Abstract. In this paper, we focused on a problem of authentication on low-cost devices. We have proposed a new light-weight protocol for mutual authentication of communication entities with non-repudiation of realized events. The protocol is simple and suitable for implementation on low-cost devices. Non-repudiation of realized events is achieved by involving a Trusted Third Party (TTP) to the communication. The proposed protocol uses only an appropriate light-weight hash function and pre-shared secret data. Security of the proposed protocol was verified by the BAN (Burrows-Abadi-Needham) logic.

Keywords

Light-weight mutual authentication, hash functions, non-repudiation, Trusted Third Party, Internet of Things

1. Introduction

In recent years, the wireless communication has been used in many areas. One important area using the wireless communication is IoT (Internet of Things) which represents a type of network where low-cost devices linked to the internet perform information exchange. Low-cost devices represent computationally, memory and power constrained devices. IoT is used in smart homes, smart hospitals, smart cities, etc. According to [1], it is expected that IoT, which excludes PCs, tablets and smartphones, will grow to 24 billion units installed in 2020. This is a huge number of devices. Data created by these devices involved in the IoT must be secured in order to be credible. IoT usually uses low-cost devices to collect and exchange data from sensors and other equipments by Wireless Sensor Network (WSN), Radio Frequency Identification (RFID) technology, 3G and 4G mobile connections, Wi-Fi, WiMAX and etc.

Common cryptographic algorithms as the asymmetric cryptosystem RSA (Rivest, Shamir, Adleman) with the modulus size 2048 bits or the symmetric cipher AES (Advanced Encryption Standard) with the key size 256 bits cannot be implemented on low-cost devices since these algorithms are demanding in terms of the computing power and memory resources. For that reason, security on low-cost devices is ensured by algorithms from light-weight cryptography. Light-weight cryptography uses algorithms which have small demands to the computing power and the memory resources. In most cases, light-weight cryptography uses symmetric cryptography to ensure security. Symmetric cryptography is generally less computationally and memory demanding in comparison with asymmetric cryptography. The basic terms in security are authentication, integrity, confidentiality and non-repudiation. Authentication ensures a verification of authenticity of communication entities. Integrity ensures a verification of data integrity. Confidentiality ensures that secret data cannot be available to unauthorized entities. Non-repudiation ensures that an entity cannot deny some fact which was realized in the past.

In this paper, we focus on authentication on low-cost devices which are used in IoT. We have proposed a new light-weight protocol for mutual authentication of communication entities with non-repudiation of realized events. Non-repudiation is ensured by involving a TTP to the communication. In Sec. 2 an overview of the most widely used light-weight authentication protocols is given. The principle of the proposed protocol is described in Sec. 3 and its formal security analysis is performed in Sec. 4. Demands to computing resources and transmitted data in the protocol are described in Sec. 5. Finally, the paper is concluded in Sec. 6.

2. State of the Art

Many authentication protocols for low-cost devices were proposed. There are various different ways how to create a light-weight authentication protocol. The authors of [2], [3] presented an authentication protocol suitable for implementation in low-cost RFID which uses Lattice based cryptography. In the papers [4], [5], the authors use Elliptic Curve Cryptography (ECC) for authentication. The authors of [6] presented an authentication scheme for WSN which is based on implicit certificates and it provides application level end-to-end security. In the articles [7], [8], light-weight authentication protocols which use McEliece public cryptography were presented. The authors of [9] and [10] designed light-weight authentication protocols for RFID which use an error correction code. The authors of [11] use the
Fermat Number Transform (FNT) and the Chinese Remainder Theorem (CRT) for light-weight authentication. In [12], a novel authentication protocol for RFID tags using shared pseudonyms and Cyclic Redundancy Check (CRC) to achieve a reader to tag authentication was proposed. More suggestions of light-weight authentication protocols using CRC were described in [13], [14]. The authors of [15] presented a lightweight authentication protocol which uses a ring variant of the LPN (Learning Parity with Noise) problem. In the paper [16], a lightweight authentication protocol for RFID using a stream cipher was described. The authors of [17] proposed a lightweight message authentication scheme for smart grids which uses the Diffie-Hellman protocol to establish the shared session key and a hash function for authentication. Another lightweight authentication protocol using a hash function was presented in [18]. In [19], a lightweight authentication protocol for RFID using pseudonyms and simple bitwise operations as AND, OR, XOR and bitwise rotation was described. The authors of [20] presented a lightweight authentication protocol which uses Physical Unclonable Functions (PUF), Linear Feedback Shift Registers (LFSR) and XOR operations. In [21], a lightweight authentication protocol which uses a PUF and the Hopper Blum (HB) protocol was described.

