptographic primitives (keys, passwords), especially to preserve and monitor the validity (lifetime) of the primitives.

Such a time-dependent management of primitives is sometimes crucial due to the increasing number of elements guaranteeing data confidentiality, such as encryption keys, timestamps, and number of protocol steps. Having precisely defined lifetimes of primitives, administrators have full control over the data necessary to maintain secure communication and can choose parameters for it. In the case of protocols used for devices equipped with sensors, such tasks are even more complicated and the time aspect even more important. These devices usually have limited resources, relatively small memory, limited battery life, hence the limited energy, and do not have the computing power to generate complex ciphertexts. Therefore, it is essential to be able to match the protocol and its parameters to such limitations.

Since the first projects and implementations, many SPs have been developed and successfully used. Unfortunately, many examples have shown that protocols sometimes can be cheated by malicious intruders [6,7]. That is the reason for investigating SP properties, especially widely understood correctness. From the 1990s, many well-grounded methodologies and practically used verification tools helped in verifying and designing SP. The first attempts of checking SP properties were by testing real or virtual environments. Nevertheless, in this case, we must say that even many testing days, months, or years cannot answer the question of whether the investigated SP achieves its goal or not. After a consierable period of testing, we can only say that the system works correctly so far.

Another way of SP property verification is by using the formal deduction methods. Logics created for reasoning about SP properties were widely known as authentication logics. These logics have been successfully applied to investigating SP correctness [8]. Formal reasoning allowed to discover errors in several SP schemes by proving the bugs or weaknesses in their constructions. However, such methods had serious problems. Imperfect deductive systems or the huge size of proof trees caused the verification process to be ineffective. In cases of logics with well-grounded (adequate or complete) semantics, it could be seen that inference can be made on the semantic side by examining the computational model on which the semantics of logic was based. This was one of the reasons why authentication logics gradually ceased to be popular. In fact, it can even be said that in this field, authentication logics have been successfully supplanted by model verification methods that examine computational structures representing the protocol's executions.

Taking the above into account, we should state that the most important formal ways for the SP properties' verification are, from many years, model checking techniques. In such approaches, an appropriate and adequate formal model of the protocol executions is created and investigated. There were many concepts of SPs' modelling and many algorithms of how created formal models can be searched. In the case of SP verification, we have to notice a few beneficial and very well-grounded approaches. The first one is the well-known AVISPA tool [9], which uses HLPSL, a specially designed language for SP specification. Others are Scyther proposed by Cremers and Mauw [10], and ProVerif by Blanchet [11]. These tools allow for automatic investigations of SP properties, especially their correctness. We can also annotate Kurkowski's solutions in the VerICS tool, where protocols' executions were modelled as work of networks of synchronised automata [12]. Such methods and tools are still developed and practically used to verify many protocols. See for example [13–18].

Here, we have to annotate that only several solutions that can express or investigate SP time properties have been introduced so far. It is a very important problem because for several years developers added into protocols schemes time tickets and their lifetimes' values. It should be noted that time modelling is not an easy problem at all. In this case several ways of expressing time and time properties in formal computer systems models were introduced [14,16,19–21].

Our work follows for works of Jakubowska and Penczek [22], and Kurkowski and Penczek [12]. In the first approach, the authors did not go beyond investigating security

properties connected with only one SP's session. In [23] new formal, discrete, mathematical methods for protocol verification have been introduced. With the presented methods, it is possible to prove the correct operation of the time-dependent security protocol. This model was used to study authentication processes. In further works, one can find the next considerations regarding network delays and calculations regarding the duration of the communication session. The analysed time constraints revealed the influence of time on protocol security.

Tne above-mentioned allow further successful investigations that were published in [24–27]. In the research conducted so far, we have used synchronised automata networks, modelling by chains of states, SAT, and SMT techniques. The latest research concerns the temporal aspects. In this paper, we present developed methods and the newest improvements and the extension to further examples of protocols. The developed model showed the strengths and weaknesses of the tested protocols and the fact that one can use even potentially weak protocols with appropriate time constraints, which can be particularly useful in the case of sensor systems. It can also be noticed that there is a way to increase the protocol's security by strengthening key points. We used the tools we implemented as well as SAT and SMT solvers for the experimental research.

In our work we distinguish two types of intruder behaviour. The first one is the actual attack where the intruder can get confidential data or cheat someone their identity. We can precisely say that such behaviours are attacks. The second one is some case of Man in the Middle situation, where the intruder, staying between two communicating sides, can only receive and resend the whole message during protocols execution without breaking ciphertexts, possessing secret data, or cheat someone their identity. The second type of behaviour we call Man in the Middle Intruder's behaviour (MiTM behaviour). Such a behaviour is a simple passive resending of messages. In some papers, such behaviours are called attacks, which may be confusing in understanding the problem.

The main contributions of the paper are as follows:

- Formal modelling of timed security protocols dedicated to sensor devices on the example of the SNEP protocol,
- Timed analyses of the SNEP protocol,
- Applying SAT- and SMT-based to reachability testing,
- And analyses of dependence of lifetime on delays.

Related Work

In the last several years only a few research groups have investigated problems connected to modern timed solutions in SP schemes. In works [28] the authors developed and considered THLPSL language (Timed High-Level Protocols Specification Language). However, only the simplest timed properties were investigated. These works did not contain considerations about networks' delays, times of messages compositions, and ciphering/deciphering. Szymoniak et al. [25,29–31] and Li et al. [32] took into account networks delays. It allowed consideration about time dependencies between the possibility of protocol executing and lifetimes of time tickets values. Li et al. created an automatic system for computing proper time dependencies between network delays and lifetimes. However, they only considered one value each for lifetime and delay. In the case of sensors devices devoted protocols, only a few papers were dedicated to verifying their properties. Here the AVISPA tool was used. However, only untimed versions of such protocols were taken into account [33].

The rest of the paper is organised as follows. In the next chapter, we describe example protocols that will help us explain how to model and verify. Next, we present the necessary definitions for timed automata and the network of timed automata. In a further chapter, we present the formal language and computational mathematical structure. In the next section, we will describe how to apply SAT and SMT techniques. Then we present the research assumptions and the results of the experiments. We will present the conclusions in the last section.

Table 1 presents the notations used in the article together with explanations:

Table 1. Basic notations.

Notations	Explanations
α_i^j	the i-th step in the j-th execution of the protocol
$X \cdot Y$	a concatenation of messages X and Y
A, B, S, I	participants of communication
I_U	user ID
T_U	the timestamp created by user U
N_U	pseudorandom number (nonce) generated by user U
K_{UX}	the encryption symmetric key shared between users U and X
$\langle D \rangle_K$	the message D encrypted with key K
$\langle D \rangle_{M(K)}$	the computation of the message authentication code (MAC) of message D, with MAC key K
$\langle D \rangle_{K,C}$	the message D encrypted with key K and the counter C
$\langle D \rangle_{M(K)}$	the computation of the message authentication code (MAC) of message D, with MAC key K and the counter C

2. Sample Protocols

In this section, we will present two examples of protocols. The first one is the Woo Lam Pi standard security protocol used in more complex communication protocols. The second one—the SNEP protocol is part of sensor communication systems. Both will be used later in work to explain formal modelling and present the experimental results.

2.1. WooLamPi Protocol

We have chosen the WooLamPi (WLP) protocol for our presentation because, like SNEP, it uses symmetric cryptography and has similar construction of some transmitted data (with double encryption). The WLP protocol was designed by Thomas Y. C. Woo and Simon S. Lam's and described in work [34]. This protocol uses symmetric cryptography, which is a one-way authenticator using a trusted server. WLP is the basis for the next version of this protocol (WooLamPi1, WooLamPi2, and WooLamPi3). In the original, the WooLamPi protocol are used nonces (pseudorandom numbers).

To perform research with time parameters, we, in a usual way, change nonce values into timestamps, thus gaining the opportunity to take into account time and be able to consider and verify time properties. It is important to note that changing nonces to timestamps is a practice confirmed by suitable security standards and widely used in real solutions in this area. Nonces and timestamps as time-variant parameters are officially considered to be equivalent from a time-dependent properties point of view. To increase the security level of communication, we can add pseudorandom values (nonces) in two ways: Firstly as part of the timestamp or second as an additional part of the whole message. The details of such a mechanism can be found in the ISO/IEC 9798 norm.

The timed version of this protocol in the so-called 'Alice and Bob' notation is as follows:

$$\begin{array}{llll} \alpha_1 & A \to B & : & I_A \\ \alpha_2 & B \to A & : & T_B \\ \alpha_3 & A \to B & : & \langle T_B \rangle_{K_{AS}} \\ \alpha_4 & B \to S & : & \langle I_A \cdot \langle T_B \rangle_{K_{AS}} \rangle_{K_{BS}} \\ \alpha_5 & S \to B & : & \langle T_B \rangle_{K_{BS}}. \end{array}$$

The WLP protocol consists of five steps. In the first step, Alice (user A) sends its identifier to Bob (user B), informing him of her willingness to initiate a new session. In response, Bob generates a timestamp T_B and sends it to Alice. Then Alice creates a message in which the timestamp T_B is received. Bob forwards this message to the server (marked with the letter S) by adding the I_A identifier to it. The entire message is encrypted with a symmetric key K_{BS} shared between B and the server S. In the last step of this protocol, the server sends its timestamp to Bob in the message encrypted with the K_{BS} symmetric key.

The WLP protocol has many versions; their descriptions can be found in [34]. Each version of the WLP has the same number of steps, and the same operations are performed in the first and second steps. Let us show the next version from the third step, where some modifications were made to ensure the security of the protocol:

$$\begin{array}{lllll} \alpha_3 & A \to B & : & \langle I_A \cdot I_B \cdot T_B \rangle_{K_{AS}} \\ \alpha_4 & B \to S & : & \langle I_A \cdot I_B \cdot \langle I_A \cdot I_B \cdot T_B \rangle_{K_{AS}} \rangle_{K_{BS}} \\ \alpha_5 & S \to B & : & \langle I_A \cdot I_B \cdot T_B \rangle_{K_{BS}}. \end{array}$$

The message from the third step includes the generated timestamp T_B and the identifiers of both users. The following two messages also contain the identifiers of both users.

For the WooLamPi2 protocol, the steps are as follows:

$$\begin{array}{lllll} \alpha_3 & A \to B & : & \langle I_A \cdot T_B \rangle_{K_{AS}} \\ \alpha_4 & B \to S & : & \langle I_A \cdot \langle I_A \cdot T_B \rangle_{K_{AS}} \rangle_{K_{BS}} \\ \alpha_5 & S \to B & : & \langle I_A \cdot T_B \rangle_{K_{BS}}. \end{array}$$

From step 3, the initiator ID (user A) has been added for all messages.

For the WooLamPi3 protocol, the base protocol is as follows:

$$\begin{array}{lllll} \alpha_4 & B \to S & : & \langle I_A \cdot \langle T_B \rangle_{K_{AS}} \rangle_{K_{BS}} \\ \alpha_5 & S \to B & : & \langle I_A \cdot T_B \rangle_{K_{BS}}. \end{array}$$

There are no severe changes in the last steps. In the fourth and fifth steps, the initiator's ID appears. No attack on secrecy or authentication was detected on any of the WooLamPi protocols. Only Man in the Middle behaviour is possible (without taking over confidential information).

2.2. SNEP Protocol

The Secure Network Encryption Protocol (SNEP) is a security protocol designed for sensor networks. It was introduced at [35] by Perrig et al. This protocol ensures data confidentiality, two-way data authentication, and its validity. SNEP provides semantic security in the form of a strong security feature. Thanks to this, the intruder cannot infer the content of the plaintext from the multiple-encrypted message.

In the SNEP protocol, authors have introduced an additional cryptographic mechanism to reduce the energy required to transmit random data over the wireless channel. This mechanism uses two counters (one for each communication direction). The communicating parties share these counters. However, they are not sent and incremented with each message as in the traditional approach. The exchange of counters takes place using an additional protocol (counter exchange protocol) that synchronises the counters between the communicating parties. In turn, the Message Authentication Code (MAC) achieves two-way authentication and data integrity.

SNEP uses independent symmetric encryption keys and MAC codes for all its operations. Here symmetric encryption works in a counter mode of operation, and MAC function over data is a function that computes a MAC code of data with additional parameters K-key and C-counter. Firstly we assume that the two communicating parties A and B share an $\mathcal{X}_{AB}$ master secret key and obtain from it independent encryption keys for each communication direction and MAC keys using some fixed pseudorandom function F over $\mathcal{X}_{AB}$. In the further part of the analysis, we will use the keys derived from the key $\mathcal{X}_{AB}$, namely the keys K_{AB} and K_{BA}. They are symmetric keys, known by both users A and B, for communication between them. It should be emphasised that they are not mathematically the same keys, although they perform the same data encryption role. This is essential information because of modelling details in the following sections of the paper.

