SECURE DATA TRANSMISSION IN WIRELESS SENSOR NETWORKS OF THE INTERNET OF THINGS

Olexander Belej[1], Volodymyr Chaplyha[2]

[1] associate professor of CAD department NULP, [2] lecturer of UBS LI

Abstract

Security studies of the Internet of things are increasingly attracting the attention of academia. This study addresses three key security requirements with an emphasis on the IoT system: authentication, privacy, and access control. It addresses security issues that have not yet been resolved, and discusses problems and future trends in the field of the Internet of things. Given the features of the devices, as well as their various nature, security issues of network interaction require consideration of new aspects. Most of the security threats identified were related to unencrypted data, personal data collection, vulnerable user interfaces and insecure connections. The main security problems are related to the fact that the existing methods and means of protection were originally developed for desktop computers and did not take into account the features and limitations of the Internet of things. Today, along with the adaptation of existing security technologies, standardization issues in the field of the Internet of things are important.

Introduction

The IoT concept includes many different technologies, services, standards and is perceived as the cornerstone in the market of information and communication technologies (ICT) for at least the next ten years. From a logical point of view, the IoT system can be represented as a set of jointly interacting smart devices. From a technical point of view, IoT can use various ways of data processing, communication, technology and methodology, based on their purpose. For example, an IoT system can use the capabilities of a wireless sensor network (WSN), which collects environmentally relevant environmental information (Belej, O.; Lohutova, T.; Banaś, M.;, 2019). A high level of heterogeneity, combined with a wide range of IoT systems, is expected to increase the number of threats to the security of device owners, which are increasingly used for the interaction of people, cars and things in any variation. Traditional security and confidentiality measures cannot be applied to IoT technologies, in particular because of their limited computing power. In addition, the large number of target connected devices poses a scalability problem. To prevent unauthorized users from accessing the system, authentication and authorization mechanisms should be used, and security, confidentiality and integrity of personal data should be guaranteed. Relative to the personal data of users and information, protection and confidentiality should be ensured, primarily because devices have access to it and are able to manage it. Finally, trust is the main problem, because the IoT environment is characterized by various types of devices that must process data in accordance with the needs and rights of users (Belej, 2019).