Presented lightweight authentication protocols often focus on concrete low-cost devices which determine their security. Simple authentication schemes using CRC and bitwise logical operations do not provide integrity of transmitted data. Authentication protocols which use a stream cipher can produce a lower level of diffuse which facilitates cryptanalysis. Authentication protocols built on elliptic curves can be broken using the Shor’s algorithm [22] in the case of constructing a universal quantum computer. Due to advances in the quantum area, the future use of these authentication protocols seems unpromising. The protection against attacks led from universal quantum computers providing protocols which use Lattice based cryptography, McEliece cryptography, hash functions or generally symmetric cryptography. Hash functions have small demands to the computing power and memory resources while provide integrity of data and robust security to input data, therefore, they are the most used cryptographic primitive on low-cost devices to ensure authentication. PUF represent a new perspective way how to ensure authentication on low-cost devices. PUF represent an alternative to common storage secret keys in nonvolatile memories. PUF utilize manufacturing heterogeneities and differences of components of a physical device to generate random outputs (secret keys) on the fly. The generated output of PUF is called a hardware fingerprint of the device. The advantages of PUF are reduction of a price and increasing of security. The main disadvantage of PUF is a noise in generated outputs. These errors in PUF outputs are usually corrected by an error correction code. The disadvantage of an error correction code is that it requires a permanent memory which increases the price of the device.

3. Our Proposal of Authentication Protocol

Based on the analysis of lightweight authentication protocols provided in Sec. 2, we have proposed a new lightweight mutual authentication protocol with non-repudiation of realized events. The protocol is suitable for implementation on low-cost devices used in IoT. There are a Trusted Third Party and the User A and the User B in the protocol. The proposed protocol uses only an appropriate lightweight hash function and pre-shared secret data. The protocol ensures authenticity of communication entities, integrity of transmitted data, security of secret authentication keys and non-repudiation of realized events. The proposed protocol uses a TTP to ensure non-repudiation of realized events. According to the standard ISO/IEC 13888-2:2010 the TTP can be used to ensure non-repudiation using symmetric cryptography. This standard provides a description of generic structures that can be used for non-repudiation services and it also describes some specific communication-related mechanisms which can be used to provide non-repudiation of origin and non-repudiation of delivery. The ISO/IEC 13888-2:2010 relies on the existence of a TTP to prevent fraudulent repudiation or accusation. An online TTP is usually needed. Table 1 shows notations used in our protocol and their meaning. Figure 1 shows the principle of mutual authentication of the User A and the User B using the TTP with non-repudiation of realized events.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_x</td>
<td>The unique identifier of an entity x. The size of ID_x = 128 b.</td>
</tr>
<tr>
<td>$\parallel$</td>
<td>The bitwise concatenation operation.</td>
</tr>
<tr>
<td>001 – 111</td>
<td>The value in bits which defines the composition of following data. The size of 000 – 111 = 3 b.</td>
</tr>
<tr>
<td>$h_1 \rightarrow h_{10}$</td>
<td>A one-way cryptographic hash function with the digest size = 160 b.</td>
</tr>
<tr>
<td>$K_{AB}$</td>
<td>The secret authentication key shared between the User A and the User B. The size of $K_{AB}$ = 160 b.</td>
</tr>
<tr>
<td>$K_{ATTP}$</td>
<td>The secret authentication key shared between the User A and the TTP. The size of $K_{ATTP}$ = 160 b.</td>
</tr>
<tr>
<td>$K_{BTTP}$</td>
<td>The secret authentication key shared between the User B and the TTP. The size of $K_{BTTP}$ = 160 b.</td>
</tr>
<tr>
<td>$S_{\text{AB}}$</td>
<td>The public sequence number shared between the User A and the User B. The size of $S_{\text{AB}}$ = 16 b.</td>
</tr>
<tr>
<td>$S_{\text{ATTP}}$</td>
<td>The public sequence number shared between the User A and the TTP. The size of $S_{\text{ATTP}}$ = 16 b.</td>
</tr>
<tr>
<td>$S_{\text{BTTP}}$</td>
<td>The public sequence number shared between the User B and the TTP. The size of $S_{\text{BTTP}}$ = 16 b.</td>
</tr>
<tr>
<td>$x$ = 1</td>
<td>It represents increasing the sequence number x by one.</td>
</tr>
</tbody>
</table>