Encrypted data in the SNEP protocol takes the following format:

$$E = \langle D \rangle_{K,C} \tag{1}$$

where $\langle D \rangle_{K,C}$ denotes a ciphertext that contains plaintext D encrypted in a counter mode of operation with the symmetric encryption key and the counter C.

Observe that ciphertexts encrypted with keys K_{AB}, K_{BA}, and the counter C: $\langle D \rangle_{K_{AB},C}$, and $\langle D \rangle_{K_{BA},C}$ are not identical.

We will describe the MAC message authentication code of the data E computed with the parameters K (the key) and C (the counter) as $\langle E \rangle_{M(K,C)}$.

The SNEP protocol original (untimed) version consists of two parts. The goal of the first one is setting the user's counters, and the second is the part responsible for authentication.

Let us describe the first part as follows:

$$\begin{array}{llll} \alpha_1 & A \to B & : & C_A \\ \alpha_2 & B \to A & : & C_B \cdot \langle C_B \rangle_{M(K'_{BA},C_A)} \\ \alpha_3 & A \to B & : & \langle C_B \rangle_{M(K'_{AB},C_A)}. \end{array}$$

In the first step the user A send to the user B the counter value C_A. In the second step, the user B sends to A its counter value C_B with this value's MAC code computed using the key K'_{BA} and obtained C_A value. Observe that both users A and B know $\mathcal{X}_{AB}$ and can compute from keys K'_{BA} and K'_{AB}. In the third step A send to B the MAC code of the value C_B computed using the key K'_{AB} and value of its counter C_A. As we mentioned before, in this part of SNEP, A and B exchange their counter values. This part can be done rarely in the usual communication process, for example, once a day.

The second part of SNEP is devoted to one side authentication of the user B to the user A. This part should be done as a start in every communication session. This part consists of two steps:

$$\begin{array}{llll} \alpha_4 & A \to B & : & N_A \cdot R_A \\ \alpha_5 & B \to A & : & \langle R_B \rangle_{K_{BA},C_B} \cdot \langle \langle R_B \rangle_{K_{BA},C_B} \rangle_{M(K'_{BA},N_A \cdot C_B)}. \end{array}$$

In the α_4 step, the user A generates its own nonce N_A and sends it to Bob with a special code or text R_A that plays the role of the request for the answer. Observe that N_A is generated strictly at the moment of starting this part of SNEP. Before this time, the number N_A did not exist. In the α_5 step, B answers to A through a message containing the ciphertext $\langle R_B \rangle_{K_{BA},C_B}$ (remember that only A and B know the key). This message also contains the MAC computation value of $\langle R_B \rangle_{K_{BA},C_B}$ computed with using the N_A value, freshly generated and obtained by B in α_4 step (R_B, similarly R_A is a text or a code). After the MAC code has been appropriately verified, A will know that B generated a response to the message request. The usage of N_A ensures the high freshness of the response. As we mentioned before, this part of SNEP guarantees a one-side authentication of the user B to the user A. Remember additionally that only A and B knows $\mathcal{X}_{AB}$ and computed from this keys K_{AB}, K_{BA}, K'_{AB}, and K'_{BA}. This SNEP's part can be executed in the reverse direction and confirm the authentication of A to B.

For the next consideration point of view, we should note that the authors of the SNEP protocol use the MAC function because of the small computational and memory power of sensors devices. Today on the cryptographic market, there are many examples of light cyphers, for example, Cha-Cha or Salsa, that can be successfully used in investigated devices (see [36–38]). Thus in the following considerations, we use a slightly modified version of SNEP, with symmetric light encryption instead of the MAC function. From the cryptographic point of view, there is no difference here. Additionally, we add for the SNEP scheme timestamps that allow investigation about SNEP time properties.

In our research, we model and check the security time properties of two versions of SNEP. In the first one, both parts of the protocol (reconciliation of meters and proper authorisation) follow each other. In this case, the protocol scheme is as follows:

$$\begin{array}{lllll}
\alpha_1 & A \to B & : & T_A \cdot C_A \\
\alpha_2 & B \to A & : & T_B \cdot C_B \cdot \langle T_A \cdot C_A \cdot T_B \cdot R_B \rangle_{K'_{BA}} \\
\alpha_3 & A \to B & : & N_A \cdot R_A \cdot \langle T_A \cdot C_A \cdot T_B \cdot R_B \rangle_{K'_{AB}} \\
\alpha_4 & B \to A & : & \langle T_B \cdot R_B \cdot C_B \rangle_{K_{BA}} \cdot \langle N_A \cdot T_B \cdot R_B \cdot \langle C_B \cdot T_B \cdot R_B \rangle_{K_{BA}} \rangle_{K'_{BA}}.
\end{array}$$

If we allow a certain distance in time and re-establish communication between users, we can write SNEP in this way:

$$\begin{array}{lllll}
\alpha_1 & A \to B & : & T_A \cdot C_A \\
\alpha_2 & B \to A & : & T_B \cdot C_B \cdot \langle T_A \cdot C_A \cdot T_B \cdot R_B \rangle_{K'_{BA}} \\
\alpha_3 & A \to B & : & \langle T_A \cdot C_A \cdot T_B \cdot R_B \rangle_{K'_{AB}} \\
\alpha_4 & B \to A & : & I_B \cdot I_A \\
\alpha_5 & A \to B & : & N_A \cdot R_A \\
\alpha_6 & B \to A & : & \langle T_B \cdot R_B \cdot C_B \rangle_{K_{BA}} \cdot \langle N_A \cdot T_B \cdot R_B \cdot \langle C_B \cdot T_B \cdot R_B \rangle_{K_{BA}} \rangle_{K'_{BA}}.
\end{array}$$

Both proposed versions meet all assumptions of the original version of SNEP.

For clarity of further analysis, let us also show the MiTM behaviour for one SNEP version. Let us present it on a shorter version of the protocol. For such a situation to occur, we need to interlace the two executions. In the first execution (α_i^1), the intruder impersonates user B and communicates with A. In the second execution (α_i^2), they impersonate A and communicates with B.

$$\begin{array}{lllll}
\alpha_1^1 & A \to I(B) & : & T_A \cdot C_A \\
\alpha_1^2 & I(A) \to B & : & T_A \cdot C_A \\
\alpha_2^2 & B \to I(A) & : & T_B \cdot C_B \cdot \langle T_A \cdot C_A \cdot T_B \cdot R_B \rangle_{K'_{BA}} \\
\alpha_2^1 & I(B) \to A & : & T_B \cdot C_B \cdot \langle T_A \cdot C_A \cdot T_B \cdot R_B \rangle_{K'_{BA}} \\
\alpha_3^1 & A \to I(B) & : & N_A \cdot R_A \cdot \langle T_A \cdot C_A \cdot T_B \cdot R_B \rangle_{K'_{AB}} \\
\alpha_3^2 & I(A) \to B & : & N_A \cdot R_A \cdot \langle T_A \cdot C_A \cdot T_B \cdot R_B \rangle_{K'_{AB}} \\
\alpha_4^2 & B \to I(A) & : & \langle T_B \cdot R_B \cdot C_B \rangle_{K_{BA}} \cdot \langle N_A \cdot T_B \cdot R_B \cdot \langle C_B \cdot T_B \cdot R_B \rangle_{K_{BA}} \rangle_{K'_{BA}} \\
\alpha_4^1 & I(B) \to A & : & \langle T_B \cdot R_B \cdot C_B \rangle_{K_{BA}} \cdot \langle N_A \cdot T_B \cdot R_B \cdot \langle C_B \cdot T_B \cdot R_B \rangle_{K_{BA}} \rangle_{K'_{BA}}.
\end{array}$$

For both versions of the SNEP protocol (as presented earlier WooLamPi), the intruder's presence inside the communication is not a direct attack. The intruder does not gain confidential information and can only store entire ciphertexts. It should be emphasised that this is also an undesirable situation that we wish to avoid.

3. Networks of Synchronised Timed Automata

Now we start introducing formal structures used for protocols' modelling and verification. As mentioned before, we model executions of timed security protocols as a network of synchronised timed automata. Our work bases on methodology introduced precisely in [12]. Below we show needed formal definitions due to such networks required for the next considerations conducted in the following sections. As an example, we will show how to model the executions of a protocol by the runs of a network of synchronised timed automata, where each timed automaton represents one component of the protocol. We will also introduce automata representing messages transmitted during a protocol execution and automata representing users' knowledge. We need to model the executions of the protocol according to the knowledge acquired by users. We have introduced an appropriate synchronisation between the automata of the network.

Let $\mathcal{X} = \{x_0, \ldots, x_{n-1}\}$ be a finite set of variables, called clocks. By clock valuation we mean a total function $v : \mathcal{X} \mapsto \mathbb{R}$ that assigns to each clock x a non-negative real number $v(x)$. We also denote by $\mathbb{R}^{|\mathcal{X}|}$ the set of all the clock valuations.

Let $x \in \mathcal{X}$, $c \in \mathbb{N}$, and $\sim \in \{<, \leqslant, =, \geqslant, >\}$. Clock constraints over $\mathcal{X}$ are conjunctions with information about comparisons of a clock with a time constant from the set of non-negative natural numbers $\mathbb{N}$. The set $\mathcal{C}$ of clock constraints over the set of clocks $\mathcal{X}$ is defined by the following grammar:

$$\mathfrak{cc} := x \sim c \mid \mathfrak{cc} \wedge \mathfrak{cc}.$$

Additionally let v be a clock valuation, and $\mathfrak{cc} \in \mathcal{C}$. We say that a clock valuation v satisfies a clock constraint $\mathfrak{cc}$, iff $\mathfrak{cc}$ evaluates to true using the clock.

Definition 1. *A timed automaton [19,39] (TA, for short) is a seven-tuple* $\mathcal{A} = (Act, L, l^0, \mathcal{X}, AP, E, V)$*, where:*

- Act *is a finite set of* actions,
- L *is a finite set of* locations,
- $l^0 \in L$ *is the* initial location,
- $\mathcal{X}$ *is a finite set of clocks,*
- AP *is a set of atomic propositions,*
- $E \subseteq L \times Act \times \mathcal{C} \times 2^{\mathcal{X}} \times L$ *is a* transition relation,
- $V : L \mapsto 2^{AP}$ *is a valuation function assigning to each location a set of atomic propositions.*

Every element e from the set E is denoted by $l \xrightarrow{a,\mathfrak{cc},X} l'$*. It represents a transition from the location l to the location* l'*, executing the action a, with the set* $X \subseteq \mathcal{X}$ *of clocks to be reset, and with the clock condition* $\mathfrak{cc} \in \mathcal{C}$ *defining the enabling condition (guard) for e.*

The clocks of a timed automaton allow for expressing the timing properties. A guard restricts the execution of the transition and at the same time does not force the execution of this transition.

We give at the next sections several examples of timed automata used for modelling of protocols executions.

3.1. Semantics of Timed Automata

Let $\mathcal{A} = (Act, L, l^0, E, \mathcal{X})$ be a timed automaton. A concrete state of $\mathcal{A}$ is defined as an ordered pair (l, v), where $l \in L$ and $v \in \mathbb{R}_+^{|\mathcal{X}|}$ is a clock valuation ($\mathbb{R}_+$ denotes the set of positive real numbers). In what follows, by $[\![\mathfrak{cc}]\!]$ we mean a set of all the valuations v that satisfy a time condition $\mathfrak{cc}$. By $v[X := 0]$ we mean the valuation v' such that it assigns 0 to all clocks from the set X and it agrees with v on all the remaining clocks, i.e., $v'(X) = \{0\}$ and $v'(x) = v(x)$ for each $x \in \mathcal{X} \setminus X$. The formal definitions of these notations can be found in [12].

The *state space* of $\mathcal{A}$ is a transition system $C(\mathcal{A}) = (Q, \mathfrak{s}^0, \rightarrow, \mathcal{V})$ [39], where:

- $Act \cup \mathbb{R}_+$ is the set of labels,
- $Q = L \times \mathbb{R}_+^{|\mathcal{X}|}$ is the set of all the *concrete states*,
- $\mathfrak{s}^0 = (l^0, v^0)$ with $v^0(x) = 0$ for all $x \in \mathcal{X}$ is the *initial state*,
- $\rightarrow \subseteq Q \times (Act \cup \mathbb{R}_+) \times Q$ is the *transition relation*, defined by action- and time successors as follows:
  - For any $\delta \in \mathbb{R}_+$, $(l, v) \xrightarrow{\delta} (l, v + \delta)$ (*time successor*),
  - For $a \in Act$, $(l, v) \xrightarrow{a} (l', v')$ iff $(\exists \mathfrak{cc} \in \mathcal{C})(\exists X \subseteq \mathcal{X})$ such that $l \xrightarrow{a,\mathfrak{cc},X} l' \in E$, $v \in [\![\mathfrak{cc}]\!]$ and $v' = v[X := 0]$ (*action successor*).
- A valuation function $\mathcal{V} : Q \mapsto 2^{AP}$ is defined such that $\mathcal{V}((l, v)) = V(l)$ for all $(l, v) \in Q$.

