Noname manuscript No.
(will be inserted by the editor)

Anonymous Lightweight Proxy Based Key Agreement for IoT (ALPKA)

An Braeken · Madhusanka Liyanage · Anca Delia Jurcut

Received: date / Accepted: date

Abstract The Internet of Things (IoT) technologies interconnect a broad range of network devices, differing in terms of size, weight, functionality, and resource capabilities. The main challenge is to establish the required security features in the most constrained devices, even if they are unknown to each other and do not share common pre-distributed key material. As a consequence, there is a high need for scalable and lightweight key establishment protocols.

In this paper, we propose a key agreement protocol between two IoT devices without prior trust relation, using solely symmetric key based operations, by relying on a server or proxy based approach. This proxy is responsible for the verification of the authentication and the key agreement between the IoT devices, without being capable of deriving the established session key. We propose two versions. The first version does not require interactive input from the key distribution center to the proxy, but is not resistant if a compromised user and proxy are collaborating. The second version on the other hand is collision resistant, but needs an interactive key distribution center. In addition, we add the interesting features of anonymity and unlinkability of the sender and receiver in both protocol versions. The security properties of the proposed protocol are verified by using formal verification techniques.

Keywords Internet of Things · Authentication · Key Establishment · Proxy · Resource-Constrained Devices · Anonymity · Formal Verification

An Braeken
Industrial Sciences Department, Vrije Universiteit Brussel, Belgium
Tel.: +32-468-104767
E-mail: abraeken@vub.be

Madhusanka Liyanage
School Of Computer Science, University College Dublin, Ireland
Centre for Wireless Communications, University of Oulu, Finland

Anca Delia Jurcut
School Of Computer Science, University College Dublin, Ireland.

1 Introduction

The Internet of Things (IoT) makes connections anywhere, with anything and anytime possible [1,2]. It allows the development of a new generation of services and applications. The security and privacy aspects from both end-users and devices are essential for the credibility of future IoT systems. This is not an evident task, as the measures should be implemented in a highly efficient way due to the power and processing constraints which characterises many IoT devices. As such, existing standard protocols for security protection cannot be directly applied in the very constrained devices.

In order to establish a security key between two devices that are not necessarily known to each other, there are three options. (1) One could pre-share every time a common secret key in each of the devices. However, this method is quite impractical and represents an unscalable approach. (2)The second option is to use asymmetric security techniques. It results in adding high computational costs for resource constrained devices. (3)As last option, one could rely on a secure third party, also called the server or the proxy. In this last option, it is possible to establish the communication by means of low cost cryptographic operations, in the assumption that each of the devices share a common secret key with the proxy.

In this paper, we propose such a proxy-based key agreement protocol, using solely symmetric key based operations. Consequently, our proposed protocol is specifically designed for highly resource-constrained devices. The proxy is considered honest, but curious. This means that it will perform all the required actions, but might be interested in collecting the data for its own purposes (e.g. to offer it later for sale). Therefore, the task of the proxy is mainly to ensure the communication between two authenticated entities in the system and to establish a key agreement such that the transmitted ciphertext can be decrypted by the intended receiver. This results in a securely shared session key between the two entities. Moreover, no information about the sender and receiver can be derived by an outsider from the transmitted messages. Since some of the IoT devices can be linked to a person and other IoT devices linked to a particular application, some user behaviour can be derived from eavesdropping the communication channel. As people get more and more aware of the importance of anonymity, it is essential to add this feature to the current systems [3].

Our protocol can be seen as a proxy re-encryption protocol, which is made anonymous using techniques from the area of multi-server authentication protocols. In a proxy re-encryption scheme, a ciphertext is converted in another ciphertext by the proxy, ensuring the involvement of the claimed entities. A technique to establish anonymity from the domain of multi-server authentication protocols is based on the asymmetric share of secrets between the entities and the servers, also called proxies here. Furthermore, we formally verify our proposed protocol by proving that the goals of the protocol are established (e.g. authentication, freshness, session-key establishment) and demonstrating the absence of design weaknesses that may be exploitable by mountable at-

tacks [4], [5], [6]. We now discuss two usecases, one in the healthcare domain and the other for the industrial internet .

Usecases

A patient that wants to make use of the different services offered by a hospital can be an example of a use-case. During registration, the patient receives its individual required security material on its smart phone. When the patient requests a service (e.g. measurements of a physical parameter), it sends its pseudo identity information to that service, which is used by the service to communicate the resulting output via the proxy. Also health related info stored at the smartphone of the patient can be securely send to another authenticated IoT device in the hospital environment via the proxy. As a consequence, both patient's smartphone and any IoT device authenticated by the hospital can communicate with each other, without being mutually authenticated beforehand. Another example, is when a patient of a hospital requires a special follow up, based on a heart beat monitor attached to the patient. Every time, the heart beat monitor reaches a certain threshold, the most nearby light should be activated and a nurse call containing the history of the latest measurements should be send out. Again, through the usage of our proposed ALPKA protocol, there is no pre-shared key agreement required between the heart beat monitor and the smart light, or nurse call system. Figure 1 illustrates the communication flow in this usecase

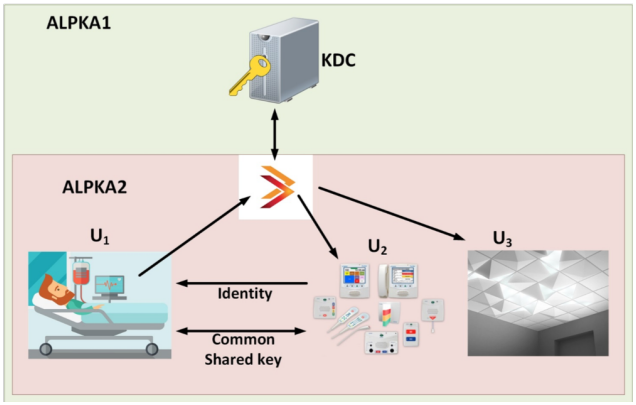

Fig. 1 Network system model for the healthcare usecase

A mobile robot in a factory is another example of a usecase. In todays factory environment, the communication is happening between entities via common networks through WiFi networks. However, security provided by WiFi network are not sufficient to satisfy the security requirements in industrial standards such as ISO/IEC 27000. Therefore, an extra level of security is required to ensure the proper operation in a factory. By using the ALPKA

protocol, the robot can retrieve the individual required security material during the registration. While moving across the factory, the robot requests a service form another device or robot. So, it can send its pseudo identity information to that service, which is used by the service to communicate the resulting output via the proxy. Also, control or monitoring info stored at the robot can be securely send to another authenticated IoT device in the factory via the proxy. Consequently, the robot can communicate to any authenticated IoT device in the factory, without being mutually authenticated beforehand. This is ideal for a factory environment since a robot has to communicate with a large number of IoT devices and it will reduce the additional authentication delay. It will drastically improve the efficiency of the production.