Tab. 1. Notations used in our protocol.
The proposed protocol consists of the two phases, an initialization phase and an authentication phase. The initialization phase must be performed before the authentication phase. Entities involved in the protocol exchange pre-shared data in the initialization phase. A communication between communication sides is ensured by a wireless communication in the initialization phase. The communicating entities approach so that transmitted data cannot be captured by an attacker in the initialization phase. The User A and the User B exchange their unique public identifiers IDA, IDB and a secret authentication key KAB. The User A and the TTP exchange their unique public identifiers IDA, IDTTP and a secret authentication key KATTP. The User B and the TTP exchange their unique public identifiers IDB, IDTTP and a secret authentication key KBTTP. The User A must keep secret the authentication keys KAB and KATTP. The User B must keep secret the authentication keys KAB and KBTTP. The TTP must keep secret the authentication keys KATTP and KBTTP. The unique identifiers IDA, IDB and IDTTP are public values. The principle of mutual authentication with non-repudiation of realized events is depicted in Fig. 1. In this case, the communication is started by the User A. If the communication starts by the User B, the principle will be mirrored opposite. Seven messages, which are exchanged between the TTP, the User A and the User B, are used in the authentication phase.

In the messages 1 – 7, the first ID defines the recipient of the messages 1 – 7, the second ID defines the sender of the messages 1 – 7 and the third ID defines the recipient of authentication data (in the messages 1, 3 and 5) and the sender of authentication data (in the messages 2, 4 and 6). Authentication data represent hashes (which include the secret authentication key KAB) and their corresponding sequence numbers. The hashes h2, h3, h5, h6, h8, h9 and h10 ensure integrity and authenticity of the messages 1 – 7. All transmitted data with the secret authentication key (KATTP or KBTTP) are inserted to the input of the hash function. Output hashes are inserted after transmitted data in the messages 1 – 7. Authentication between the User A and the TTP is ensured using the secret authentication key KATTP and the sequence number SNAB in the hashes h2, h3, h5, h6, h8, h9 and h10. Authentication between the User B and the TTP is ensured using the secret authentication key KBTTP and the sequence number SNBTTP in the hashes h3, h5, h6, h8 and h10. Authentication between the User A and B is ensured using the secret authentication key KAB and the sequence number SNAB in the hashes h1, h4 and h7.

The principle of authentication between communication entities is as follows. An entity creates an authentication hash using the secret authentication key and the sequence number (in the hashes h1, h4 and h7) and using other data (in the hashes h2, h3, h5, h6, h8, h9 and h10). After that, the entity sends the created authentication hash with sequence number and other data to an opposite entity. The opposite entity compares the received authentication hash with its own authentication hash which the entity computed by the hash function using the received sequence number (and other data) and its own secret authentication key which shares with the communication entity. If they are equal, integrity and au-