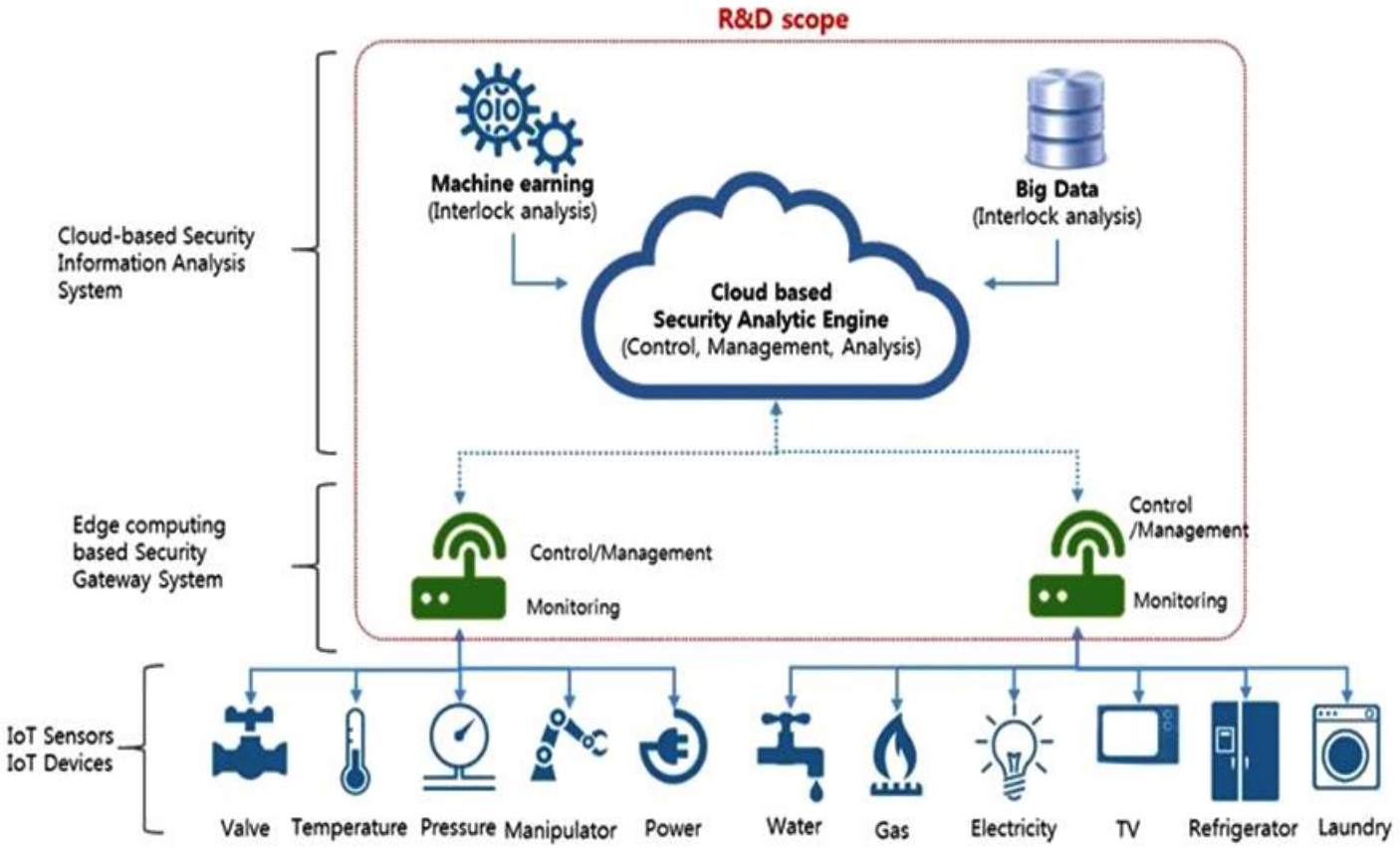

Figure 1. Scheme of step-by-step implementation of the DEU association stage

Also, during the operation of the protocol, the MC key is entered, which is associated with the Alice user. This key allows you to provide symmetric encryption of the data transmitted and stored on the DED, and also serves as an additional protection against leakage of confidential Alice user information stored on a separate DED. Further protocol steps are indicated by the corresponding numbers in fig. 2.

1. The Alice user generates the MC secret key for the new DED w_i and transmits it over the secure channel.
2. DED w_i transfers the result of the execution of the hash function from its own software to the MC using a secure channel *(hash (SW_i)).*
3. User Alice transfers his public key PK_A and ID_A to the US.
4. MC generates US certificate for Alice. The certificate has the form $cert_A = sign_{cloud}(w_i, ID_A, hash(SW_i))$. This step is carried out in order to ensure system data integrity.
5. The MC, using a secure channel, transfers the $cert_{cloud}$ to the Alice user.
5. The Alice user signs the certificate received from the MS with his $cert_A = sign_A(cert_{cloud})$.
7. The Alice user submits a certificate to the DED.