Figure 2 illustrates how a single robot can securely communicate with multiple sensors by using the ALPKA protocol. Some proposals can be replicated, when multiple robots wants to communicate with multiple sensors.

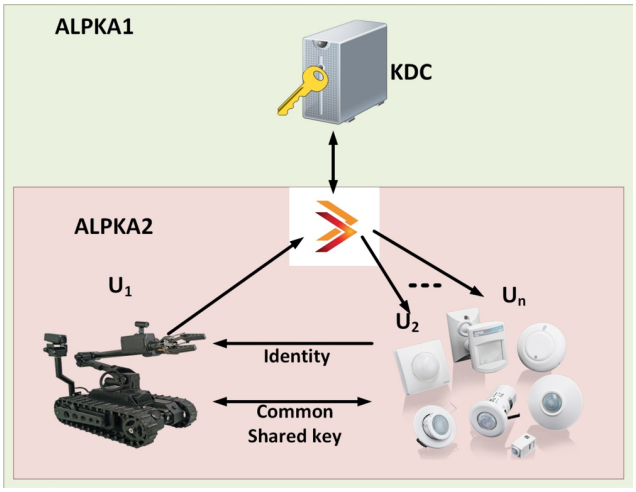

Fig. 2 Network system model for the industrial internet usecase

Paper outline

The paper is organized as follows. Section 2 describes related work. In Section 3, details are provided on the preliminaries that are used in explaining the protocol. Section 4 describes the proposed solution. Section 5 provides a brief analysis of the protocol. Section 6 presents the formal versification of the proposed protocol. In Section 7, the computational overhead of the protocol is discussed. Finally, Section 8 summarizes the paper by giving the conclusions.

2 Background and Related Work

Several standards have been proposed to establish End-to-End (E2E) security and key establishment in IoT, like Datagram Transport Layer Security [7], Internet Key Exchange (IKEv2) scheme [8], and HIP-DEX [9] protocols. An interesting approach to further improve these standards has been suggested in [10], [11] and [12]. In these papers, a proxy-based solution is proposed for delegating the heavy cryptographic operations from a constrained device to less constrained nodes in its neighbourhood. In both [10] and [11], only one set of intermediary proxies are used in the key establishment protocols. This approach is generalized in [13], where two different sets of proxies are used. The main disadvantage of these standards and their derivatives is that they still either rely on asymmetric key based operations based on the assumption of individual pre-shared secret key material, or proxies that are able to decrypt the transmitted messages and thus, able to derive the common shared session key.

Besides the standards, there have been many other server assisted key establishment protocols proposed in literature. Most of them use public key based operations, such as modular exponentiations [14], [15] and even pairing operations [16], [17]. We now focus on the domain of the symmetric key based re-encryption schemes, where we distinguish the schemes [18–20]. The approach in [18] is efficient, but insecure, as each user is able to derive the secret key of the other user and thus impersonate that user. Moreover, in [18] and [19], an unrealistic assumption is made that both sender and receiver possess in advance a common shared key. In [20], the scheme AKAPR, Authentication Key Agreement mediated by a Prxoy Re-Encryptor for IoT, has been presented. This scheme is the first symmetric key based proxy re-encryption scheme in literature, capable to establish a common shared key between two IoT devices which do not have a common prior trust relation. However, this scheme and also [18], [19] require an interactive Key Distribution Center (KDC) to derive the re-encryption key for the proxy. Due to this requirement, the KDC needs to be permanently online, increasing its vulnerability for denial of service and other related security attacks. Moreover, none of the schemes [18–20] are able to offer identity protection for the entities. An outsider, able to eavesdrop on the communication channel, is able to follow the communication flow between the different entities and potentially derive interesting patterns out of this information. Especially for the last two issues, we will give an adequate solution. Finally, there is also a recent paper [21], which is mainly limited to the authentication between user and IoT device through a completely trusted proxy. This proxy takes the role of KDC. Only symmetric key mechanisms are used in the protocol.

The technique to establish the anonymity comes from the area of smart card based server authentication protocols. There are again symmetric key based and public key based approaches in this domain. When limiting the discussion to the symmetric key based schemes and the more general multi-server authentication protocols, we can distinguish the following most relevant and

secure schemes, [22–24]. In [23], the server requires an additional communication with the KDC during each authentication. This communication is not required in [22] and [24]. The schemes of [22] and [24] are quite similar, except that [24] also includes unlinkability. The proposed re-encryption schemes in this paper are based on the techniques from [24], but have some significant changes as the setting is not a server-based, but a peer-to-peer architecture.

3 Network Architecture and Preliminaries

3.1 Setting and assumptions

We assume there are three entities participating in the system: the Key Distribution Center (KDC), the proxy P_k, and the nodes or entities $(U_1, \ldots, U_m)$. Node U_i wants to securely communicate with node U_j $(1 \leq i, j \leq m)$, using an arbitrary proxy P_k. We assume that U_i and U_j are highly resource constrained. Note that U_j can also be an external more powerful entity, requesting information on one of the entities in the network. The KDC is responsible for generating and sharing the secret key material with the different nodes and the proxies. In the case of the interactive KDC, the proxy communicates with the KDC to receive the re-encryption key. Figures 1 and 2 illustrate the network system model for the 2 versions of the ALPKA scheme in both the healthcare domain and industrial internet respectively.

We further assume that the KDC is fully trusted and the proxy semi-trusted. This means that the proxy will execute all the required actions as prescribed in the protocol, but is also curious to derive the content of the ciphertexts and the common shared session key.

3.2 Cryptographic requirements

Proxy re-encryption schemes are characterized according to different criteria [16, 25]. We also add the anonymity and unlinkability feature to the list, as they are more and more considered as important requirements in current IoT systems [3]. These criteria are enumerated below:

- **Directionality:** There are two options. If the re-encryption key of the proxy can only be used in one direction by the proxy, it is called unidirectional. In contrast, a bidirectional scheme allows the proxy to reuse the re-encryption key for messages between the same two entities, independent of their role of sender or receiver.
- **Interactivity:** In a non-interactive scheme, the sender can generate a re-encryption key independent of the participation of the KDC or the proxy. In an interactive scheme, the active participation of both proxy and KDC is required.
- **Usability:** If the proxy re-encryption scheme can re-encrypt a ciphertext multiple times for different entities, it is called multiple-use. If the proxy is able to perform only one re-encryption, it is called a single use scheme.