Intuitively, a time successor does not change the location l of a concrete state, but it increases the clocks. An action successor corresponding to an action a is executed when the guard $\mathfrak{cc}$ holds for v and the valuation v' obtained by resetting the clocks in X.

We shall say that a location l is reachable in a given transition system for a timed automaton $\mathcal{A}$ if for some clock valuation v the state (l, v) is reachable in this transition system.

For $(l, v) \in Q$ and $\delta \in \mathbb{R}_+$, let $(l, v) + \delta$ denote $(l, v + \delta)$. A $\mathfrak{s}_0$-*run* ρ of $\mathcal{A}$ is a maximal sequence $\rho = \mathfrak{s}_0 \xrightarrow{\delta_0} \mathfrak{s}_0 + \delta_0 \xrightarrow{a_0} \mathfrak{s}_1 \xrightarrow{\delta_1} \mathfrak{s}_1 + \delta_1 \xrightarrow{a_1} \mathfrak{s}_2 \xrightarrow{\delta_2} \ldots$, where $a_i \in Act$ and $\delta_i \in \mathbb{R}_+$, for each $i \in \mathbb{N}$.

Now, we are going to use networks of timed automata (NTA) for modelling executions of the protocol and the knowledge of the participants.

3.2. Product of a Network of Timed Automata

Now we consider a product **A** of $i-$th timed automata constructed before. In a standard way, **A** is a timed automaton too, so it consists of seven components. A state of **A** is a $i-$tuple of states of constituents automata. In **A**, a transition labelled by α-label can be executed iff in all constituents automata that possess α, at least one transition labelled by α can be executed too. All other conditions that determine other components of **A** are intuitively similar (see [12] for detail).

4. Formal Language and Computational Structure

This section focuses on the essential elements of the formal language necessary to build a computational structure. The entire formal model and the computational structure was shown in [31]. Initially, a protocol step definition is required.

In the following considerations, among these introduced before, we use the additional notations mentioned in Table 2.

Table 2. Additional notations.

Notations	Explanations
τ	the time of sending the message
τ_U	the timestamp created by the user U
L_{τ_U}	the lifetime of the given timestamp
D_i	the delay for the given step
$\mathcal{A}_M^U$	the knowledge automaton that represent knowledge of the user U about the element M
x_0	the global clock
x_{τ_U}	the clock for timestamp created by the user U

Formally a timed protocol step (including delay in the network) is defined by two tuples, where:

$$\begin{array}{rcl} \alpha^1 & = & (S_{\rightarrow}, R_{\leftarrow}, L) \\ \alpha^2 & = & (\tau, D, X, G, tc). \end{array}$$

In this notation, L is the message sent in the step, $S_{\rightarrow}$ is the sender, and $R_{\leftarrow}$ is a receiver. From studying the properties of temporal security protocols point of view, the second tuple is very important. Here, τ is the time of sending the message, D is a delay in the network, X is the set of letters necessary to compose the message L, G is the set of letters that the sender must generate to compose the message L, and tc is the set of timed conditions that should be met in order to enable the protocol execution.

As an example, we presented the formal definition of the WLP Protocol:

$$\begin{array}{c} \alpha_1 = (\alpha_1^1, \alpha_1^2) \quad \alpha_1^1 = (A, B, I_A) \\ \alpha_1^2 = (\tau_1, D_1, \{I_A\}, \{\emptyset\}, true) \end{array}$$

$$\begin{array}{c} \alpha_2 = (\alpha_2^1, \alpha_2^2) \quad \alpha_2^1 = (B, A, \tau_B) \\ \alpha_2^2 = (\tau_2, D_2, \{\tau_B\}, \{\tau_B\}, true) \end{array}$$

$$\begin{array}{c} \alpha_3 = (\alpha_3^1, \alpha_3^2) \quad \alpha_3^1 = (A, B, \langle \tau_B \rangle_{K_{AS}}) \\ \alpha_3^2 = (\tau_3, D_3, \{\tau_B, K_{AS}\}, \{\emptyset\}, \tau_3 + D_3 - \tau_B \leq L_{\tau_B}) \end{array}$$

$$\begin{array}{c} \alpha_4 = (\alpha_4^1, \alpha_4^2), \quad \alpha_4^1 = (B, S, \langle I_A, \langle \tau_B \rangle_{K_{AS}} \rangle_{K_{BS}}), \\ \alpha_4^2 = (\tau_4, D_4, \{I_A, \langle \tau_B \rangle_{K_{AS}} \rangle_{K_{BS}}\}, \{\emptyset\}, \tau_4 + D_4 - \tau_B \leq L_{\tau_B}). \end{array}$$

$$\begin{array}{c} \alpha_5 = (\alpha_5^1, \alpha_5^2), \quad \alpha_5^1 = (S, B, \langle \tau_B \rangle_{K_{AS}}), \\ \alpha_5^2 = (\tau_3, D_5, \{\tau_B, K_{BS}\}, \{\emptyset\}, \tau_5 + D_5 - \tau_B \leq L_{\tau_B}). \end{array}$$

According to the WLP protocol structure, we will discuss the third step in detail. At this point, user A sent to user B a message. To compose this message, user A needs the following set of objects: $\{\tau_B, K_{AS}\}$. The set of generated objects is empty. It means that user A does not need to generate any objects in this step. The sending time of this message was extended by delay in the network and shortened by the timestamp value. This time must be less than or equal to the assumed lifetime value. The previous and next steps of the WLP protocol should be considered in the same way.

We can model executions of security protocols as specially designed discrete, mathematical transition structures. One of these structures is a network of synchronised timed automata. Our network works according to the formal definition of a network of synchronised timed automata presented in [12,21]. In this network, the global state is the tuple that consists of precisely one state from each automaton. The initial state of the network is a tuple that consists of all the initial states of all automata. A given action α can be executed in the network if and only if α is enabled in all the automata in which this action appears.

We consider two types of timed automata in our network: Knowledge and execution automata. Execution automata model executions protocol steps together with time conditions. The second type of automata models the process of gaining knowledge by users. These automata are synchronised by labels that allow modelling the need to acquire specific knowledge by users to execute the next protocol step.

Figure 1 shows a part of the network of synchronised timed automata. The network models an execution of the WooLamPi protocol (including delays in the network). In this picture, we marked the initial global state of our network, which is a tuple of the local initial states (denoted by the black dot). Automaton $\mathcal{A}$ models the execution of all protocol steps and time conditions. Each of $\mathcal{A}$'s transitions are connected with the proper protocol's step. The automata in Figure 1 model the changes in users' knowledge during the execution of the protocol. For example, the first automaton $\mathcal{A}^A_{\tau_B}$ models gaining knowledge about the timestamp τ_B by the user A. The first, initial state of $\mathcal{A}^A_{\tau_B}$ models a state where the user A does not possess knowledge about τ_B. The first transition in $\mathcal{A}^A_{\tau_B}$ labelled by α_2 is synchronised with the second transition of execution automaton $\mathcal{A}$. Such synchronisation guarantees possessing knowledge about the timestamp τ_B by the user A. The second local state of $\mathcal{A}^A_{\tau_B}$ describes a situation when the user A knows the timestamp τ_B. A loop labelled with the label α_3 is connected with the situation when the knowledge about τ_B is necessary to execute another step connected with transitions in execution automaton $\mathcal{A}$, when the user A needs the knowledge about the timestamp τ_B. Such ideas and constructions were proposed in [12]. For a network of synchronised timed automata, we use the global clock x_0 and clocks x_{τ_U} for all timestamps created by the users. For the presented part of the network, we use the clock x_{τ_B} that measures the time from the point of creating the timestamp τ_B. Please note that after each transition we reset a global clock x_0 and compare its value with the appropriated value of D, according to protocol structure. In addition, in the case of knowledge automata, we compare the value of clock x_{τ_B} with L_1 value, according to the protocol structure.

Note that each transition compares the global clock x_0 with the appropriated delay D, and then resets x_0. In the case of knowledge automata, the value of clock x_{τ_B} is compared with the lifetime L_{τ_B}, according to the protocol structure.

The initial global state changes to another by executing the action α_1—the only one enabled in the initial state. The result of executing the action α_1 is shown in Figure 2. Observe that according to protocol execution, the users' knowledge is not changed in the first step.

After that, the second step of the protocol can be executed (Figure 3). The second transition in the automaton A is synchronised with the first transitions in automata $\mathcal{A}^A_{\tau_B}$ and $\mathcal{A}^B_{\tau_B}$. For the WLP protocol, the second step's execution requires changing user B's knowledge about its timestamp because user B generates timestamp τ_B. In addition, user A's knowledge about this ticket is changing. User A possess this timestamp. Clock x_{τ_B} must be reset.

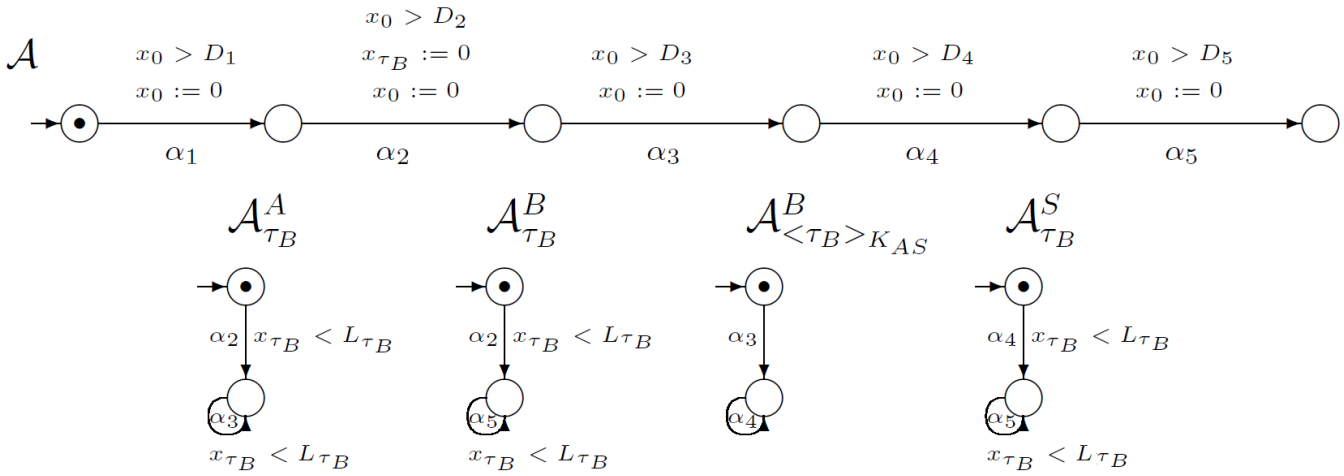

Figure 1. Network of synchronised timed automata for the WooLamPi protocol.

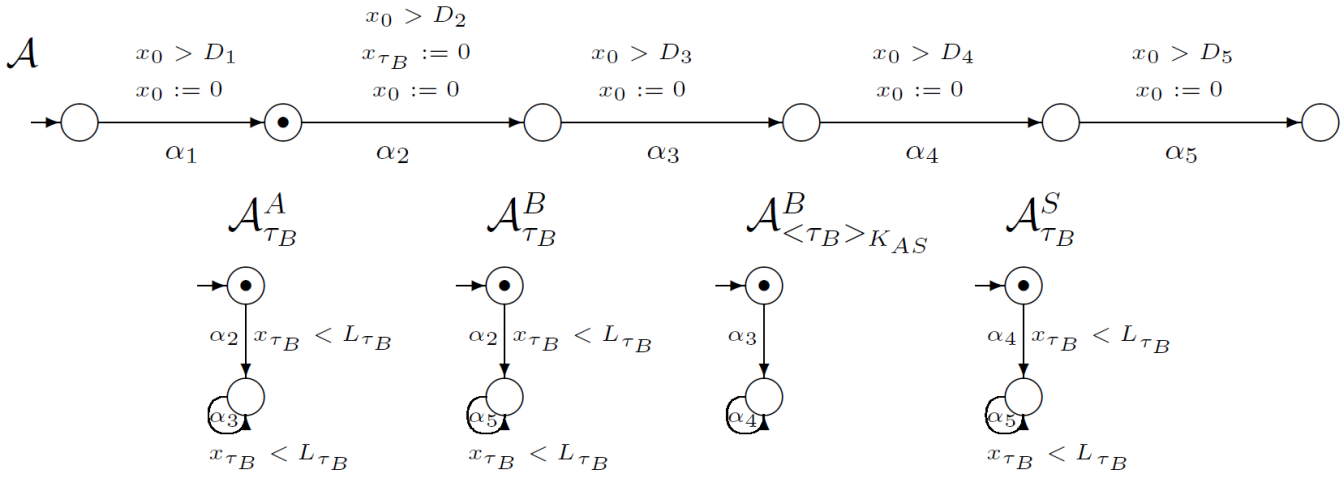

Figure 2. The first step of the WooLamPi protocol.