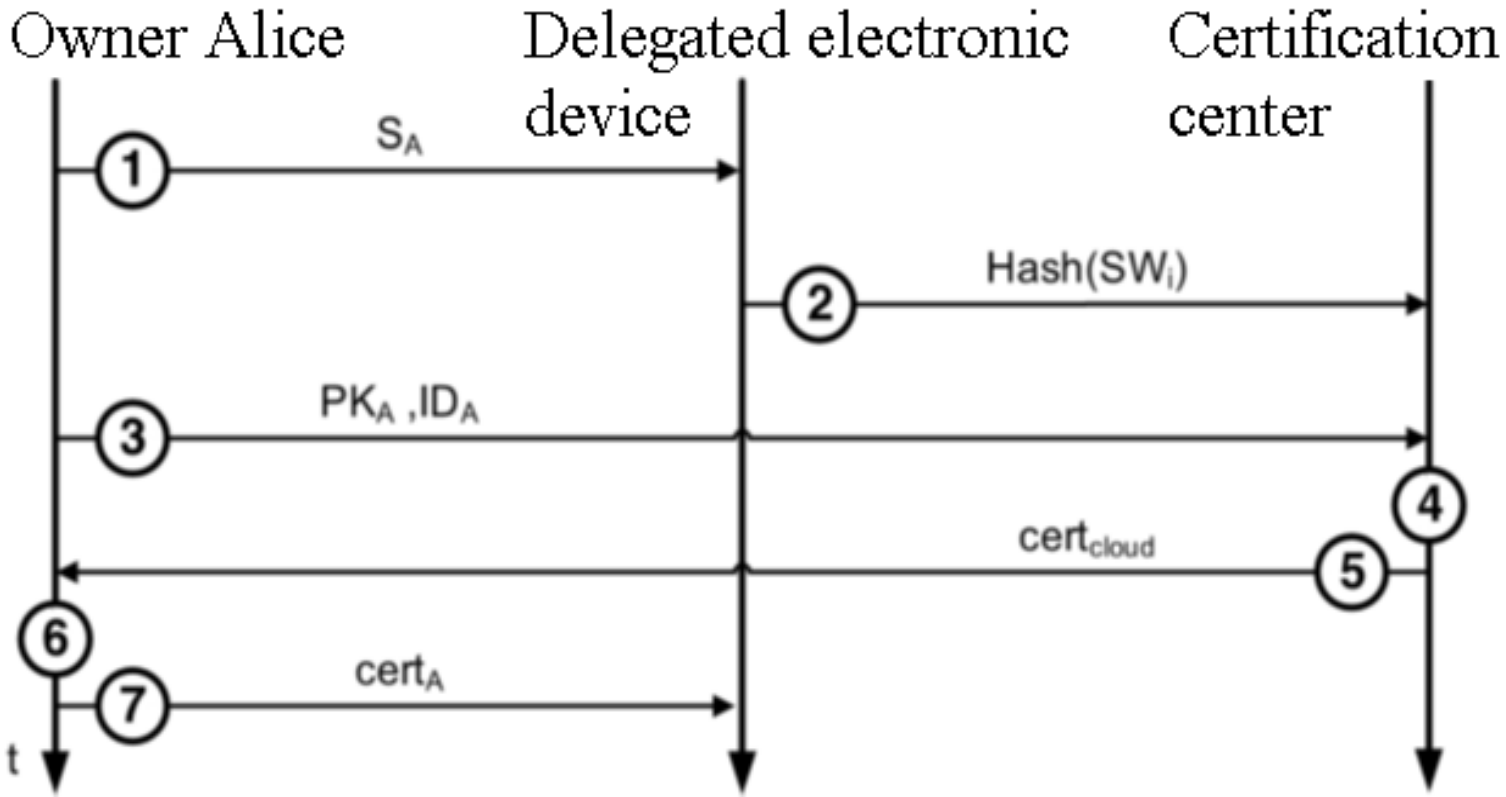

Figure 2. Scheme of step-by-step implementation of the DED association stage.

Protocol 2A - the presence of a stable connection for both users (Fig. 3).

1. User Alice sets the maximum delegation time (t_d) for DED wi using a service message signed by $m[D]_A = sign_A(w_i, t_d, ID_A, ID_A, \{\text{other delegation conditions}\})$, according to step 1.

2. User Alice transfers $m[D]_A$ to the MC using a secure channel, according to step 2.

3. The MC verifies the authenticity of the message from Alice using PK_A. If the test fails, the protocol stops working according to step 3.

4. The MC signs the delegation message $m[D]_{cloud} = sign_{cloud}(m[D]_A)$ according to step 4.

5. The MC transmits $m[D]_{cloud}$ and $cert_A$ for Bob, in accordance with step 5.

6. User Alice sends the service message $(m[C(S_A)]_A)$ to the DED, deleting the private key MC from the DED, in accordance with step 6.

7. If the user Bob does not trust Alice, the protocol performs step 7. The DEU is reset to the factory settings. The reset occurs with the $m[D]_{cloud}$ and the $cert_{cloud}$ certificate stored in the secure storage. These values were obtained in step 6 of the initialization protocol in order to preserve data integrity and confirm ownership rights in the absence of connection with the MC. User Bob compares the result of the hash function from the current DED software with that stored in $cert_{cloud}$. If they do not match, the algorithm stops execution. Thus, the Bob user loses the ability to use the DED, since it is assumed that the software could be skommeted by the owner. It is important to note that the protected timer and storage remain unchanged even when reset to factory settings.

8. If user Bob trusts Alice, the software component of the DED remains unchanged, and the temporary user has the opportunity to use the software of the device owner.

9. User Bob generates an S_B secret key for direct interaction between the DED, according to protocol step 12.

10. User Bob sends the S_B to the DED via a secure channel, as described in step 9.

11. To ensure data integrity, the Bob user calculates a new value $sign_B(w_i, SW_i)$.

12. In case of expiration of the td delegation timer on the DED side, the device parameters are reset to the factory settings while maintaining the contents of the protected storage. The timer can be remotely updated if there is a simultaneous connection to the owner's MC using the service message $m[D]_A = sign_A(w_i, t_d, ID_A, ID_A, \{\text{other delegation conditions}\})$.