- **Transitivity:** In a transitive proxy re-encryption scheme, the proxy can derive the proxy re-encryption key for communication from node U_i to U_l, given the proxy re-encryption keys for communications from nodes U_i to U_j and U_j to U_i. If not, the scheme is called non-transitive.
- **Collusion resistant:** If a node collaborates with the proxy, it is able to decrypt all the messages sent in the system.
- **Anonymity:** From the transmitted messages, no information on the identity of the sender and the intended receiver can be derived by an outsider.
- **Unlinkabiity:** When collecting the transmitted messages over a long time, no relation can be made between them to link the messages with the same sender or receiver.

3.3 Operations

The operations used in the scheme are limited to simple and symmetric key based operations. The operation $H : \{0,1\}^* \Rightarrow \{0,1\}^n$ is a one-way cryptographic hash function (eg. SHA2 or SHA3). An authenticate encryption algorithm is used to encrypt the message (e.g. AES-GCM or AES-CCM).

The concatenation of two messages M_1 and M_2 is denoted by $M_1 \| M_2$. We denote by M the message sent by the sender. The ciphertext C, obtained by the encryption of M through a symmetric key encryption algorithm using key K, is defined as $C = E_K(M)$. Similar, the decryption by the symmetric key K is denoted by $M = D_K(C)$. We assume that these functions are implemented in each entity and proxy which are participating in the scheme.

Table 1 summarises the most important notations to be used in the rest of the paper.

Table 1 Notations and abbreviations

Abbreviation	Explanation
x, y, z	Master secret keys of KDC
ID_i	Real identity of U_i
A_i	Secret identity of U_i, known by KDC and proxy
B_i	Public identity of U_i
s_i	Secret key between U_i and KDC
$H(x)$	Secret parameter known by all U_i and KDC
ID_k	Public identity of proxy P_k
$H(z\|y)$	Secret parameter, known by all proxies and KDC
$H(ID_k\|H(x))$	Secret value stored by proxy P_k
rk_{ij}	Re-encryption key between U_i and U_j
M, C	Message and ciphertext

4 Proposed Key Agreement Protocol

A symmetric key based proxy re-encryption scheme consists of five phases, being the Key Generation (KeyGen), the Re-encryption Key Generation (ReKeyGen), the Encryption (Encrypt), the Re-Encryption (ReEncrypt), and Decryption (Decrypt) phase. The KDC will be responsible for the KeyGen and ReKeyGen phases, the sender for the Encrypt phase, the proxy for the ReEncrypt phase, and the receiver for the Decrypt phase. We suppose that U_i will delegate the decryption right of the ciphertext C to U_j with the help of the proxy P_k, resulting in the derivation of the common shared session key.

We now describe these different phases of our proposed key agreement protocols into detail. Note that we explain both schemes together as they are very similar and differ only at a few places. In the first version of the scheme, also called ALPKA 1, the proxy needs an interactive KDC to receive a re-encryption key. In the second case, ALPKA 2, the proxy does not need a separate re-encryption key and thus, corresponds with a non interactive scheme.

4.1 Key Generation (KeyGen)

We suppose the KDC possesses three master secret values x, y, z. The KDC shares a different type of information to the entities U_i and to the proxies P_k. We discuss both, the initialisation of the key material and the required updates.

- **Initialization:** For each entity U_i with serial number ID_i, the KDC computes the following information

$$\begin{aligned} A_i &= H(ID_i \| y) \\ B_i &= H(z\|y) \oplus A_i \\ s_i &= H(A_i \| z) \end{aligned}$$

  The KDC securely sends the information $B_i, H(A_i), s_i, H(x)$ to the entity U_i. From now on, the identity of the entities is replaced by the publicly known value B_i. The other values $H(A_i), s_i, H(x)$ are kept secret.
  For the proxies P_k with identity ID_k, the KDC sends the information $H(z\|y)$ and $H(ID_k\|H(x))$. The identity of the proxy ID_k is public, while the values $H(z\|y)$ and $H(ID_k\|H(x))$ are kept secret.
  Note that a secure channel between the KDC and each of the entities, the KDC and each of the proxies is required during this phase. For the entities, this can be established by pre-installation, while for the proxies standardized security protocols can be applied as they are not considered as constrained devices.
- **Update:** As the value $H(x)$ is shared among all devices in the network, a regular update on the secret parameter x is required. Therefore, a unicast message containing the new secret value will be send from the KDC to

the entity U_i, using the shared secret key s_i. Simultaneously, updates on $H(ID_k \| H(x))$ will be send to the different proxies.
Also regular updates on $H(z\|y)$ are required, as this value is shared among the different proxies. In order to avoid an impact on a change of the value A_i of the entities, and thus indirectly on the values $H(A_i), s_i$ that are stored in the nodes, it is sufficient to only update the parameter y. Consequently, an update on y, influences $H(z\|y)$ of the proxies and also only the value B_i on the side of the entities.

4.2 Re-encryption Key Generation (ReKeyGen)

Note that the re-encryption key is only required for the ALPKA 1 scheme. In ALPKA 2, this key does not exist. The re-encryption key for the communication between entities U_i and U_j, denoted by rk_{ij}, is determined by

$$rk_{ij} = (H(s_i \| B_j) \oplus H(s_j \| B_i))$$

Consequently, it holds that $rk_{ij} = rk_{ji}$. This leads to a bidirectional scheme, as also discussed further in the next section.

4.3 Encryption (Encrypt)

First, the entity U_i, chooses a random value $t \in Z_p$ and then computes, using its secret value s_i and the public identity B_j of U_j, the random value:

$$N_i = t \oplus H(s_i \| B_j)$$

Next, the following computations are performed:

$$\begin{aligned} C_1 &= H(ID_k \| H(x)) \oplus H(B_i \| N_i) \\ CID_i &= B_i \oplus H(H(ID_k \| H(x)) \| H(B_i \| N_i)) \\ C_2 &= H(H(A_i) \| C_1)) \oplus N_i \\ C_3 &= H(C_1 \| H(A_i)) \oplus B_j \\ C &= E_{K_i}(M) \end{aligned}$$

For the key K_i in the last encryption, there are two possibilities $i = 1$ and $i = 2$, leading to the definitions of

$$\begin{aligned} K_1 &= H(t) \\ K_2 &= H(B_i \| B_j \| H(x) \| H(N_i)) \end{aligned}$$

The key K_1 is used in the ALPKA 1, while the key K_2 in the ALPKA 2.

Finally, as last step in the Encrypt phase, the message $\{C_1, C_2, C_3, CID_i, C\}$ is sent to the proxy P_k. Here CID_i is the part of the message that refers to the dynamic identity of B_i. The dynamic character is obtained based on the value C_1. The parameter C_2 hides the random value N_i that is used in the key definition, and C_3 hides the identity of the receiver B_j.