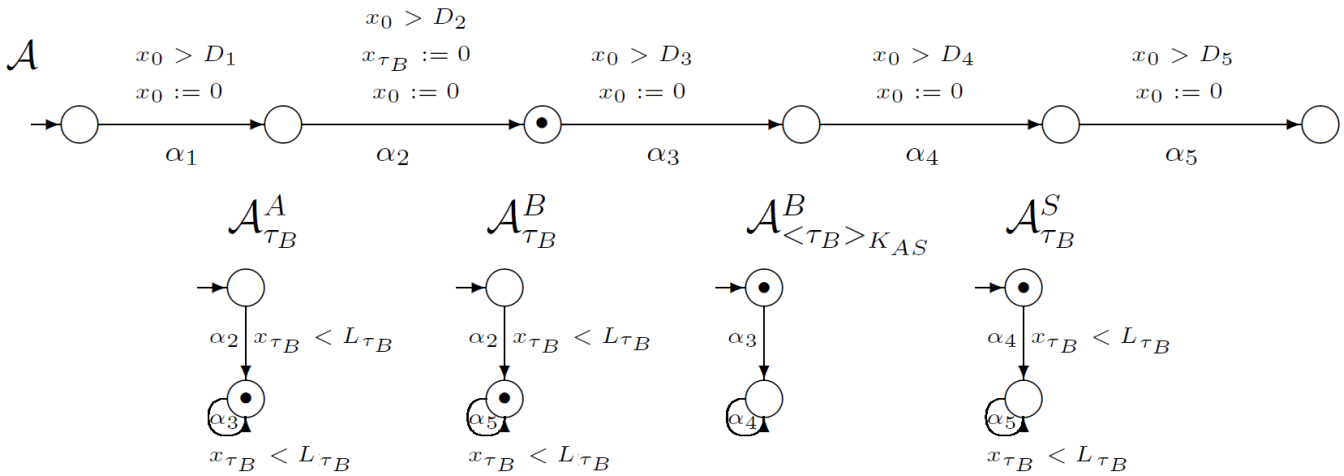

Figure 3. The second step of WooLamPi protocol.

Now, the third step of the WLP protocol can be executed. Considering Figure 4, we can see that the third transition in automaton A is synchronised with the loop in automaton $\mathcal{A}^A_{\tau_B}$ because A needs timestamp τ_B for the third step execution. Furthermore, automaton

$\mathcal{A}^{A}_{<\tau_B>_{K_{AS}}}$ is synchronised with automaton A. During this step, user B possesses knowledge about ciphertext $<\tau_B>_{K_{AS}}$.

Figure 5 shows that in the fourth step user B needs ciphertext $<\tau_B>_{K_{AS}}$ for executing this step. Additionally, the server S possesses a ticket τ_B after decryption of ciphertext $<\tau_B>_{K_{AS}}$. The four transition in automaton A is synchronised with the loop in automaton $\mathcal{A}^{A}_{<\tau_B>_{K_{AS}}}$ and with the automaton $\mathcal{A}^{S}_{\tau_B}$.

In the last step (Figure 6), the server uses knowledge about a ticked τ_B and the user B gets its ticket again (loop in automaton $\mathcal{A}^{B}_{\tau_B}$).

The method of automatic generation of automata and considered space of the users is given in [12].

The networks of synchronised timed automata for the SNEP protocol's one honest execution will be constructed similarly. We will consider these networks for both our proposed versions of this protocol.

Figure 7 shows a part of the network of synchronised timed automata for the first version of the SNEP protocol. This network consists of 22 execution automata and 67 knowledge automata. In the picture we placed one execution automaton and 14 knowledge automata. The execution automaton models execution of four steps of SNEP protocol, including time conditions. Note that on the first (α_1) transition the clock for timestamp τ_A is reset. In addition, on the second (α_2) transition, the clock for timestamp τ_B is reset.

The knowledge automaton models the changes in users knowledge about cryptographic objects. For example, the first knowledge automaton $\mathcal{A}^{A}_{\tau_A}$ models the process of acquisition and use of timestamp τ_A by user A. On the first transition (α_1) user A generates timestamp τ_A, so the automaton will change its state if the imposed time condition is met. In addition, on the second (α_2) and third (α_3) transitions, the loops are defined.

Figure 8 shows the network of synchronised timed automata for the second version of the SNEP protocol. This network consists of 22 execution automata and 58 knowledge automata. In the picture, we placed one execution automaton and 14 knowledge automata. In this case, the execution automaton models execution of six steps of the SNEP protocol, including time conditions. The clocks for the timestamps are reset on the same transitions as in Figure 7.

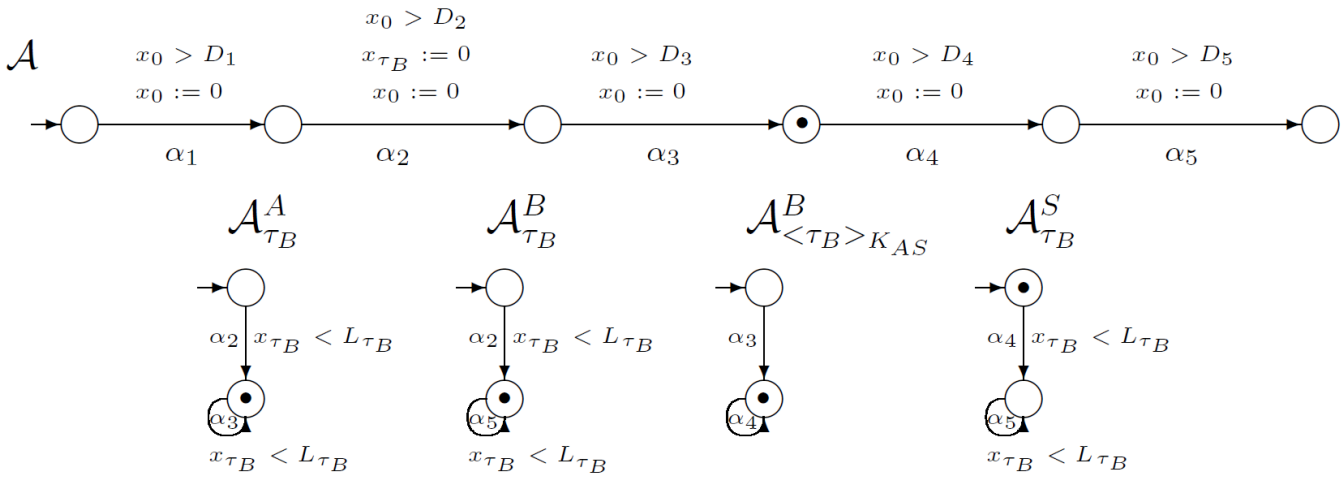

Figure 4. The third step of the WooLamPi protocol.

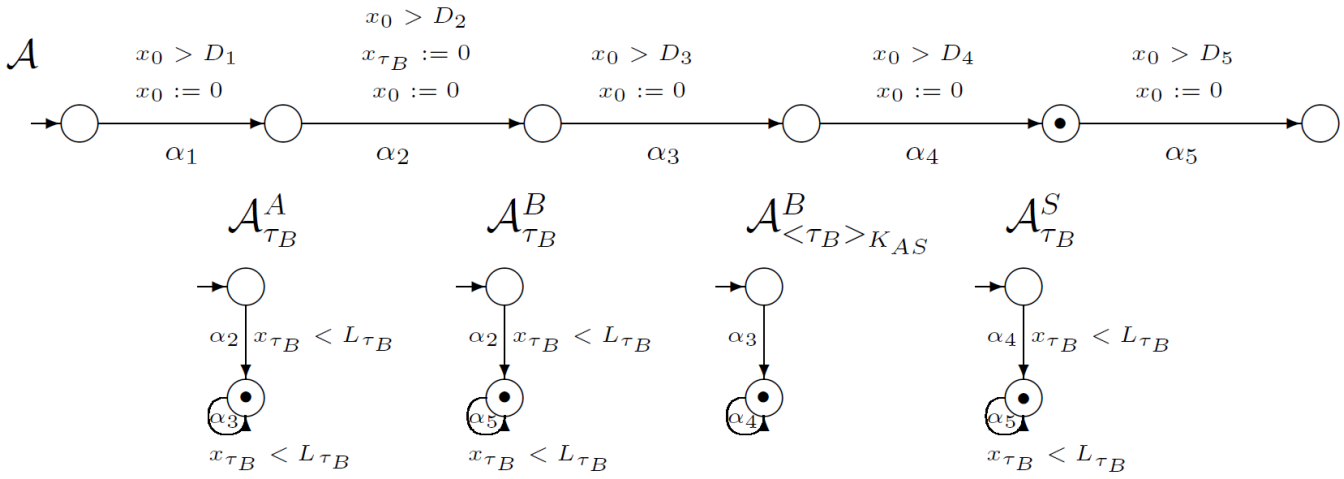

Figure 5. The fourth step of the WooLamPi protocol.

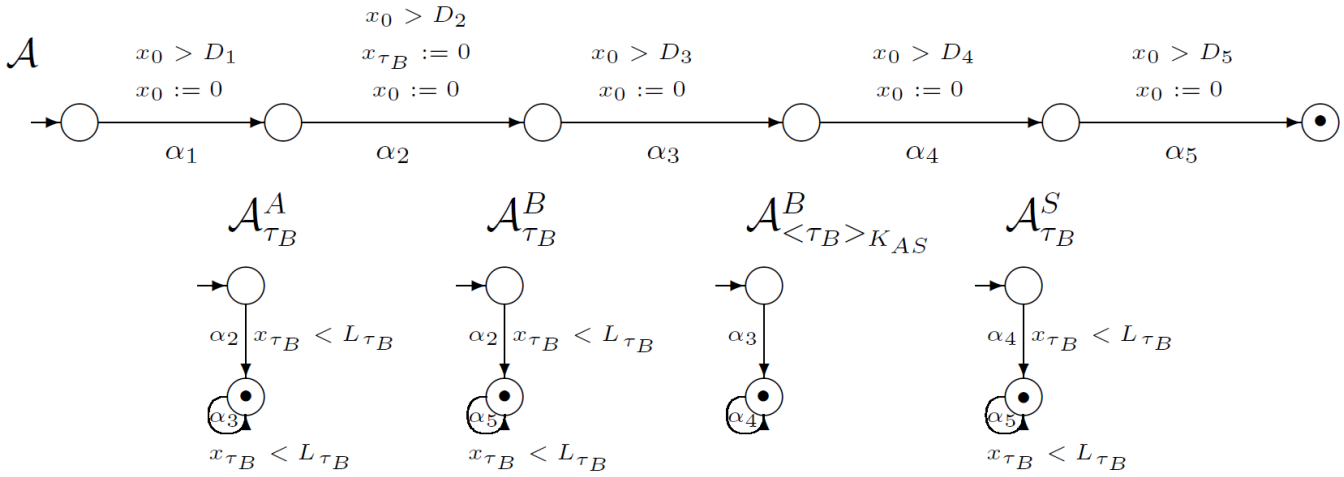

Figure 6. The fifth step of the WooLamPi protocol.