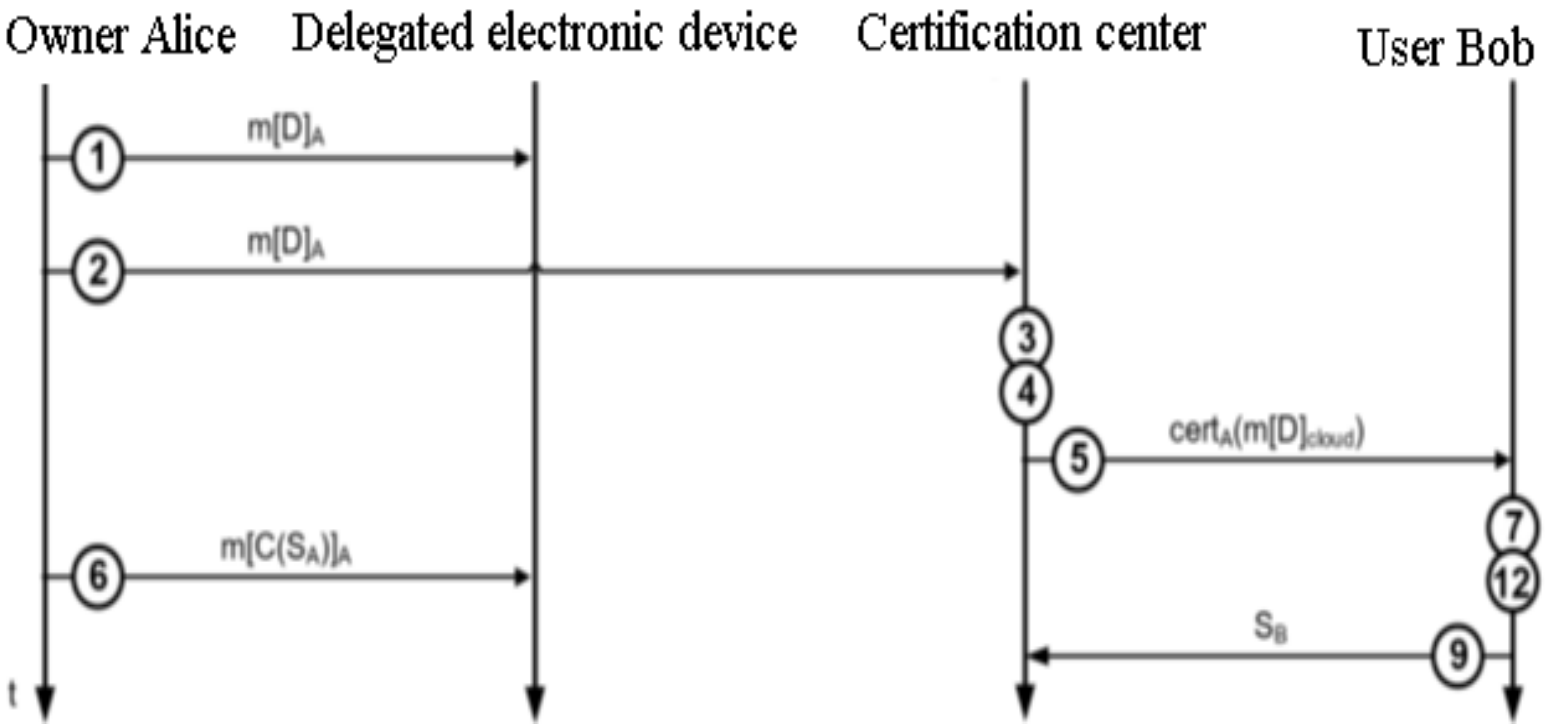

Figure 3. Scheme of step-by-step implementation of the delegation of DED in the presence of a stable connection with the SA.

1. User Alice sets the maximum delegation time t_d to DED w_i using a service message signed by $m[D]_A = sign_A(w_i, t_d, ID_A, ID_B, \{\text{other delegation conditions}\})$, according to step 1.

2. The Alice user passes the certA to the Bob user over the secure channel, according to step 2.

3. Bob authenticates $cert_A$ using $cert_{cloud}$. If the test fails, the protocol stops working, according to step 3.

4. User Alice sends the service message $(m[C(S_A)]_A)$ to the DED, deleting the private key S_A from the DED, according to step 4.