<table>
<thead>
<tr>
<th>User A (A)</th>
<th>User B (B)</th>
<th>Trusted Third Party (TTP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(IDA, IDB, IDTTP, KAB, KATTP)</td>
<td>(IDA, IDB, IDTTP, KAB, KBTTP)</td>
<td>(IDA, IDB, IDTTP, KATTP, KBTTP)</td>
</tr>
</tbody>
</table>

Fig. 1. Principle of authentication of User A and User B using TTP with non-repudiation of realized events.
The corresponding sequence numbers through the TTP. The User A saves the hash for the case of dispute, when the User A will argue that he did not authenticate the User B through the TTP. The User A claims that he did not authenticate the User B and he did not send the request for authentication from the User A through the TTP while the User A claims that he did not authenticate the User B through the TTP. The User B saves the hash for the case of dispute, when the User A will argue that he did not authenticate the User A and he did not send the request to authentication to the User A through the TTP. The User A saves the hash for the case of dispute, when the User B will argue that he did not authenticate the User A and he did not send the request to authentication to the User B through the TTP. The User A saves the hash for the case of dispute, when the User B will argue that he did not authenticate the User A and he did not send the request to authentication to the User B through the TTP.

In the case when a communication is started by the User A, the hash $h_1$ represents the requirement for authentication of the User A to the User B and the hash $h_2$ represents the proof of sending the hash $h_1$. The hash $h_4$ represents the requirement for authentication of the User B to the User A (also represents the confirmation of realized authentication of the User A to the User B) and the hash $h_5$ represents the proof of sending the hash $h_4$. The hash $h_7$ represents the confirmation of realized authentication of the User B to the User A and the hash $h_6$ represents the proof of sending the hash $h_7$. The User B saves the hashes $h_1$ and $h_2$ and the corresponding sequence numbers $Sn_{AB}$ and $Sn_{ATTP}$ to his database for the case of dispute, when the User A will argue that he did not send the request to authentication to the User B through the TTP. The User A saves the hashes $h_4$ and $h_5$ and the corresponding sequence numbers $Sn_{AB}$ and $Sn_{BTTP}$ to his database for the case of dispute, when the User B will argue that he did not authenticate the User A and he did not send the request to authentication to the User A through the TTP.

In the case when a communication is started by the User A, three following disputes may occur in our protocol:

1.) The User B claims that he received the request for authentication from the User A through the TTP while the User A claims that he did not send the request for authentication to the User B through the TTP.

2.) The User A claims that he was authenticated by the User B and that he received the request for authentication from the User B through the TTP while the User B claims that he did not authenticate the User A and he did not send the request for authentication to the User A through the TTP.

3.) The User B claims that he was authenticated by the User A while the User A claims that he did not authenticate the User B.

The TTP solves these disputes in the following way. The User B will send hashes $h_1$, $h_2$, $h_7$, $h_9$ and the corresponding sequence numbers $Sn_{AB}$, $Sn_{ATTP}$ to the TTP for the solution of the dispute 1 and 3. The User A will send hashes $h_4$, $h_5$ and the corresponding sequence numbers $Sn_{AB}$ and $Sn_{BTTP}$ to the TTP for the solution of the dispute 2. The TTP creates hashes $h_4'$, $h_5'$, $h_6'$ using constants, the received values from the User B and the User A and using its own secret key $K_{ATTP}$ and $K_{BTTP}$. The TTP compares the received hashes with computed hashes. If they are equal, the TTP will agree with the User B in the dispute 1 and 3 and with the User A in the dispute 2.

4. Security Analysis of Authentication Protocol

Security of the initialization phase is based on the secure channel. Security of the authentication phase with non-repudiation of realized events is based on cryptographic properties of hash functions, secret authentication keys, sequence numbers and the trust of the User A and the User B to the TTP. If the authentication key is revealed by an attacker, the authentication key will be invalidated for future communications. In the authentication phase the following cryptographic properties are ensured:

**Authenticity** – Authentication of the User A and the User B is ensured by a hash function and the secret shared key $K_{AB}$, which is transmitted in the hashes $h_1$, $h_3$ and $h_7$. Authentication of the User A and the TTP is ensured by a hash function and the secret shared key $K_{ATTP}$, which is transmitted in the hashes $h_2$, $h_6$ and $h_8$. Authentication of the User B and the TTP is ensured by a hash function and the secret shared key $K_{BTTP}$, which is transmitted in the hashes $h_3$, $h_5$, $h_9$ and $h_{10}$.