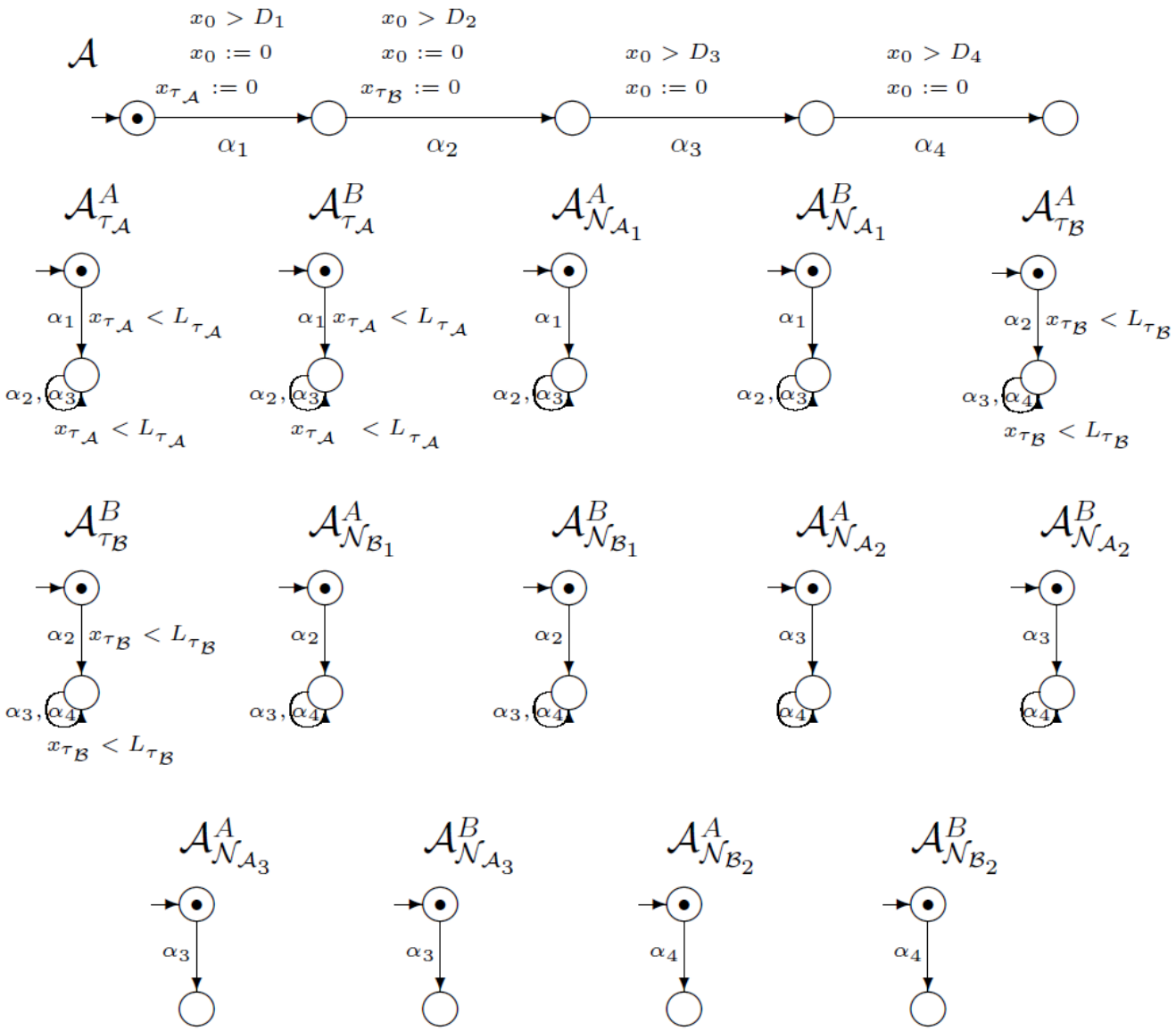

Figure 7. Network of synchronised timed automata for SNEP protocol v1.

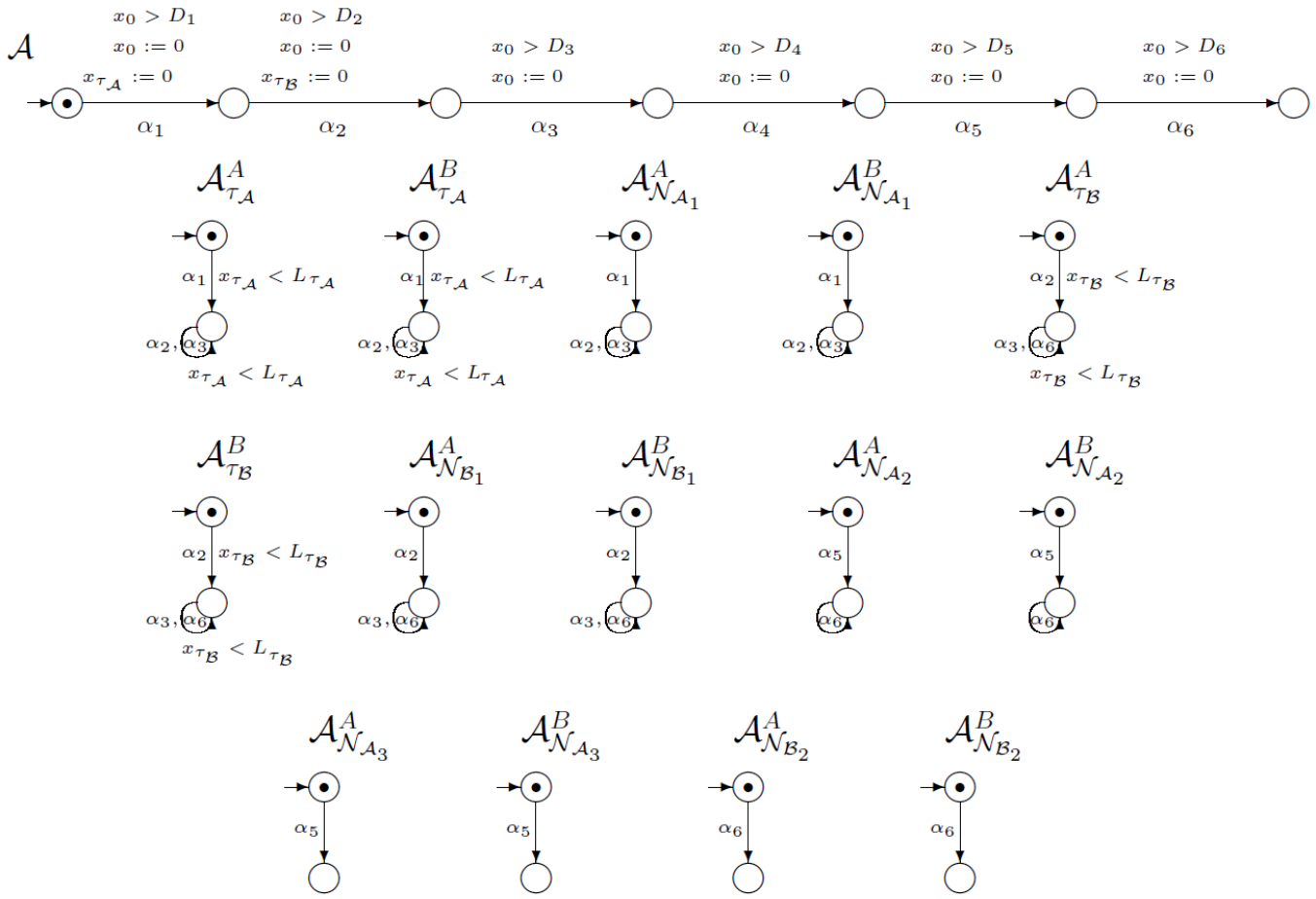

Figure 8. Network of synchronised timed automata for SNEP protocol v2.

5. Reachability Analysis

In this section we formally define the reachability problem for security protocols modelled as a network of timed automata and we present a solution to the problem which uses SMT-solvers and SAT-solvers. We begin with defining the problem and then we describe our solution.

The transition system (timed model) $C(\mathcal{A})$ of a timed automaton $\mathcal{A}$ usually has infinitely many states and infinitely many labels. However, the reachability problem of $C(\mathcal{A})$ can be reduced to the reachability problem for a transition system with finitely many states and finitely many labels.

Let c_{max} be the largest constant c such that some clock x is compared with c in some constraint appearing in a guard of $\mathcal{A}$. By $\hat{C}(\mathcal{A})$ we denote the transition system for a timed automaton which differs from $C(\mathcal{A})$ in the set of labels only: As the set of labels of $\hat{C}(\mathcal{A})$ we take the set $A \cup [0, c_{max} + 1]$.

The region equivalence (the equivalence relation $\simeq$) is defined over the set of all clock valuations for X. For two clock valuations v and v' in $\mathbb{R}^{|\mathcal{X}|}$, we say that $v \simeq v'$ iff for each $0 \leqslant j < n$, where n is a number of clocks, either $v(x_j) > c_j^{max}$ and $v'(x_j) > c_j^{max}$ or $v(x) \leqslant c_j^{max}$ and $v'(x) \leqslant c_j^{max}$ and $v(x) = v'(x)$.

It is a well-known fact, that the relation $\simeq$ is an equivalence relation, which gives rise to the construction of a finite abstract model.

The reachability problem for a network of the timed automata modelled by timed model $\hat{C}(\mathcal{A})$ is the question of whether for a given set of target locations, a state with a target location is reachable from some initial state. We assume that the set of target locations is described by a propositional formula expressing some property. To check the reachability

of a state satisfying the property by the BMC method, first, the transition relation of the model is unfolded iteratively to some depth k and encoded as a propositional formula (for SAT-based method) or a quantifier-free first order formula (or SMT-based method) of state variables. Next, the property is translated into a propositional/a quantifier-free first-order formula of the state variables and satisfiability of the conjunction of the two above formulae is checked by a SAT-solver or by an SMT-solver.

If the conjunction, denoted in the algorithm by β_k is satisfiable, one may conclude that a path to a target location was found. Otherwise, the value of k is incremented by 2, as time transitions do not change the global locations (Algorithm 1). The parameter n stands for the number of steps of a given protocol.

Algorithm 1 The standard BMC algorithm BMC for testing reachability

```
1:  procedure REACHABILITY
2:      k := 0
3:      loop
4:          result := checkSAT(β_k)
5:          if result = SATISFIABLE then
6:              return REACHABLE
7:          else if result = UNKNOWN then
8:              return UNKNOWN
9:          end if
10:         k := k + 2
11:         if k > 4 · n then
12:             return UNREACHABLE
13:         end if
14:     end loop
15: end procedure
```

The presented SAT and SMT encoding of the reachability problem for a network of timed automata is based on the SAT encoding presented in [40]. However, we extended the encoding using actions and we also defined a SMT-based encoding.

Let $\hat{C}(\mathcal{A})$ be a model. To formulate and solve the reachability problem for NTA, we have to define the unfolding of the transition relation to the depth $k \in \mathbb{N}$. For that purpose, we define a k-path to be a finite prefix of a path. Note that arbitrary state $q = (l, v)$ is reachable in $\hat{C}(\mathcal{A})$ iff it is reachable on a k-path, for some $k \geq 0$.

We define the formula $path_k(\overline{\mathbf{w}}_0, \ldots, \overline{\mathbf{w}}_k)$ which symbolically encodes all the k-paths starting at the initial state of $\hat{C}(\mathcal{A})$. The definition of the formula $path_k(\overline{\mathbf{w}}_0, \ldots, \overline{\mathbf{w}}_k)$ assumes that each concrete state $q \in Q$ of $\hat{C}(\mathcal{A})$ can be represented by a valuation of a symbolic state $\overline{\mathbf{w}} = ((w_1, v_1), \ldots, (w_n, v_n))$ that consists of symbolic local states. Each symbolic local state is a pair (w_j, v_j) of individual variables ranging over the natural numbers that consists of a location of the automaton j and a clock valuation. Similarly, each action can be represented by a valuation of a symbolic action $\overline{\mathbf{a}}$ that is a vector of the individual variables ranging over natural numbers.

In the case of SAT encoding, we use vectors (of the proper length) of propositional variables.

Let $\overline{\mathbf{w}}$ and $\overline{\mathbf{w}}'$ be two symbolic states, $\overline{\mathbf{a}}$ a symbolic action, and $\mathbf{d}$ a symbolic non-negative real number.

We assume definitions of the following quantifier-free first-order formulae: $I_q(\overline{\mathbf{w}})$ encodes the state q of the model $\hat{C}(\mathcal{A})$, $\mathcal{T}_{Act}(w, \bar{\mathbf{a}}, w')$ encodes an action transition, and $\mathcal{T}_\tau(w, \mathbf{d}, w')$ encodes a time transition in $\hat{C}(\mathcal{A})$.

Now for each even $k \in \mathbb{N}$ we can define the formula $path_k(\overline{\mathbf{w}}_0, \ldots, \overline{\mathbf{w}}_k)$ as:

$$\bigvee_{q \in \mathfrak{s}^0} I_q(\overline{\mathbf{w}}_0) \wedge \bigwedge_{\substack{j=0 \\ j \bmod 2=0}}^{k-2} \left(\mathcal{T}_\tau(\overline{\mathbf{w}}_j, \mathbf{d}, \overline{\mathbf{w}}_{j+1}) \wedge \mathcal{T}_{Act}(\overline{\mathbf{w}}_{j+1}, \bar{\mathbf{a}}_j, \overline{\mathbf{w}}_{j+2})\right).$$

Using the above formula and a quantifier-free first-order formula $reach(\overline{\mathbf{w}})$, which encodes the set of states satisfying a given property, we try to establish whether a state that satisfies $reach(\overline{\mathbf{w}})$ is reachable. We do this by checking the satisfiability of the following formula: $\psi_k = path_k(\overline{\mathbf{w}}_0, \ldots, \overline{\mathbf{w}}_k) \wedge \bigvee_{j=0}^{k} reach(\overline{\mathbf{w}}_j)$. The method described relies on the following theorem.

Theorem 1. *Let $\hat{C}(\mathcal{A})$ be a model and q be a concrete state. Then for every $k \in \mathbb{N}$, q is reachable in $\hat{C}(\mathcal{A})$ on a path of length k if, and only if, the formula ψ_k is satisfiable.*

The proof by induction on k is straightforward and is presented in [40].

We terminate the unfolding of the transition relation if either the formula ψ_k is satisfiable or it is impossible for a given SMT-solver to check satisfiability of the formula in question. We could also terminate if the value of k is equal to the reachability diameter of the system—the minimal number of steps required for reaching all the reachable states. Unfortunately, for many systems the diameter cannot be calculated and the estimates are too rough. It makes BMC incomplete in practice.