5. If the user Bob does not trust Alice, the protocol is run in accordance with step 5. The DED is reset to the factory settings.

The reset occurs with the $(m[D)]_{cloud})$ and the certcloud certificate saved in the secure store. These values were obtained at step 6 of the initialization protocol in order to preserve data integrity and confirm ownership in the absence of connection to the S_A. User Bob compares the result of the hash function from the current DED software with that stored in certcloud. If they do not match, the algorithm stops execution. Thus, the Bob user loses the ability to use the DED, since it is assumed that the software could be compromised by the owner. It is important to note that the protected timer and storage remain unchanged even when reset to factory settings.

6. If user Bob trusts Alice, the DED software component remains unchanged, and the temporary user has the opportunity to use the device owner's software.

User Bob generates an S_B secret key for direct interaction with the DED, according to step 10.

8. The user Bob transfers the S_B to the DED via a secure channel, according to step 11.

9. To ensure data integrity, Bob calculates the new value of $sign_B(w_i, SW_i)$. In the case of the expiration of the t_d delegation time on the DED side, the device parameters are reset to the factory settings while maintaining the contents of the protected storage. The timer can be updated using the service message $m[D]_A = sign_A(w_i, t_d, ID_A, ID_B, \{\text{other delegation conditions}\})$ if there is a direct connection between users.