**Unrepeatability** – Unrepeatability of transmitted data is ensured by the public sequence numbers $Sn_{AB}$, $Sn_{ATTP}$ and $Sn_{BTTP}$. The sequence number $Sn_{AB}$ is transmitted in the hashes $h_1$, $h_4$ and $h_7$. The sequence number $Sn_{ATTP}$ is transmitted in the hashes $h_2$, $h_6$ and $h_8$. The sequence number $Sn_{BTTP}$ is transmitted in the hashes $h_3$, $h_5$, $h_9$ and $h_{10}$. Entities involved in the protocol do not respond to messages containing the sequence number which is less or equal to the last correctly used sequence number between communication parties.

**Integrity** – Integrity of transmitted data is ensured by a hash function in the steps 1 – 7. All transmitted data are inserted on the input of the hash function in the each step. The output hashes $h_2$, $h_3$, $h_5$, $h_8$, $h_9$ and $h_{10}$ are attached after the transmitted data in the steps 1 – 7. The receiver computes a hash with using received data and his secret authentication key and compares it with the hash which received together with transmitted data. If they are equal, integrity and authenticity of transmitted data will be ensured.

**Security** – Security of the secret authentication keys $K_{AB}$, $K_{ATTP}$ and $K_{BTTP}$ is ensured by a hash function. The
hash function is a one-way compression function \( \Rightarrow \) it is not possible to get the input values from the output of the hash function.

**Uniformity** – Uniformity of hashes \( h_2, h_3, h_5, h_6, h_9, h_{10} \) is ensured by the sequence numbers \( S_{nAB}, S_{nATTP} \) and \( S_{nBTTP} \) and a hash function. A change of one bit on the input of the hash function causes an unpredictable random change of all output bits with 50% probability. This feature of hash functions ensures that very similar messages have different output hashes.

**Non-repudiation** – Non-repudiation of sending the request for authentication and realized authentication is ensured by the TTP. When the communication is started by the User A, the hash \( h_2 \) represents the proof of submission of the request for authentication to the User B from the User A through the TTP. The hash \( h_5 \) represents the proof of realized authentication of the User A to the User B and the proof of submission the request for authentication to the User A from the User B through the TTP. The hash \( h_9 \) represents the proof of realized authentication of the User B to the User A through the TTP.

Authentication protocols must work correctly and safely. Formal methods for an analysis of security cryptographic protocols can be used for this purpose. These methods are able to find security threats in cryptographic protocols. The most widely used method for the formal analysis of authentication protocols is the BAN (Burrows, Abadi, Needham) logic [23]. We provided formal analysis of the authentication phase in our protocol by the BAN logic. To analyse the authentication phase, we give the following assumptions by the BAN logic:

- **A** believes \( K_{AB} \leftrightarrow B \), **B** believes \( K_{AB} \leftrightarrow B \).
- TTP believes \( K_{ATTP} \leftrightarrow \) TTP, **A** believes \( K_{ATTP} \leftrightarrow \) TTP.
- TTP believes \( K_{ATTP} \leftrightarrow \) TTP, **B** believes \( K_{ATTP} \leftrightarrow \) TTP.
- **A** believes fresh \( (S_{nAB}, S_{nATTP}) \),
- **B** believes fresh \( (S_{nAB}, S_{nBTTP}) \).

and TTP believes fresh \( (S_{nATTP}, S_{nBTTP}) \).

We analyse the idealized version of our protocol by applying rules of the BAN logic to the assumptions. We give many of the formal details necessary for the proof only for the message 1 for brevity.