6. Experiments

For research, we used our tool described in [25,26,29,31,40–42]. Thanks to it, we generated protocol executions and the network of synchronised timed automata. The network models the protocol executions.

The tests were carried out on a computer unit with the Linux Arch operating system, Intel Core i7-3770 processor, and 32 GB RAM. For satisfiability checking, we used the Yices SAT-solver and the Yices SMT-solver in version 2.6.2 [43].

The network of synchronised timed automata for the WLP protocol consisted of 22 executions automata and 21 knowledge automata. The first two execution automata model honest executions. In those executions, only honest users appears (A, B, and server S). The first automaton represents the execution initiated by A (communication between A, B, and S). The second automaton model the execution initiated by B (communication between B, A, and S). The remaining automata model either executions with an intruder who acts as themself or executions in which the intruder impersonates one of the honest users. In automata 3–12, the intruder either acts as themself or impersonates A, and in automata 13–22, they either act as themself or impersonates B. We have assumed that the intruder could not impersonate a trusted S server.

The network of synchronised timed automata for the SNEP protocol (both versions) consists of 22 executions automata. For the four-step version of the SNEP protocol, the network consists of 47 knowledge automata and 49 knowledge automata for the six-step version. Similarly to the WLP protocol, the first two knowledge automata models executions only with the honest users (A, B). The first automaton represents the execution initiated by A (communication between A and B), while the second automaton models the execution initiated by B (communication between B and A). The remaining automata model either executions with the intruder who acts as themself or executions in which the intruder impersonated one of the honest users. The automata with numbers 3 and 11 model executions with the intruder who acts as themself and appears on the position of the user A. Automata 4–10 and 12 model executions in which the intruder impersonates user

A. The automata with numbers 13 and 21 model executions with the intruder who acts as themself and appears on the position of user B. Automata 14–20 and 22 model executions in which the intruder impersonates user B.

Each path in the network of timed automata in which the intruder comes into possession of confidential data is interpreted as a full attack. Paths in which they only stand in the middle of communication impersonating individual users will be interpreted as MiTM behaviour. This behaviour is also undesirable. The mere detection of the presence of the eavesdropping party is important from a security point of view. Time parameters will help us track and, in the long run, limit the possibilities of the intruder.

These studies were related to checking the reachability of states that model protocols executions, including time parameters. For the WLP protocol, we set three different values of delays and two lifetimes values. For the SNEP protocol (both versions), we set five different values of delays and two lifetimes values.

We present the obtained results for the SAT-based and the SMT-based methods in Table 3. We tested the reachability for the last location in each execution automata.

Table 3. Experimental results for SAT- and SMT-based methods and protocols (reachability of the last location in each automaton).

Protocol	Yices-SAT		Yices-SMT	
	Time (s)	**Memory (kB)**	**Time (s)**	**Memory (kB)**
Four-step version of the SNEP	151.79	21516	129.05	18604
Six-step version of the SNEP	418.62	45748	307.16	25544
WLP	271.7	32840	102.86	18384

6.1. WLP Protocol

For the WLP protocol (Figure 9 and 10), we found two paths indicating a Man in the Middle behaviour (not a full attack). It means that we found the paths on which locations 71 or 131 were reachable. We also observed the availability of two paths that represent an honest execution (the paths ending in one of the locations 5 or 11) and 10 paths representing execution with the intruder (the paths ending in one of the locations 17, 29, 35, 41, 53, 77, 83, 89, 101, or 113). By executions with the intruder, we mean executions in which the intruder acts as a regular user (the paths ending in one of the locations 17, 41, 77, or 101) or tries to impersonate honest users without success (the paths ending in one of the locations 29, 35, 53, 83, 89, or 113).

For the WLP protocol, we observed that values $L_1 = 8$ and $L_2 = 8$ are minimal values which allow to perform both honest executions (these values do not allow a Man in the Middle behaviour). The minimal values that allow to appear a Man in the Middle behaviour for locations 71 and 131 are $L_1 = 10$ and $L_2 = 10$.

The honest executions are possible:

- For location 5 when $L_1 \geqslant max(D_1, D_3) + D_2$ and any L_2,
- For location 11 when $L_2 \geqslant max(D_1, D_3) + D_2$ and any L_1.

The executions which represent a Man in the Middle behaviour are possible:

- For location 71 when $L_2 \geqslant max(D_1 + D_2 + D_3, 2 \cdot D_1 + D_2)$ and any L_1,
- For location 131 when $L_1 \geqslant max(D_1 + D_2 + D_3, 2 \cdot D_1 + D_2)s$ and any L_2.

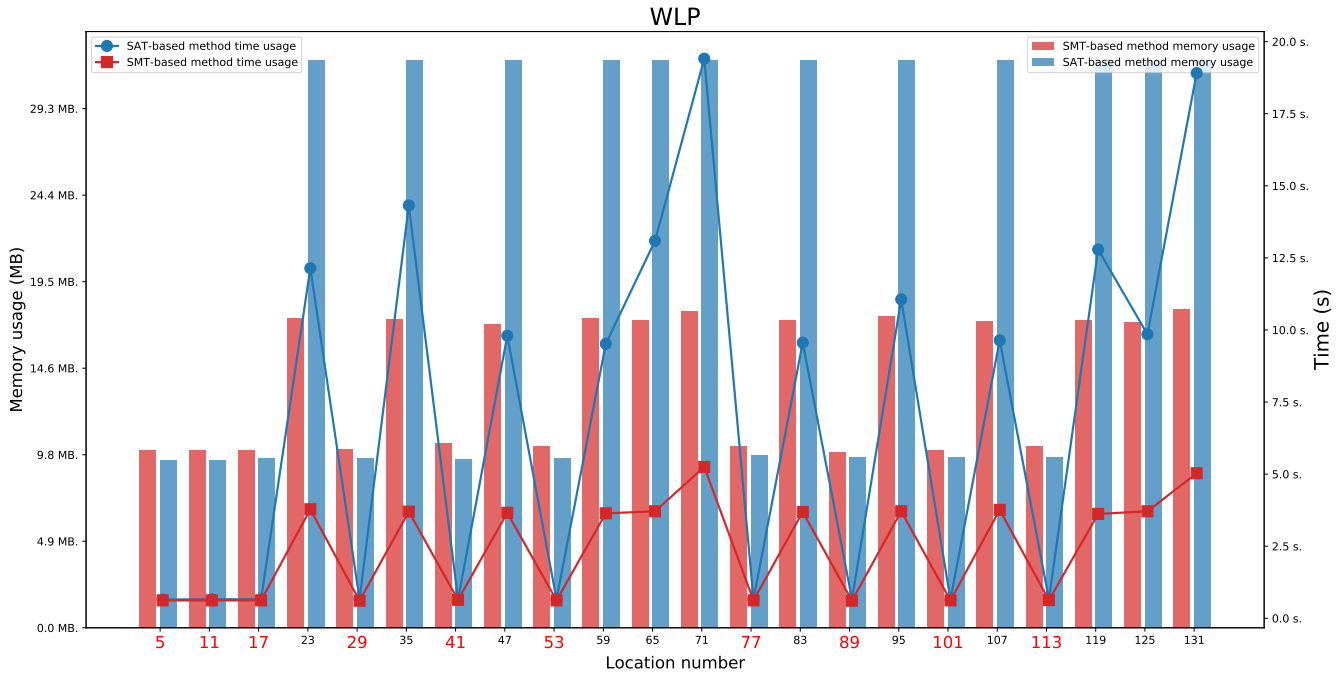

Figure 9. The WLP: Time and memory usage for checking the last locations' reachability in each execution automaton for both lifetime values equal to 8. The reachable states are marked in red.

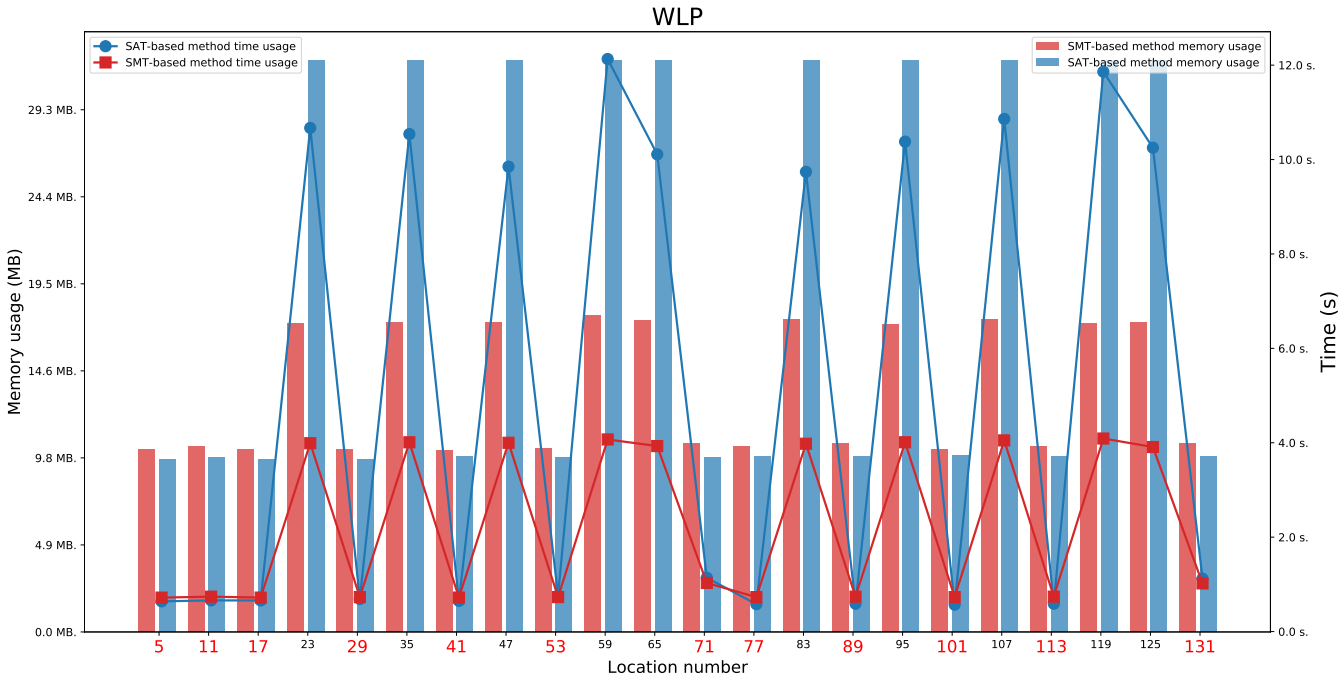

Figure 10. The WLP: Time and memory usage for checking the last locations' reachability in each execution automaton for both lifetime values equal to 10. The reachable states are marked in red.

6.2. Four-Step Version of the SNEP Protocol

For the four-step version of the SNEP protocol (Figure 11 and 12), we found two paths indicating a Man in the Middle behaviour (the paths ending in one of the locations 49 or 99). The full attack was not found. We also observed the availability of two paths that represent an honest execution (the paths ending in one of the locations 4 or 9) and six paths

that represent execution with the intruder (the paths ending in one of the locations 29, 54, 59, 79, 104, or 109).

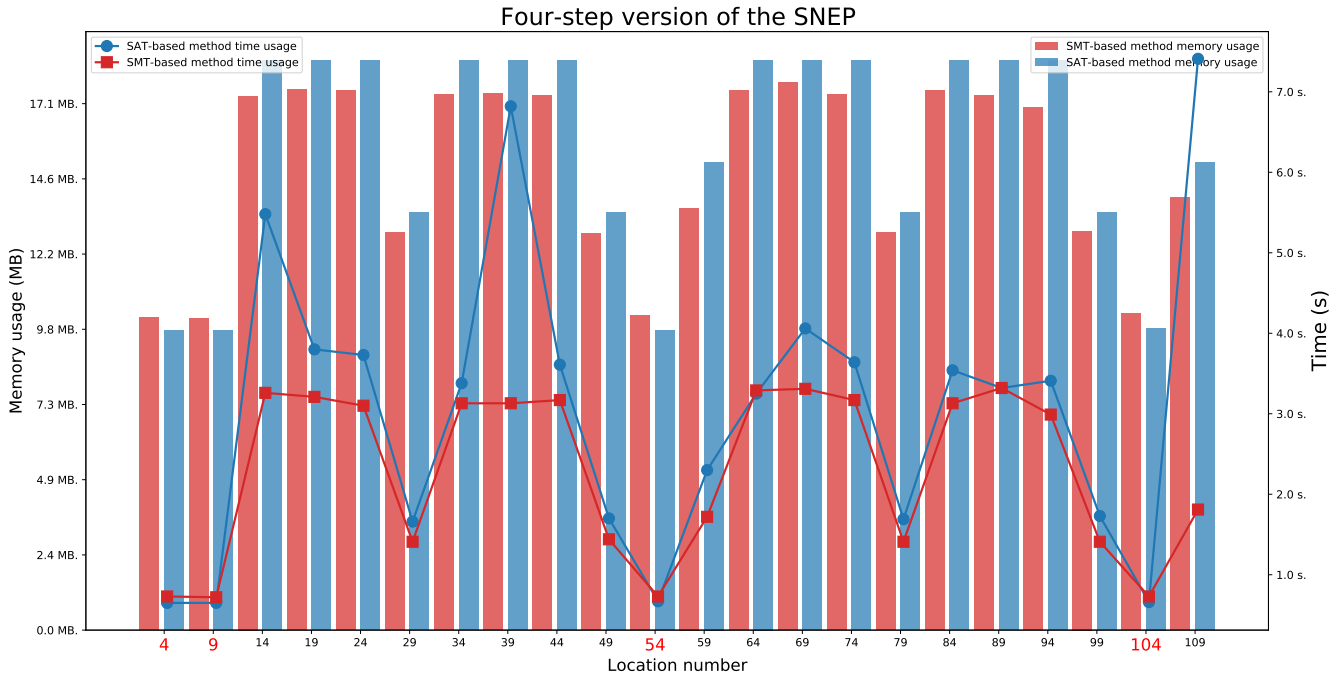

Figure 11. The four-step version of the SNEP: Time and memory usage for checking the last locations' reachability in each execution automaton for the lifetime value equal to 4. The reachable states are marked in red.