11. The user Bob sends the S_B to the DED via a secure channel, according to step 9 in Fig. 4.

12. To ensure data integrity, the Bob user calculates a new value $sign_B(w_i, SW_i)$.

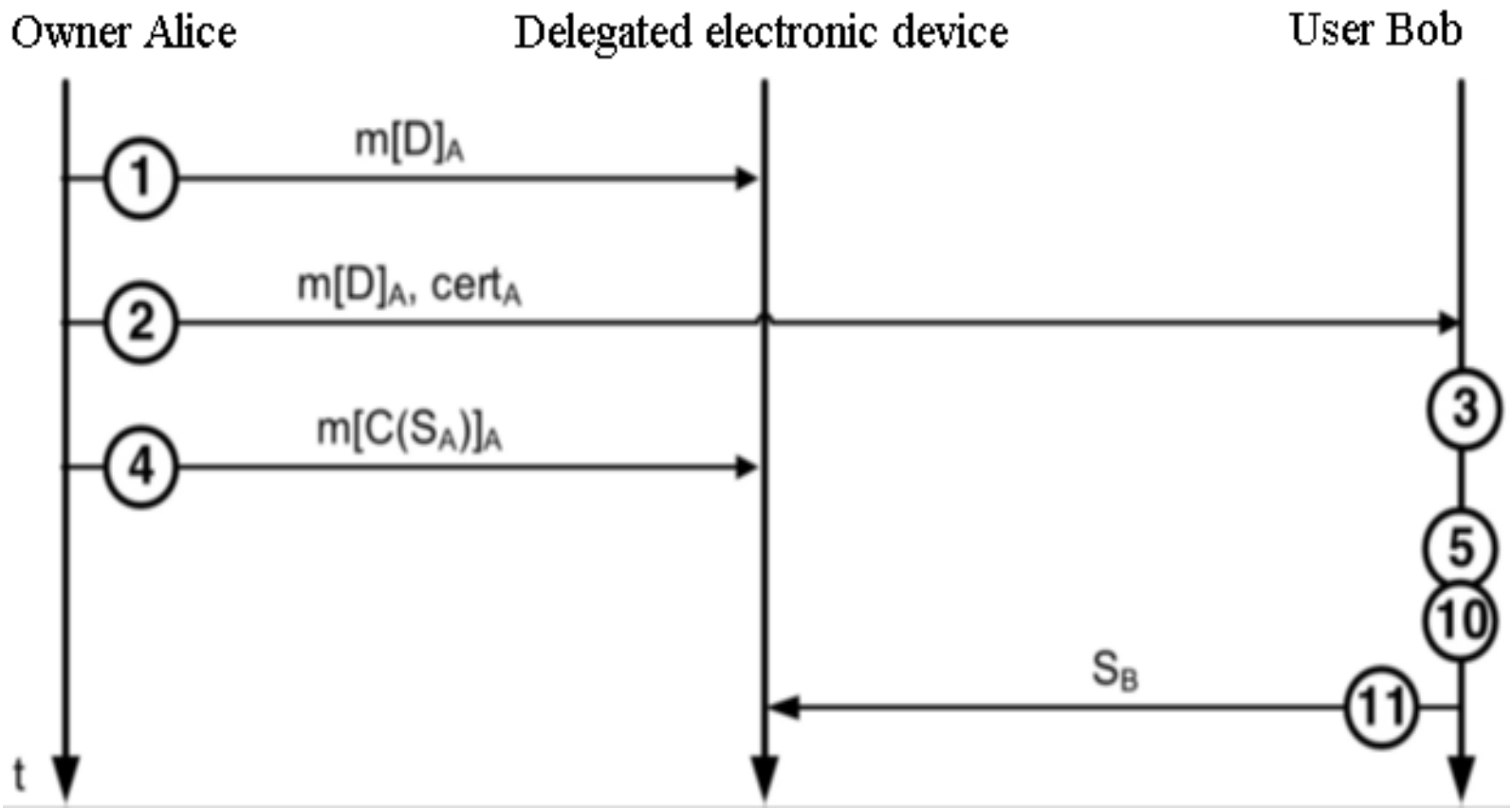

Figure 4. The scheme of step-by-step implementation of the DED in the absence of a stable connection with the CA.

Protocol 3A - the presence of a stable connection for both users (Fig. 5)

1. User Bob generates a service message signed by SK_B as $m[R]_B = sign_A(w_i, R)$, according to step 1.
User Bob transmits $m[R]_B$ to the MC using the secure channel, according to step 2.
3. The MC verifies the authenticity of the message from Bob using SK_B. If the test fails, the protocol stops working, according to step 3.
4. The MC signs the return message $m[R]_{cloud} = sign_{cloud}(m[R]_B)$, according to step 4.
5. The MC transmits the $m[R]_{cloud}$ to Alice, according to step 5.
6. User Bob sends the service message $m[C(S_B)]_B$ to the DED, deleting the S_B secret key from the DED, according to step 6.
7. If the Alice user does not trust Bob, step 7. The DED is reset to the factory settings and the data is stored in a secure storage. Alice's user account compares the result of the hash function from the current DED software with that stored in certcloud. If they do not match, the protocol stops working. Thus, the Alice user loses the ability to use the DED, since it is assumed that the software could be compromised by the owner. It is important to note that the protected timer and storage remain unchanged even when reset to factory settings.
8. If the Alice user trusts Bob, the DED software component remains unchanged, and the device owner has the opportunity to use the software installed during the delegation process.
9. The Alice user generates an S_A secret key for direct interaction with the DED, according to step 8.
10. User Alice transfers the S_A to the DED using a secure channel, according to step 9.
11. To ensure data integrity, the Alice user calculates a new value $sign_A(w_i, SW_i)$. Only the $cert_A$ and certcloud are in the secure DED storage.