4.4 Re-Encryption (ReEncrypt)

This phase can be split into two parts. The first part is required to check the message of the sender, while in the second part the preparation of the message to be forwarded to the receiver is made. Below, we discuss into detail the actions to be performed by the proxy in this phase.

Upon arrival of the message $\{C_1, C_2, C_3, CID_i, C\}$, the proxy executes the following steps:

$$\begin{aligned} H(B_i\|N_i) &= H(ID_k\|H(x)) \oplus C_1 \\ B_i &= CID_i \oplus H(H(ID_k\|H(x))\|H(B_i\|N_i)) \\ A_i &= H(z\|y) \oplus B_i \\ B_j &= H(C_1\|H(A_i)) \oplus C_3 \\ N_i &= H(H(A_i)\|C_1)) \oplus C_2 \end{aligned}$$

In the ALPKA 2 scheme, the proxy needs to communicate N_i to the entity U_j, while in the ALPKA 1 scheme, this value should be updated by means of the re-encryption key rk_{ij} (generated in the ReKeyGen phase).

$$N_i = N_i \oplus rk_{ij} = t \oplus H(s_j\|B_i)$$

Figure 3 gives a schematic overview of the different steps to be performed in the Encrypt, the ReKeyGen, and the first part of the ReEncrypt phase by the sender U_i and proxy P_k. The result of the KeyGen phase is also depicted. Note that the public part of the key material, present at each entity and proxy, is on the first line and the secret part on the second line in the figure. Finally, the only difference up to now between ALPKA 1 and ALPKA 2 is in the definition of N_i, where for ALPKA 1 the ReKeyGen phase is required to use the resulting re-encryption key for updating N_i.

In the second part of the ReEncrypt phase, the following computations are executed by the proxy:

$$\begin{aligned} A_j &= H(z\|y) \oplus B_j \\ C_4 &= H(H(A_j)\|C) \oplus N_i \\ C_5 &= H(B_j\|N_i)) \oplus B_i \end{aligned}$$

Finally, the message containing $\{C_4, C_5, C\}$ is sent to U_j. Here, the parameter C_4 is meant to hide the random value N_i to be used in the key definition, while C_5 hides the identity of the sender B_i.

4.5 Decryption (Decrypt)

Upon arrival of this message, U_j performs the following computations:

$$\begin{aligned} N_i &= H(H(A_j)\|C) \oplus C_4 \\ B_i &= H(B_j\|N_i) \oplus C_5 \end{aligned}$$

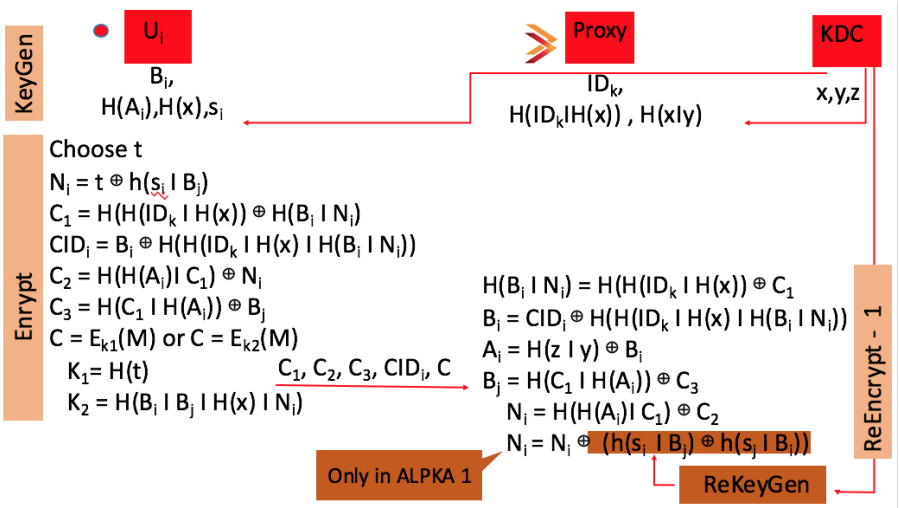

Fig. 3 Schematic overview of the cryptographic operations in the ReKeyGen, Encrypt, and first part of the ReEncrypt phase by the sender U_i and proxy P_k. The resulting key material from the KeyGen phase is also presented.

Only for the ALPKA 1 scheme, U_j needs to derive in addition the value t, which equals due to the definition of the re-encryption key, to

$$t = N_i \oplus H(s_j \| B_i)$$

Consequently, for the decryption of the ciphertext C, the shared secret key should be derived, which equals to K_1 in the ALPKA 1 and K_2 in the ALPKA 2. The exact values are determined by:

$$K_1 = H(t)$$
$$K_2 = H(B_i \| B_j \| H(x) \| H(N_i))$$

Figure 4 gives a schematic overview of the different steps to be performed by the proxy P_k and receiver U_j, in the second part of the ReEncrypt phase and the DeCrypt phase.

5 Protocol Analysis

This section discusses the strength of the ALPKA 1 and ALPKA 2 protocols against the required security features of a proxy re-encryption scheme, as mentioned in Section 3.

– **Directionality:** In the case of the ALPKA 1 scheme, where the proxy needs to receive a re-encryption key of the KDC, the scheme is bidirectional.

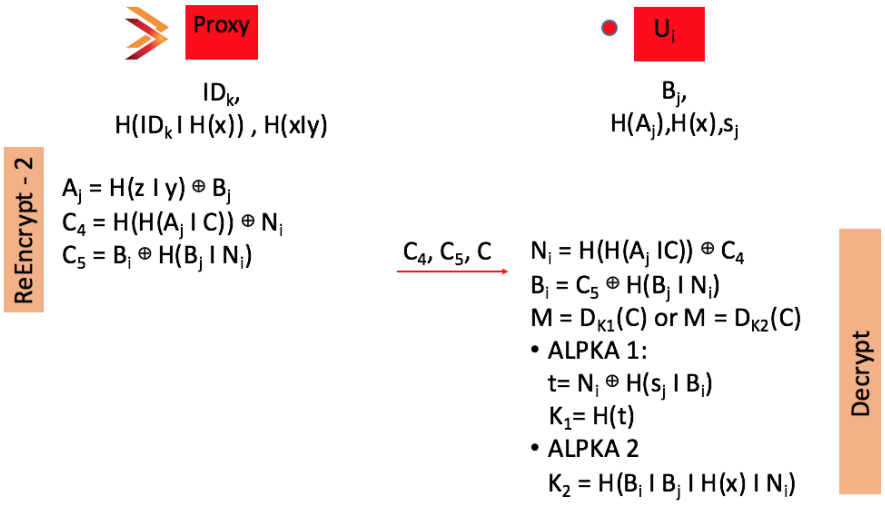

Fig. 4 Schematic overview of the cryptographic operations in the second part of the ReEncrypt phase and the DeCrypt phase, performed by the proxy P_k and receiver U_j.