Based on the BAN logic, we idealize the message 1 as:

\[
\text{A} \rightarrow \text{TTP}: \ < S_{nAB}, A \leftrightarrow B > K_{AB}, \ < S_{nAB}, A \leftrightarrow B > K_{AB}, \ S_{nATTP}, A \leftrightarrow \text{TTP} > K_{ATTP}.
\]

The main steps of the proof are as follows:

The TTP receives the message 1. The annotation rules yield that:

- TTP sees \( S_{nAB}, A \leftrightarrow B > K_{AB} \)
- \( < S_{nAB}, A \leftrightarrow B > K_{AB}, S_{nATTP}, A \leftrightarrow \text{TTP} > K_{ATTP} \)

holds afterward. Since we have the hypothesis

- TTP believes \( K_{ATTP} \leftrightarrow \text{TTP} \).

The message-meaning rule for shared secrets applies and yields the following:

- TTP believes **A** said \( (< S_{nAB}, A \leftrightarrow B > K_{AB}, S_{nATTP}, A \leftrightarrow \text{TTP} > K_{ATTP}) \).

We break conjunctions and then we produce

- TTP believes **A** said \( (S_{nATTP}, A \leftrightarrow \text{TTP}) \).

Moreover, we have the following hypothesis:

- TTP believes fresh \( (S_{nATTP}) \).

The nonce-verification rule applies and yields

- TTP believes **A** believes \( (S_{nATTP}, A \leftrightarrow \text{TTP}) \).

Again, we break the conjunction to obtain the following:

- TTP believes **A** believes \( K_{ATTP} \leftrightarrow \text{TTP} \).

This concludes the analysis of the message 1 of the authentication phase in the proposed protocol.

The proposed protocol should be resistant to the attacks which are shown in Tab. 2. Table 2 also shows the ways which are used to a protection against the mentioned attacks. The protection against the replay attack is ensured by the sequence numbers \( S_{nAB}, S_{nATTP} \) and \( S_{nBTTP} \). The Man in the Middle (MIM) attack is not possible because the secret authentication keys \( K_{AB}, K_{ATTP} \) and \( K_{BTTP} \) are exchanged by the secure channel in the initialization phase. The eavesdropping attack is not possible because the secret authentication keys \( K_{AB}, K_{ATTP} \) and \( K_{BTTP} \) are protected by a hash function. It is not possible to get the input values from the output hash of the

<table>
<thead>
<tr>
<th>Attack</th>
<th>Method of protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replay attack</td>
<td>The sequence numbers ( S_{nAB}, S_{nATTP} ) and ( S_{nBTTP} ).</td>
</tr>
<tr>
<td>Man in the middle (MIM) attack</td>
<td>The secret authentication keys ( K_{AB}, K_{ATTP} ) and ( K_{BTTP} ) are exchanged in the initialization phase by a secure channel.</td>
</tr>
<tr>
<td>Eavesdropping attack</td>
<td>The secret authentication keys ( K_{AB}, K_{ATTP} ) and ( K_{BTTP} ) are protected by a one-way hash function.</td>
</tr>
<tr>
<td>Desynchronization attack</td>
<td>The sequence numbers ( S_{nAB}, S_{nATTP} ) and ( S_{nBTTP} ) are transmitted as the public values.</td>
</tr>
<tr>
<td>Attack using Shor’s algorithm [22]</td>
<td>Protocol does not use the IF and DL problem, elliptic curves, hyperelliptic curves, class groups, etc.</td>
</tr>
</tbody>
</table>

**Tab. 2.** Ineffective attacks to our proposed protocol and methods of protections against these attacks.
hash function. The desynchronization attack is not possible since the sequence numbers $S_{AB}, S_{ATTP}$ and $S_{BTTP}$ are transmitted as public values. From this reason, communication parties always know the value of the sequence number used in the authentication hash ($h_1 – h_{10}$). The attack using the Shor’s algorithm [22] is not possible since our protocol does not use the integer factorization problem, the problem of the discrete logarithm, elliptic curves, hyperelliptic curves, class groups and etc.