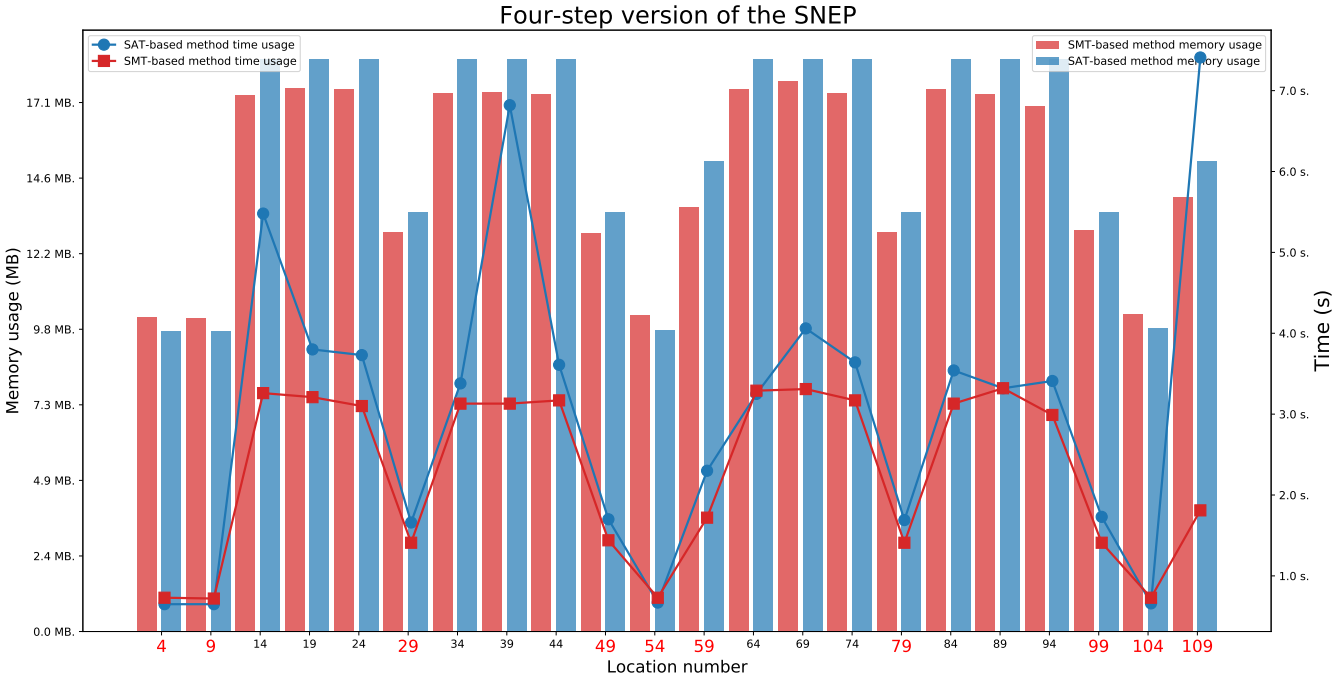

Figure 12. The four-step version of the SNEP: Time and memory usage for checking the last locations' reachability in each execution automaton for the lifetime value equal to 7. The reachable states are marked in red.

As before, we set five different values of delays and two lifetimes values for this protocol. We have checked for which time parameter values the protocol executions are feasible. We observed that values $L_1 = 4$ and $L_2 = 4$ are minimal values to allow for both honest executions. These values also do not allow for a Man in the Middle behaviour. The

minimal values that allow for a Man in the Middle behaviour for location 49 are $L_1 = 3$ and $L_2 = 7$, and for location 99 are $L_1 = 7$ and $L_2 = 3$.

The honest executions are possible:

- For location 4 when $L_1 \geqslant D_1$ and $L_2 \geqslant D_1 + D_2$,
- For location 9 when $L_1 \geqslant D_1 + D_2$ and $L_2 \geqslant D_1$.

The executions which represent a Man in the Middle behaviour are possible:

- For location 49 when $L_1 \geqslant max(D_1, D_2)$ and $L_2 \geqslant D_1 + D_2 + D_3$,
- For location 99 when $L_1 \geqslant D_1 + D_2 + D_3$ and $L_2 \geqslant max(D_1, D_2)$.

6.3. Six-Step Version of the SNEP Protocol

For the six-step version of the SNEP protocol (Figure 13 and 14), we found two paths indicating a Man in the Middle behaviour, without full attack (the paths ending in one of the locations 69 or 139), two paths that represent an honest execution (the paths ending in one of the locations 6 or 13), and 10 paths that represent execution with the intruder (the paths ending in one of the locations 20, 34, 48, 76, 83, 90, 104, 118, 146, or 153).

As in the case of the four-step version of the SNEP protocol, we checked for which time parameter values the protocol executions are feasible. We observed that values $L_1 = L_2 = 9$ are minimal values to allow for both honest executions. These values also do not allow for a Man in the Middle behaviour. The minimal values that allow for a Man in the Middle behaviour are $L_1 = L_2 = 10$ for locations 69 or 139.

We also observed that the honest executions are possible:

- For location 6 when $L_1 \geqslant D_1$ and $L_2 \geqslant D_2 + D_4$,
- For location 13 when $L_1 \geqslant D_2 + D_4$ and $L_2 \geqslant D_2$.

The executions which represent a Man in the Middle behaviour are possible:

- For location 69 when $L_1 \geqslant max(D_1, D_2)$ and $L_2 \geqslant D_1 + D_2 + D_3$,
- For location 139 when $L_1 \geqslant D_1 + D_2 + D_3$ and $L_2 \geqslant max(D_1, D_2)$.

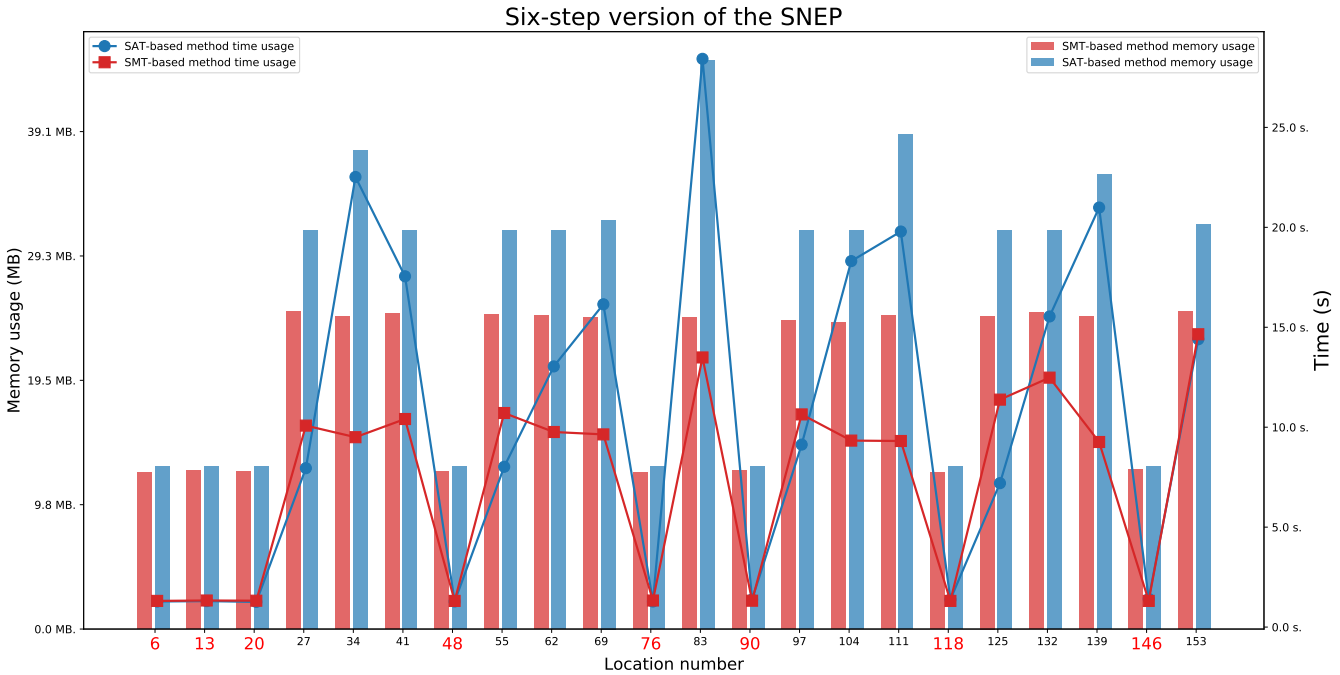

Figure 13. The six-step version of the SNEP: Time and memory usage for checking the last locations' reachability in each execution automaton for lifetime values equal to 9. The reachable states are marked in red.

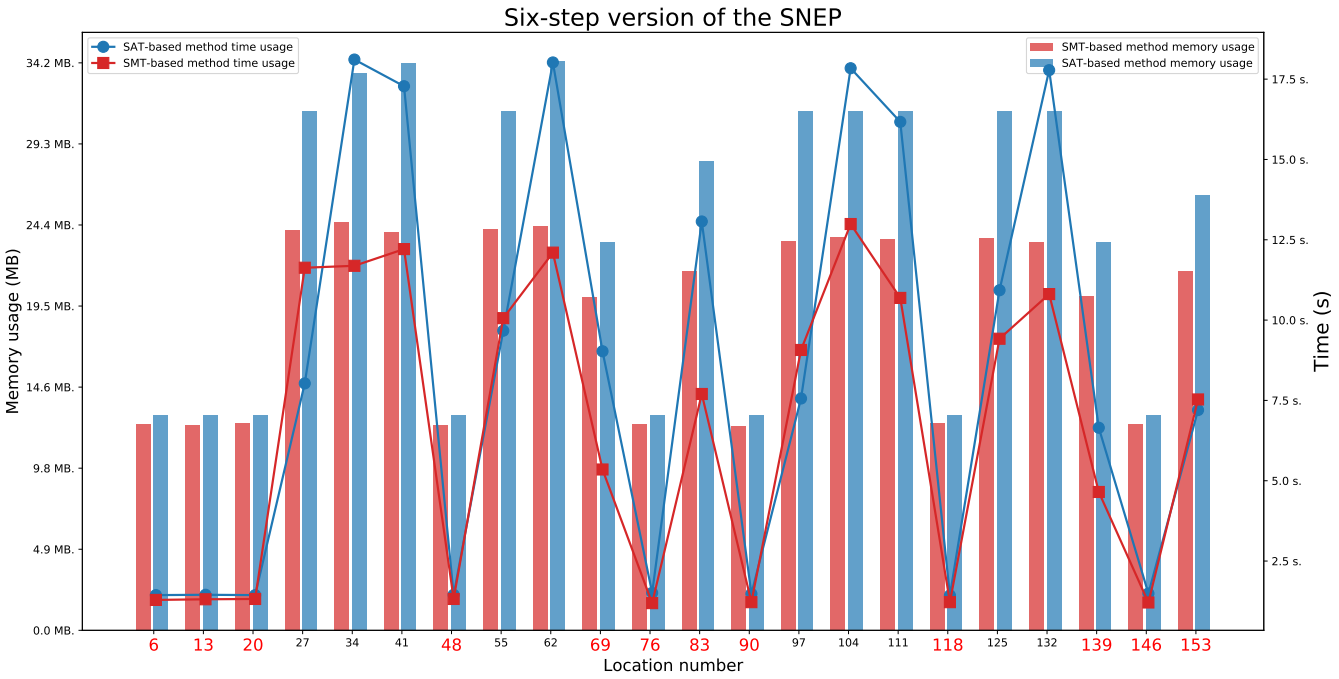

Figure 14. The six-step version of the SNEP: Time and memory usage for checking the last locations' reachability in each execution automaton for lifetime values equal to 10. The reachable states are marked in red.