Both for the communication from U_i to U_j and vice versa, the re-encryption scheme is similar, due to construction, i.e.

$$\begin{aligned} rk_{ij} &= (h(s_i \| B_j) \cdot h(s_j \| B_i))^{-1} \\ &= rk_{ji} = (h(s_j \| B_i) \cdot h(s_i \| B_j))^{-1} \end{aligned}$$

In the ALPKA 2 scheme, this feature is not applicable since there is no re-encryption key used.

- **Interactivity:** The presence of this feature is one of the main differences between ALPKA 1 and ALPKA 2 schemes. The protocol ALPKA 1 is not interactive as for the computation of the key K_1 by the receiver, since the proxy first needs to use additional information (proxy re-encryption key) coming from the KDC. For the key K_2 in ALPKA 2 on the other hand, it suffices to securely forward the random value N_i, generated by the sending entity. Unfortunately, we explain later that it is not possible to combine the non-interactive property with a collusion resistant feature.
- **Usability:** The ALPKA 1 scheme cannot be made multiple use, as the proxy requires the knowledge of the combined secret information from solely the sender and the receiver for the re-encryption key.
  The ALPKA 2 scheme could allow multiple use if the construction of the key slightly changes. For instance, if $K_2 = H(B_i \| H(x) \| H(N_i))$, i.e., if the information of the receiver is removed from the definition of the key. However, in this case, the sender does not have any control on who will receive the message, and thus this feature might not be very interesting.
- **Transitivity:** As this feature involves a property on the proxy re-encryption key, it is only relevant for ALPKA 1. Transitivity is obtained as the re-encryption key combines the secret information of both sender and receiver

by multiplication. Since the single secret information of one of the two is not known and cannot be extracted by combining multiple re-encryption keys from different pairs of senders and receivers, it is impossible to construct a new re-encryption key from this information.

- **Collusion resistant:** First of all, a compromised proxy cannot generate new authorized entities, as the construction of the values A_i require the knowledge of the master key y of the KDC. Also a compromised node on itself is not able to decrypt messages (not intended to him). It is only able to derive the identity of the sender and the receiver from the transmitted messages. For the impact on the security when both a compromised proxy and entity collaborate, the two ALPKA schemes are discussed separately.
  - For the computation of K_1 in ALPKA 1, the secret information of either sender or receiver is required. Suppose a malicious user collaborates with the proxy, it is only able to derive the messages sent by himself or sent to a particular receiver as the re-encryption key involves the secret information of both the sender and the receiver. For the transmission of information between other entities, their combined knowledge allows only to the malicious user to derive the identity of the sender and receiver as explained later, but cannot be exploited to decrypt their messages.
  - For the derivation of K_2 in ALPKA 2, no individual secret information of the sender s_i or receiver s_j is used. Its secrecy mainly depends on two types of values, which are asymmetrically divided among the entities and the proxy. First, there is the value $H(x)$, which is shared among all authorized users and not the proxy. Second, there are also the individual secret data $H(A_i), H(A_j)$, which are only shared between the proxy and the sender and receiver respectively since the proxy possesses the key $H(z\|y)$ to transform the public identity information B_i, B_j into the values A_i, A_j. Consequently, not the proxy or another authorized entity besides B_i, B_j is able to decrypt the transmitted ciphertext C as the combined knowledge of these two types of values is required.
    However, when a malicious user collaborates with the proxy, this asymmetric division of the key material no longer holds and the proxy gets also aware of the value $H(x)$, making it possible to derive the secret key K_2. Also the malicious user gets to know $H(z\|y)$, allowing them to derive the values of $H(A_i), H(A_j)$, and thus to compute the secret key K_2.
- **Anonymity:** The identity of the entity ID_i corresponds with the publicly shared pseudo identity B_i. Let us analyze the information, which is derived from the message sent by U_i and the message sent by the proxy.
  - Message $\{C_1, C_2, C_3, CID_i, C\}$. The information related to the identity of the sender B_i is hidden in the values C_1, CID_i. Here, C_1 is required in order to make the identity related information of the sender dynamic by including the random value N_i.
    The value C_3 hides the information on the identity of the receiver, through the usage of the secret shared value $H(A_i)$ with the proxy. It

is made dynamic by including a part of the message C_1 into the hash value.
Finally, as the receiver will be able to derive N_i in the end, it is important that the value $H(A_i)$ cannot be directly derived by the receiver from this message. Therefore, $H(A_i)$ is incorporated in a hash containing another dynamic value, like in C_2.
- Message $\{C_4, C_5, C\}$. Only the identity related information B_i of the sender is hidden in the message part C_5, since the receiver already knows that the information is coming from the proxy. Again, this value C_5 is made dynamic by the inclusion of a random value.

– **Unlinkability:** As explained before, due to the usage of secret key material and the inclusion of a random value in the message parts CID_i, C_1, C_3, C_5, allowing the derivation of the identity related information of sender B_i and receiver B_j, the messages are unlinkable.

Table 2 compares the above described list of security features with the other symmetric key based proxy re-encryption schemes [18–20] in literature.

Table 2 Comparison of security features with related work

Feature	[18]	[19]	[20]	ALPKA 1	ALPKA 2
Directionality	bi-d	bi-d	bi-d	bi-d	na
Non-interactivity	No	No	No	No	Yes
Multiple use	Yes	No	No	No	Possible
Non-transitivity	No	Yes	Yes	Yes	na
Collusion resistance	No	No	Yes	Yes	No
Anonymity	No	No	No	Yes	Yes
Unlinkability	No	No	No	Yes	Yes

As can be concluded from Table 2, our ALPKA 1 and ALPKA 2 schemes add the anonymity and unlinkability features to the existing state of the art schemes. In addition, ALPKA 2 is the first non-interactive symmetric key based proxy re-encryption scheme. However, this feature is not compatible with collusion resistance. Consequently, there is also the ALPKA 1 scheme, which adds the anonymity and unlinkability features, while still resisting collusions, but requiring continuous interaction between the proxy and the KDC. Note that instead of requiring the continuous interaction of the KDC, making it vulnerable to a whole range of security attacks, the proxy can also store all possible re-encryption keys for each pair of entities in the system, as proposed in [20].

Since in ALPKA 2 no re-encryption key is used, the features of directionality and non-transitivity are not applicable.