In our proposed protocol a hash function with the digest size equal 160 bits is intended. The hash function with the digest size equal 160 bits has effective security equal to 80 bits because the birthday paradox decreases security of hash functions to half. The authentication keys must be updated after using up the range of the sequence numbers. For the sequence number equal to 16 bits, the authentication key may be used $2^{16}$ (65 536) times. Authentication keys may be saved in a Secure Element which represents a tamper resistant hardware platform, capable of storing confidential and cryptographic data.

5. Demands of Proposed Protocol

Our authentication protocol has low demands to computing resources and transmitted data. The entities involved in the protocol compute only a light-weight hash function. Table 3 shows demands to computing resources and transmitted data of our protocol in the authentication phase for the User A, the TTP and the User B.

The User A performs the calculation of a hash function 6 times during the execution of the protocol. The User B and the TTP performs the calculation of the hash function 7 times during the execution of the protocol. Transmitted data by the User A are equal to 1350 b in the authentication phase. Transmitted data by the User B are equal to 1206 b in the authentication phase. Transmitted data by the TTP are equal to 2457 b in the authentication phase. Total transmitted data by the TTP, the User A and B are equal to 5013 b in the authentication phase.

The authors of [24] implemented different light-weight hash functions on an ATMEL AVR ATtiny45 8-bit microcontroller and provided their performance evaluation. Table 4 shows memory requirements and a performance of light-weight hash functions PHOTON-160/36/36, SPONGENT-160/160/80 and Keccak $[r=40, c=160]$. These hash functions have the digest size equal to 160 b. In Tab. 4 there are the code size in bytes, the size of needed memory for RAM state and others in bytes and the cycle count for 100-byte message of selected light-weight hash functions.

Table 5 shows demands to hardware area in GE (Gate Equivalent) for selected light-weight hash functions. Hash functions PHOTON-160/36/36, SPONGENT-160/160/80 and Keccak $[r=40, c=160]$ are suitable for implementation in our protocol. From Tab. 4 and 5 it follows that Keccak $[r=40, c=160]$ is the most suitable for implementation in our protocol.

6. Conclusion

In this paper a new light-weight mutual authentication protocol with non-repudiation of realized events was presented. Our protocol is simple and uses only a light-weight hash function and incrementation of public sequence numbers. The advantages of the proposed protocol are its simplicity, low computing and memory demands, ensuring integrity of transmitted data, non-repudiation of realized events by
symmetric cryptography and resistance against attacks coming from universal quantum computers (using the Shor’s algorithm) in comparison with other light-weight authentication protocols. In the our feature work, we will implement the proposed protocol on RFID. Smart cards and wireless sensors and we will measure its performance, memory requirements and resistance against side channel attacks.

Acknowledgments

Research described in this paper was financed by the National Sustainability Program under grant LO1401. For the research, infrastructure of the SIX Center was used.

References


About the Authors . . .

Vlastimil ČLUPEK was born in Boskovice, Czech Republic. He received his Ph.D. from the Brno University of Technology, Faculty of Electrical Engineering and Communications, Department of Telecommunications in Brno in 2017. His research interests include cryptography security of ICT, authentication on low-cost devices and secure long-term archiving of electronic documents.

Václav ZEMAN was born in Plzeň, Czech Republic. He attend Brno University of Technology, Faculty of Electrical Engineering and Communications, Department of Telecommunications. Since 1993, he has been with Brno University of Technology. His research interests include cryptography security of ICT, the issue of data transfers, coding, computer modeling and simulation of electronic circuits.

Petr DZURENDA was born in Vyškov, Czech Republic. He received his Ing. from the Brno University of Technology, Faculty of Electrical Engineering and Communications, Department of Telecommunications in Brno in 2013. His research interests include cryptography security in access control systems, anonymous authentication and DDoS attacks. Since 2013, he has been with Brno University of Technology, where he works towards his Ph.D. thesis.