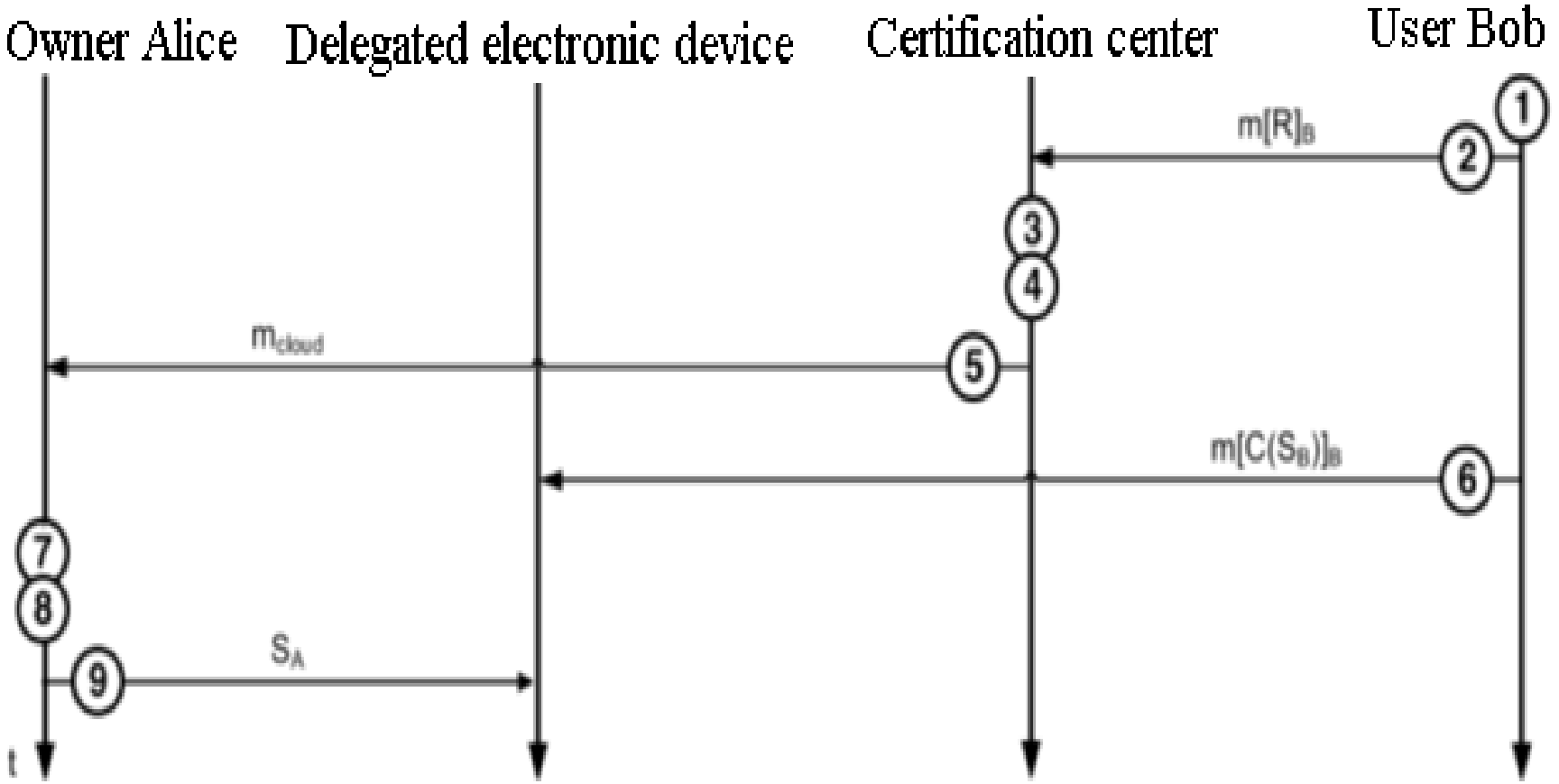

Figure 5. Scheme of step-by-step execution of the stage of returning DED in the presence of a stable connection with the SA.

Protocol 3B - the absence of a stable connection for at least one of the users (Fig. 6)

1. User Bob generates a service message signed by SK_B as $m[R]_B = sign_A(w_i, R)$ according to step 4.
2. User Bob transfers $m[R]_B$ to user Alice via a secure channel, as per step 2.
3. Alice authenticates $m[R]_B$ using certcloud. If the verification fails, the protocol stops its work, according to step 3.
4. User Bob sends the service message $m[C(S_B)]_B$ to the DED, deleting the S_B secret key from the DED, according to step 4.
5. If the Alice user does not trust Bob, the protocol operates according to step 5. The DED is reset to the factory settings with the data stored in the secure storage. Alice's user account compares the result of the hash function from the current software of DED with that stored in certcloud. If they do not match, the protocol stops working. Thus, the Alice user loses the ability to use the DED, since it is assumed that the software could be compromised by the owner. It is important to note that the protected timer and storage remain unchanged even when reset to factory settings.
6. If the Alice user trusts Bob, the DED software component remains unchanged, and the owner has the opportunity to use the device's temporary user software.
7. The Alice user generates the SA secret key for direct interaction with the DED, according to step 10.
8. User Alice transfers the SA to the DED through a secure channel, according to step 11.
9. To ensure data integrity, the Alice user calculates a new value $sign_A(w_i, SW_i)$. Only the $cert_A$ and $cert_{cloud}$ are in the secure DED storage.