Finally, we also want to note that the communication overhead in the ALPKA protocols is limited to one additional message part C_1, enabling the unlinkability of the messages. All other message parts correspond with required

information: the identity of sender, receiver, additional information for the key derivation, and ciphertext.

6 Formal Verification of the ALPKA Protocol

In this section we formally verify the correctness of our proposed protocol by formally analysing the security goals of the protocol (i.e., authentication, session-key establishment) We are using an automated modal logic-based of knowledge CDVT/AD [4], [5], [28], [29], [30], [31].

The CDVT/AD verification tool is an automated system that implements a modal logic of knowledge [29] and an attack detection theory for identification of freshness and man-in-the-middle attacks [30]. The tool uses a proving engine based on the Layered Proving Trees concept [31] and can analyze the evolution of knowledge and belief during a protocol execution. Therefore, it is useful in addressing issues of both security and trust.

Prior to the automated verification using the CDVT logic of knowledge, the scheme must be formalized, i.e. translated into the language of the tool [5]. A formalized protocol consists of three components:

1. Initial assumptions (conditions that hold before the protocol starts);
2. Protocol steps (the messages exchanged between the principals);
3. Protocol goals (conditions that are expected to hold if the protocol terminates successfully).

The following notations are used when translating the scheme into the language of the CDVT/AD tool:

Key Distribution Center: KDC
Trusted Third Party: TTP
User Ui: Principal Ui;
User Uj: Principal Uj;
Proxy Pk: Sk;
Secrets possessed by KDC x, y, z: Nx, Ny, Nz

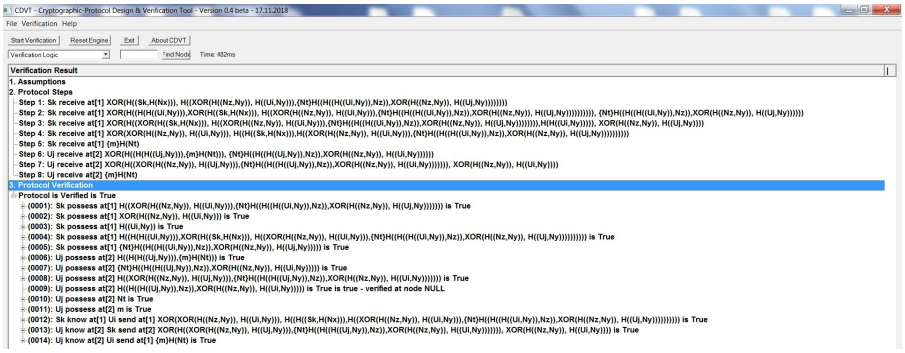

Fig. 5 Security Goals Verification Results

Hash Function H(.): H()
Identity of user Ui, IDi: Ui
Identity of proxy Pk, IDk: Sk
XOR: $\oplus$
Symmetric encryption EK(m): {m}k
send: $\rightarrow$
concatenation: $\|$

The CDVT/AD tool applies the axioms and rules of the implemented logic of knowledge in an attempt to derive the protocol goals as a logical consequence of the initial assumptions and the protocol steps. If such a derivation exists, the verification is successful and the verified protocol can be considered secure within the scope of the logic.

6.1 Formalization of the Proposed Scheme

6.1.1 Initial Assumptions

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run. The following specifies the initial assumptions of the scheme:

Express TTP possessions at time t0

- Ai = H(Ui,Ny)
- Bi = H(Nz,Ny) XOR H(U,Ny)
- si = H(H(U,Ny),Nz)

A1: TTP possess at [0] Nx;
A2: TTP know at [0] NOT (Zero possess at [0] Nx);
A3: TTP possess at [0] Ny;
A4: TTP know at [0] NOT (Zero possess at [0] Ny);
A5: TTP possess at [0] Nz;
A6: TTP know at [0] NOT (Zero possess at [0] Nz);
A7: TTP possess at [0] H(Ui,Ny);
A8: TTP know at [0] Ui possess at [0] H(Ui,Ny);
A9: TTP possess at [0] XOR(H(Nz,Ny), H(Ui,Ny));
A10: TTP know at [0] Ui possess at [0] XOR(H(Nz,Ny), H(Ui,Ny));
A11: TTP possess at [0] H(H(Ui,Ny),Nz);
A12: TTP know at [0] Ui possess at [0] H(H(Ui,Ny),Nz);
A13: TTP possess at [0] H(Uj,Ny);
A14: TTP know at [0] Uj possess at [0] H(Uj,Ny);
A15: TTP possess at [0] XOR(H(Nz,Ny), H(Ui,Ny));
A16: TTP know at [0] Uj possess at [0] XOR(H(Nz,Ny), H(Uj,Ny));
A17: TTP possess at [0] H(H(Uj,Ny),Nz);
A18: TTP know at [0] Uj possess at [0] H(H(Uj,Ny),Nz);

Express Ui possessions at time t0

- Bi= XOR(H(Nz,Ny), H(Ui,Ny))
- H(Ai)= H(H(Ui,Ny))
- si= H(H(Ui,Ny),Nz)

A19: Ui possess at [0] Uj;
A20: Ui possess at [0] Sk;
A21: Ui possess at [0] XOR(H(Nz,Ny), H(Ui,Ny));
A22: Ui possess at [0] H(H(Ui,Ny));
A23: Ui possess at [0] H(H(Ui,Ny),Nz);
A24: Ui possess at [0] H(Nx);
A25: Ui possess at [0] Nt;
A26: Ui know at [0] NOT ZERO possess at [0] Nt;

Express Uj possessions at time t0

- Bj= XOR(H(Nz,Ny), H(Uj,Ny))
- H(Aj)= H(H(Uj,Ny))
- sj= H(H(Uj,Ny),Nz)

A27: Uj possess at [0] Ui;
A28: Uj possess at [0] Sk;
A29: Uj possess at [0] XOR(H(Nz,Ny), H(Uj,Ny));
A30: Uj possess at [0] XOR(H(Nz,Ny), H(Ui,Ny));
A31: Uj possess at [0] H(H(Uj,Ny));
A32: Uj possess at [0] H(H(Uj,Ny),Nz);
A33: Uj possess at [0] H(Nx);
A34: Uj possess at [0] H(H(H(Uj,Ny),Nz), XOR(H(Nz,Ny), H(Ui,Ny)));

Express Sk possessions at time t0

A35: Sk possess at [0] H(Nz,Ny);
A36: Sk possess at [0] H(Sk,H(Nx));