7. Conclusions

In this article, we investigated the correctness of security protocols dedicated to keeping sensor devices communication safe. We presented how we can verify important properties of security protocols using formal mathematical modelling. Accurate modelling allows us to verify whether the protocol works at all, i.e., honest executions reach the final states in the network of automata. We can also check if it is susceptible to an attack, i.e., in the network of automata with the intruder, states representing knowledge about sensitive protocol elements—e.g., nonces are available. The presented model provides an extended security analysis where we investigated time properties. We took into account network delays and the lifetime of timestamps used in the protocol. Those parameters allowed us to detect unwanted network behaviour in which the intruder wants to attack the protocol. Even if a full attack is not possible, we could set the values of time parameters so that even its presence is detectable and the interference is prevented.

In our presentation, we used as examples the SNEP protocol developed for sensors devices communication and different versions of the WooLamPi Protocol. We showed protocols' schemes and their formal models. These models and the investigated security properties were translated into propositional/quantifier-free first-order formulae. Next, we performed our experiments using our dedicated protocols' verification tools and SAT and SMT solvers. Finally, we showed experimental results. Our analysis has shown that administrators can select time parameters values in such a way to make the protocol secure.

We have presented our method using relatively simple protocols however, new, more complex protocols have been proposed for securing sensor devices communication [44,45]. Now we plan to take into account and investigate more complex systems. For this, we can use our experience gained during the verification of other systems [17,46].

In the case of protocols designed for sensors, a completely different aspect worth analysing is energy consumption [47]. The limitations of sensors due to the size of resources and batteries make it worth knowing how much energy a given attack requires or how much energy the sensor will use to increase the level of communication security. It may be another interesting research direction.

Author Contributions: Conceptualization, S.S., O.S.-L., M.K., A.M.Z., and A.Z.; Formal analysis, S.S., O.S.-L., M.K., A.M.Z., and A.Z.; Methodology, S.S., O.S.-L., and M.K.; Software, S.S., A.M.Z., and A.Z.; Writing—original draft, S.S., O.S.-L., M.K., and A.M.Z.; Writing—review & editing, S.S., O.S.-L., M.K., A.M.Z., and A.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: https://cloud.icis.pcz.pl/s/werzTT3tRPmEJQz, accessed on 27 April 2021.

Acknowledgments: 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.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zeng, Y.; Lin, M.; Guo, S.; Shen, Y.; Cui, T.; Wu, T.; Zheng, Q.; Wang, Q. MultiFuzz: A Coverage-Based Multiparty-Protocol Fuzzer for IoT Publish/Subscribe Protocols. *Sensors* **2020**, *20*, 5194. [CrossRef]
2. Mastilak, L.; Galinski, M.; Helebrandt, P.; Kotuliak, I.; Ries, M. Enhancing Border Gateway Protocol Security Using Public Blockchain. *Sensors* **2020**, *20*, 4482. [CrossRef]
3. Moreno-Cruz, F.; Toral-López, V.; Escobar-Molero, A.; Ruíz, V.U.; Rivadeneyra, A.; Morales, D.P. treNch: Ultra-Low Power Wireless Communication Protocol for IoT and Energy Harvesting. *Sensors* **2020**, *20*, 6156. [CrossRef] [PubMed]
4. Yu, D.; Li, P.; Chen, Y.; Ma, Y.; Chen, J. A Time-efficient Multi-Protocol Probe Scheme for Fine-grain IoT Device Identification. *Sensors* **2020**, *20*, 1863. [CrossRef] [PubMed]
5. Steingartner, W.; Galinec, D.; Kozina, A. Threat Defense: Cyber Deception Approach and Education for Resilience in Hybrid Threats Model. *Symmetry* **2021**, *13*, 597. [CrossRef]
6. Lowe, G. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Inf. Process. Lett.* **1995**, *56*, 131–133. [CrossRef]
7. Lowe, G. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*; Springer: London, UK, 1996; pp. 147–166.
8. Burrows, M.; Abadi, M.; Needham, R. A Logic of Authentication. *ACM Trans. Comput. Syst.* **1990**, *8*, 18–36. [CrossRef]
9. Armando, A.; Basin, D.; Boichut, Y.; Chevalier, Y.; Compagna, L.; Cuellar, J.; Drielsma, P.H.; Heám, P.C.; Kouchnarenko, O.; Mantovani, J.; et.al. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Computer Aided Verification, Scotland, UK, 6–10 July 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 281–285.
10. Cremers, C.; Mauw, S. *Operational Semantics and Verification of Security Protocols*; Information Security and Cryptography; Springer: Berlin/Heidelberg, Germany, 2012.
11. Blanchet, B. Modeling and verifying security protocols with the applied Pi-Calculus and ProVerif. *Found. Trends Priv. Secur.* **2016**, *1*, 1–135. [CrossRef]
12. Kurkowski, M.; Penczek, W. Applying Timed Automata to Model Checking of Security Protocols. In *Handbook of Finite State Based Models and Applications*; CRC Press: Boca Raton, FL, USA, 2016; pp. 223–254. [CrossRef]
13. Mödersheim, S.; Bruni, A. AIF-ω: Set-Based Protocol Abstraction with Countable Families. In Proceedings of the Principles of Security and Trust—5th International Conference, POST 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, 2–8 April 2016; Volume 9635, pp. 233–253
14. Basin, D.A.; Cremers, C.; Meadows, C.A. Model Checking Security Protocols. In *Handbook of Model Checking*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 727–762.
15. Hess, A.V.; Mödersheim, S. Formalizing and Proving a Typing Result for Security Protocols in Isabelle/HOL. In Proceedings of the 2017 IEEE 30th Computer Security Foundations Symposium (CSF), Santa Barbara, CA, USA, 21–25 August 2017; pp. 451–463. [CrossRef]
16. Hess, A.; Mödersheim, S. A Typing Result for Stateful Protocols. In Proceedings of the 2018 IEEE 31st Computer Security Foundations Symposium (CSF), Oxford, UK, 9–12 July 2018; pp. 374–388. [CrossRef]
17. Bartłomiejczyk, M.; ElFray, I.; Kurkowski, M. Multifactor Authentication Protocol in a Mobile Environment. *IEEE Access* **2019**, *7*, 157185–157199.
18. Cremers, C.; Dehnel-Wild, M.; Milner, K. Secure authentication in the grid: A formal analysis of DNP3 SAv5. *J. Comput. Secur.* **2019**, *27*, 203–232. [CrossRef]
19. Alur, R.; Dill, D.L. The Theory of Timed Automata. In Proceedings of the Real-Time: Theory in Practice, REX Workshop, Mook, The Netherlands, 3–7 June 1991; Volume 600, pp. 45–73. [CrossRef]
20. Koymans, R. Specifying Real-Time Properties with Metric Temporal Logic. *Real-Time Syst.* **1990**, *2*, 255–299. [CrossRef]
21. Penczek, W.; Pólrola, A. *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 20.

22. Jakubowska, G.; Penczek, W. Modelling and Checking Timed Authentication of Security Protocols. *Fundam. Inform.* **2007**, *79*, 363–378.
23. Kurkowski, M. *Formalne Metody Weryfikacji Własności Protokołów Zabezpieczających w Sieciach Komputerowych*; Informatyka-Akademicka Oficyna Wydawnicza EXIT; Akademicka Oficyna Wydawnicza Exit: Warsaw, Poland, 2013.
24. Szymoniak, S.; Kurkowski, M.; Piątkowski, J. Timed models of security protocols including delays in the network. *J. Appl. Math. Comput. Mech.* **2015**, *14*, 127–139. [CrossRef]
25. Szymoniak, S.; Siedlecka-Lamch, O.; Kurkowski, M. Timed Analysis of Security Protocols. In *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part II*; Springer International Publishing: Cham, Switzerland, 2017; pp. 53–63.
26. Zbrzezny, A.M.; Szymoniak, S.; Kurkowski, M. Efficient Verification of Security Protocols Time Properties Using SMT Solvers. In Proceedings of the International Joint Conference: 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on EUropean Transnational Education (ICEUTE 2019), Seville, Spain, 13–15 May 2019; pp. 25–35. [CrossRef]
27. Zbrzezny, A.M.; Zbrzezny, A.; Szymoniak, S.; Siedlecka-Lamch, O.; Kurkowski, M. VerSecTis—An Agent based Model Checker for Security Protocols. In Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, 9–13 May 2020; pp. 2123–2125.
28. Benerecetti, M.; Cuomo, N.; Peron, A. TPMC: A Model Checker For Time-Sensitive Security Protocols. *J. Comput.* **2009**, *4*, 366–377. [CrossRef]
29. Szymoniak, S.; Siedlecka-Lamch, O.; Kurkowski, M. On Some Time Aspects in Security Protocols Analysis. In *International Conference on Computer Networks*; Springer: Cham, Switzerland, 2018; pp. 344–356.
30. Szymoniak, S. The Impact of Time Parameters on the Security Protocols Correctness. In Proceedings of the Computer Networks—25th International Conference, CN 2018, Gliwice, Poland, 19–22 June 2018; Volume 860, pp. 333–343.
31. Szymoniak, S. Modeling and Verification of Security Protocols Including Delays in the Network. Ph.D. Thesis, Czestochowa University of Technology, Czestochowa, Poland, 2017.
32. Li, L.; Sun, J.; Liu, Y.; Sun, M.; Dong, J. A Formal Specification and Verification Framework for Timed Security Protocols. *IEEE Trans. Softw. Eng.* **2018**, *44*, 725–746. [CrossRef]
33. Tobarra, L.; Cazorla, D.; Cuartero, F. Formal Analysis of Sensor Network Encryption Protocol (SNEP). In Proceedings of the 2007 IEEE International Conference on Mobile Adhoc and Sensor Systems, Pisa, Italy, 8–11 October 2007; pp. 1–6.
34. Woo, T.Y.C.; Lam, S.S. A Lesson on Authentication Protocol Design. *SIGOPS Oper. Syst. Rev.* **1994**, *28*, 24–37. [CrossRef]
35. Perrig, A.; Szewczyk, R.; Tygar, J.D.; Wen, V.; Culler, D.E. SPINS: Security Protocols for Sensor Networks. *Wirel. Netw.* **2002**, *8*, 521–534. [CrossRef]
36. Bernstein, D.J. The Salsa20 Family of Stream Ciphers. In *New Stream Cipher Designs—The eSTREAM Finalists*; Robshaw, M.J.B., Billet, O., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 4986, pp. 84–97.
37. Lara, E.; Aguilar, L.; Garcia, J.A.; Sanchez, M.A. A Lightweight Cipher Based on Salsa20 for Resource-Constrained IoT Devices. *Sensors* **2018**, *18*, 3326. [CrossRef]
38. Fukushima, K.; Xu, R.; Kiyomoto, S.; Homma, N. Fault Injection Attack on Salsa20 and ChaCha and a Lightweight Countermeasure. In Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS, Sydney, Australia,1–4 August 2017; pp. 1032–1037. [CrossRef]
39. Baier, C.; Katoen, J.P. *Principles of Model Checking*; MIT Press: Cambridge, MA, USA, 2008; pp. 1–975.
40. Zbrzezny, A. SAT-Based Reachability Checking for Timed Automata with Diagonal Constraints. *Fundam. Inf.* **2005**, *67*, 303–322.
41. Szymoniak, S. KaoChow Protocol Timed Analysis. In *International Multi-Conference on Advanced Computer Systems*; Springer: Cham, Switzerland, 2018; pp. 346–357.
42. Szymoniak, S. Security protocols analysis including various time parameters. *Math. Biosci. Eng.* **2021**, *18*, 1136–1153. [CrossRef]
43. Dutertre, B. Yices 2.2. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Computer Aided Verification, Vienna, Austria, 18–22 July 2014*; Springer: Cham, Switzerland, 2014; Volume 8559, pp. 737–744.
44. Guo, H.; Gao, Y.; Xu, T.; Zhang, X.; Ye, J. A secure and efficient three-factor multi-gateway authentication protocol for wireless sensor networks. *Ad Hoc Netw.* **2019**, *95*, 101965. [CrossRef]
45. Ali, Z.; Ghani, A.; Khan, I.; Chaudhry, S.A.; Islam, S.H.; Giri, D. A robust authentication and access control protocol for securing wireless healthcare sensor networks. *J. Inf. Secur. Appl.* **2020**, *52*, 102502. [CrossRef]
46. Siedlecka-Lamch, O.; El Fray, I.; Kurkowski, M.; Pejaś, J. Verification of Mutual Authentication Protocol for MobInfoSec System. In *Lecture Notes in Computer Science, Proceedings of the Computer Information Systems and Industrial Management, Warsaw, Poland, 24–26 September 2015*; Saeed, K., Homenda, W., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 461–474.
47. Potlapally, N.R.; Ravi, S.; Raghunathan, A.; Jha, N.K. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Trans. Mob. Comput.* **2006**, *5*, 128–143. [CrossRef]