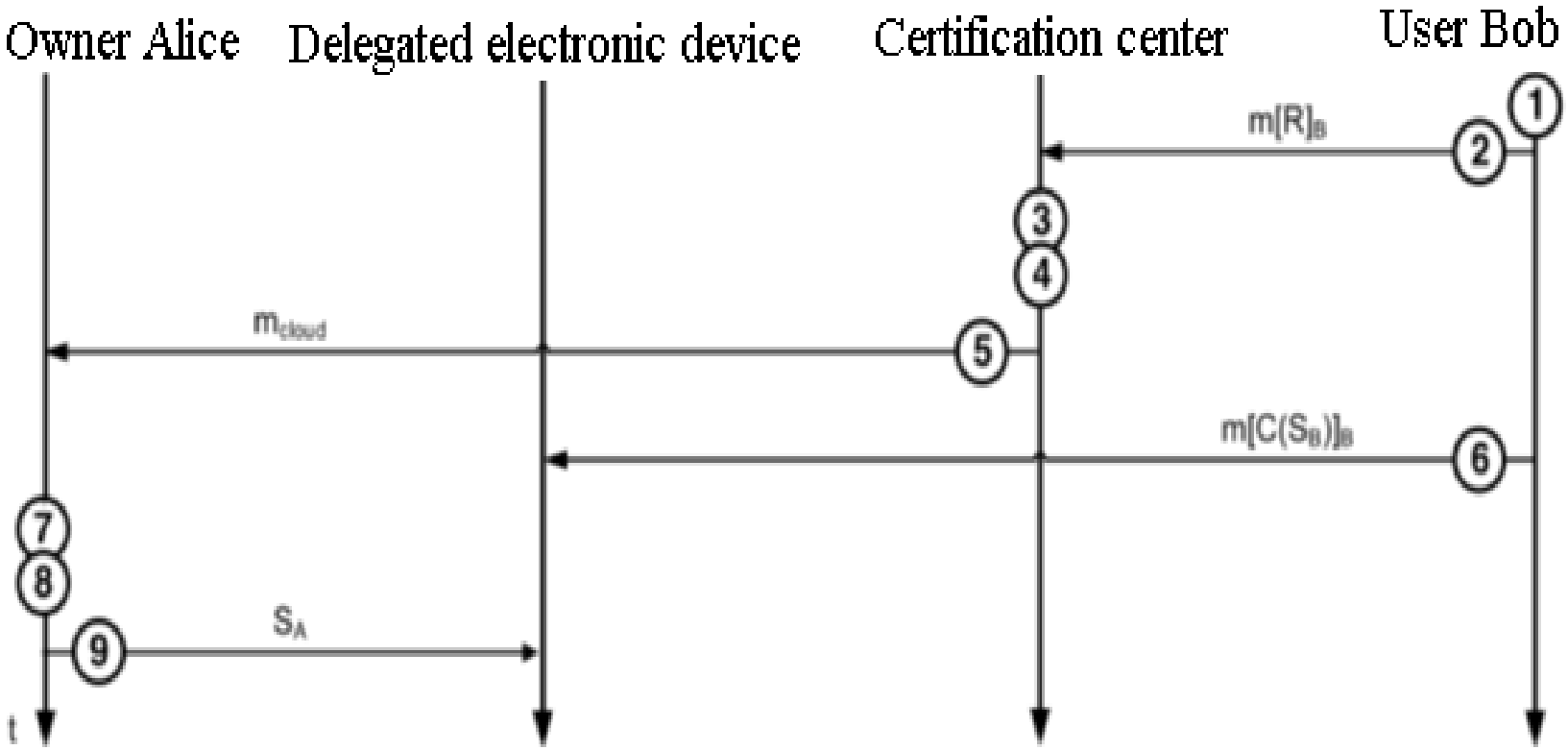

Figure 6. Scheme of step-by-step completion of the return of DED in the absence of a stable connection with the SA.

Conclusion

The authentication protocol for electronic devices monitoring is presented, developed for use in conditions of unstable communication with a certification center. Algorithms that implement the stages of functioning of the proposed protocol can be implemented both in the form of software for a universal computer of any architecture, and in the form of hardware for a specialized computer of any architecture. The protocol can be used in places with lack of infrastructure, because for the implementation of delegation does not require a constant connection to the S_A. The proliferation of IoT services requires security and privacy to be guaranteed. The review of publications published in the works clearly demonstrates how many unsolved problems remain, sheds light on the areas of research in the field of IoT security. Until now, a single concept has not been formulated regarding the requirements of security and confidentiality in such a heterogeneous environment using various communication technologies and standards. Appropriate solutions need to be developed and implemented. They should be platform independent and allow guaranteeing access control and confidentiality of users and things, reliability among devices and users, adherence to certain privacy security policies. Research is required on IoT security in mobile devices, which is becoming more widespread today. Much effort has been (and will be) made by the world scientific community to solve existing unsolved problems. At the same time, in the process of work, there will be many new questions that are yet to be faced. This article will be useful in choosing further areas of research and will contribute to the massive deployment of IoT systems in the real world.

Thank You!