6.1.2 Scheme Steps

The proposed scheme steps are formalized as follows:
Step 1: C1, C2, C3, CIDi, C

S1: Sk receive at [1] XOR(H(Sk,H(Nx)), H(XOR(H(Nz,Ny), H(Ui,Ny)),Nt H(H(H(Ui,Ny),Nz), XOR(H(Nz,Ny), H(Uj,Ny)))));
S1: Sk receive at [1] XOR(H(H(H(Ui,Ny)), XOR(H(Sk,H(Nx)), H(XOR(H(Nz,Ny), H(Ui,Ny)),Nt H(H(H(Ui,Ny),Nz), XOR(H(Nz,Ny), H(Uj,Ny)))))),Nt H(H(H(Ui,Ny),Nz), XOR(H(Nz,Ny), H(Uj,Ny))));
S1: Sk receive at [1] XOR(H(XOR(H(Sk,H(Nx)), H(XOR(H(Nz,Ny), H(Ui,Ny)), Nt H(H(H(Ui,Ny),Nz), XOR(H(Nz,Ny), H(Uj,Ny))))), H(H(Ui,Ny))), XOR(H(Nz,Ny), H(Uj,Ny)));
S1: Sk receive at [1] XOR(XOR(H(Nz,Ny), H(Ui,Ny)), H(H(Sk,H(Nx)), H(XOR(H(Nz,Ny), H(Ui,Ny)), Nt H(H(H(Ui,Ny),Nz), XOR(H(Nz,Ny), H(Uj,Ny)))));
S1: Sk receive at [1] {m}H(Nt);

Step 2: C4, C5, C

S2: Uj receive at [2] XOR(H(H(H(Uj,Ny)), mH(Nt)), NtH(H(H(Uj,Ny),Nz), XOR(H(Nz,Ny), H(Ui,Ny))));
S2: Uj receive at [2] XOR(H(XOR(H(Nz,Ny), H(Uj,Ny)), NtH(H(H(Uj,Ny),Nz), XOR(H(Nz,Ny), H(Ui,Ny)))), XOR(H(Nz,Ny), H(Ui,Ny)));
S2: Uj receive at [2] {m}H(Nt);

6.1.3 Security Goals

The objective of the proposed scheme is the establishment of the session keys and the authentication of the users. The formalized goals of the scheme are as follows:

Verify if the establishment of the session keys is done correctly (i.e. secrecy of the key components and their possessions by the legitimate users)

G1: Sk possess at [1] H(XOR(H(Nz,Ny), H(Ui,Ny)),NtH(H(H(Ui,Ny),Nz),XOR(H(Nz,Ny), H(Uj,Ny))));
G2: Sk possess at [1] XOR(H(Nz,Ny), H(Ui,Ny));
G3: Sk possess at [1] H(Ui,Ny);
G4: Sk possess at [1] H(H(H(Ui,Ny)),XOR(H(Sk,H(Nx)), H(XOR(H(Nz,Ny),H(Ui,Ny)), NtH(H(H(Ui,Ny),Nz), XOR(H(Nz,Ny), H(Uj,Ny))))));
G5: Sk possess at [1] NtH(H(H(Ui,Ny),Nz), XOR(H(Nz,Ny), H(Uj,Ny)));
G6: Uj possess at [2] H(H(H(Uj,Ny)), mH(Nt));
G7: Uj possess at [2] NtH(H(H(Uj,Ny),Nz), XOR(H(Nz,Ny), H(Ui,Ny)));
G8: Uj possess at [2] H(XOR(H(Nz,Ny), H(Uj,Ny)), NtH(H(H(Uj,Ny),Nz),XOR(H(Nz,Ny), H(Ui,Ny))));
G9: Uj possess at [2] H(H(H(Uj,Ny),Nz),XOR(H(Nz,Ny), H(Ui,Ny)));
G10: Uj possess at [2] Nt;
G11: Uj possess at [2] m;

Authentication of users

- Sk authenticate Ui

  G12: Sk know at [1] Ui send at [1] XOR(XOR(H(Nz,Ny), H(Ui,Ny)), H(H(Sk,H(Nx)), H(XOR(H(Nz,Ny), H(Ui,Ny)),Nt H(H(H(Ui,Ny), Nz), XOR(H(Nz,Ny), H(Uj,Ny))))));

- Uj authenticate Sk

  G13: Uj know at [2] Sk send at [2] XOR(H(XOR(H(Nz,Ny), H(Uj,Ny)), Nt H(H(H(Uj,Ny),Nz), XOR(H(Nz,Ny), H(Ui,Ny)))), XOR(H(Nz,Ny), H(Ui,Ny)));

- Uj authenticate Ui

  G14: Uj know at [2] Ui send at [1] {m}H(Nt);

6.2 Verification Results

The results of the automated verification for the above formalized scheme are shown in Figure 4. Goals G1-G11 relate to the establishment of the session keys and check for the secrecy of the key components and their possessions by the legitimate corresponding users, while goals G12-G14 relate to the authentication of the users. As presented in Figure 5, all security goals are successfully verified. This provides confidence in the correctness and effectiveness of the proposed protocol.

7 Performance analysis

In this section we discuss the computational complexity of our scheme and compare it with [20]. As mentioned before, [20] is the only other symmetric key based proxy re-encryption scheme in literature which does not assume that the sender and receiver possess in advance a prior secret key.

From the sender side, both ALPKA 1 and ALPKA 2 require 8 hash operations and 1 encryption. From the receiver side, 1 encryption and 5 hashes are required in ALPKA 1, while 1 encryption and 4 hashes in ALPKA 2. Taking into account that Key Derivation Functions (KDF) and Message Authentication Codes (MAC) have similar complexity as a hash operation, therefore [20], requires 2 encryptions and 5 hashes at sender side and 2 encryptions and 4 hashes at receiver side.

In order to get a better understanding of the impact of these numbers, we consider measurements performed on a smartphone with Android OS (Android 4.4 KitKat) consisting of a 2260 MHz ARM device. The timings for both AES128 as encryption algorithm and SHA2 as hash operation, correspond with 0.1 ms [32]. Consequently, both schemes have comparable and negligible computational complexity.

8 Conclusions

This paper proposes two versions of anonymous lightweight proxy based key agreement protocols, called ALPKA 1 and ALPKA 2. They are lightweight as only symmetric key based operations are involved in our proposed protocols and have limited communication overhead. Both protocols allow highly constrained entities, that have never met before in order to share a secret key in an anonymous and unlinkable way through a proxy. This proxy is responsible for the authentication of the entities and the further derivation of the required key material allowing the decryption by the receiver, while not being capable of decrypting the message itself. The main difference between the two versions is the combination of interactivity and collusion resistance versus non-interactivity and not being resistant to collisions. As continuous interactivity between the proxy and the KDC cannot always be ensured, it is important

to give an alternative with some small cost in security. Moreover, the security properties of the proposed protocol are verified by using formal verification techniques. This provides confidence in the correctness and effectiveness of the proposed protocol.

As future work, we plan to investigate how the proposed protocols can be applied in the context of internet-connected cars. As our scheme is highly efficient, it offers low latency and is very scalable, we believe that it might offer a good solution to this domain, considering the high number of vehicles required to communicate in real-time.

Acknowledgement

This work has been performed under the framework of COST Action CA15127 (RECODIS) and CA16226 (SHELD-ON) projects.

References

1. L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey", Computer Networks, vol. 54, no. 15, pp. 2787-2805, 2010.
2. D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of Things: Vision, Applications and Research Challenges", Ad Hoc Networks, vol. 10, no. 7, pp. 1497?1516, 2012.
3. X. Caron, R. Bosua, S.B. Maynard, A. Ahmad, "The Internet of Things (IoT) and Its Impact on Individual Privacy: An Australian perspective", Computer Law and Security Review, vol. 29, no. 32(1), pp. 4-15, 2016.
4. A.D.Jurcut, Tom Coffey, and Reiner Dojen. "On the prevention and detection of replay attacks using a logic-based verification tool." In International Conference on Computer Networks, pp. 128-137. Springer, Cham, 2014.
5. A.D.Jurcut, Tom Coffey, and Reiner Dojen, "Establishing and Fixing Security Protocols Weaknesses using a Logic-based Verification Tool", Journal of Communication, Vol. 8, No. 11, ISSN 1796-2021, pp. 795-806, November 2013, DOI: 10.12720/jcm.8.11.795-805
6. A.D.Jurcut, T. Coffey, and R. Dojen, "Design Guidelines for Security Protocols to Prevent Replay & Parallel Session Attacks", Journal of Computers & Security, Elsevier, Volume 45, pp. 255273, September 2014, DOI: 10.1016/j.cose.2014.05.010.
7. H. Tschofenig and T. Fossati, "A TLS/DTLS 1.2 Profile for the Internet of Things", IETF draft, RFC editor, 2013, `http://tools.ietf.org/html/draft-ietf-dice-profile-09i`.
8. C. Kaufman, "Internet Key Exchange (IKEv2) Protocol",IETF RFC 7296, 2014, `http://tools.ietf.org/html/rfc7296i`.
9. R. Moskowitz, "HIP Diet EXchange (DEX)", IETF draft, RFC editor, 2014, `http://tools.ietf.org/html/draft-moskowitz-hip-dex-02i`.
10. Y. Saied and A. Olivereau, "D-HIP: A distributed key exchange scheme for HIP-based Internet of Things", Proceeding of IEEE World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1-7, 2012.
11. Y. B. Saied, A. Olivereau, D. Zeghlache, and M. Laurent, "Lightweight Collaborative Key Establishment Scheme for the Internet of Things", Computer Networks, vol. 64, no. 0, pp. 273-295, 2014.
12. P. Porambage, A. Braeken, A. Gurtov, M. Ylianttila, and S. Spinsante, "Secure end-to-end communication for constrained devices in IoT-enabled Ambient Assisted Living systems", IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, 2015, pp. 711-714.

13. P. Porambage, A. Braeken, P. Kumar, A. Gurtov, and M. Ylianttila, "Proxy-based End-to-End Key Establishment Protocol for the Internet of Things", Proceedings of IEEE ICC Workshop on Security and Privacy for Internet of Things and Cyber-Physical Systems, 2015.
14. R. Canetti and S. Hohenberger, "Chosen-Ciphertext Secure Proxy Re-encryption", Proceedings of the 14th ACM conference on Computer and communications security, pp. 185-194, 2007.
15. S.S.M. Chow, J. Weng, Y. Yang, and R.H Deng, "Efficient Unidirectional Proxy Re-encryption", Progress in Cryptology AFRICACRYPT 2010, pp. 316-332, Springer, 2010.
16. M. Green and G. Ateniese, "Identity-based Proxy Re-encryption", Applied Cryptography and Network Security, pp. 288-306, Springer, 2007.
17. T. Matsuo, "Proxy Re-encryption Systems for Identity-based Encryption", Pairing-Based Cryptography, Pairing 2007, pp. 247-267, Springer, 2007.
18. D.L. Cook and A.D. Keromytis, "Conversion Functions for Symmetric Key Ciphers", Journal of Information Assurance and Security, vol. 2, pp. 41-50, 2006.
19. A. Syalim, T. Nishide, and K. Sakurai, "Realizing Proxy Re-encryption in the Symmetric World", Informatics Engineering and Information Science, pp. 259-274,Springer, 2011.
20. K.T. Nguyen, N. Oualha, M. Laurent, "Authenticated Key Agreement Mediated by a Proxy Re-encryptor for the Internet of Things", 21st European Symposium on Research in Computer Security (ESORICS 2016), 2016.
21. M. Wazid, M. Conti, M. Jo, "Design of Secure User Authenticated Key Management Protocol for Generic IoT Network", Vol 99, IEEE Internet of Things Journal, 2017
22. K. C. H. Baruah, S. Banerjee, M. P. Dutta, and C. T. Bhunia, "An Improved Biometric-based Multi server Authentication Scheme using Smart Card", International Journal of Security and Its Application, vol.9, no.1, pp.397-408, 2015.
23. F. Wen, W. Susilo, and G. Yang, "Analysis and Improvement on a Biometric-based User Authentication Scheme using Smart Cards", Wireless Personal Communications, vol. 80, pp. 1747-1760, 2015.
24. A. Braeken, "Efficient Anonym Smart Card Based Authentication Scheme for Multi-Server Architecture", International Journal of Smart Home, vol. 9, no. 9, pp. 177-184, 2015.
25. M. Blaze, G. Bleumer, and M. Strauss, "Divertible Protocols and Atomic Proxy Cryptography", Advances in Cryptology EUROCRYPT 98, pp. 127-144. Springer,1998.
26. B. Blanchet, "Automatic Verification of Correspondences for Security Protocols", Journal of Computer Security, vol. 17, no. 4, pp.363-434, 2009.
27. D. Dolev, A.C Yao, "On the Security of Public Key Protocols", IEEE Transactions on Information Theory, vol. 29, no. 2, pp. 198-208, 1983.
28. A. D. Jurcut, M. Liyanage, J. Chen, C. Gyorodi, and J. He, "On the Security Verification of a Short Message Service Protocol", 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, April 2018. DOI: 10.1109/WCNC.2018.8377349
29. T. Coffey, P. Saidha, "Logic for verifying public-key cryptographic protocols", IEE Proc.- Computers and Digital Techniques, vol.144, pp 28-32, Jan 1997.
30. A. D. Jurcut, T. Coffey, and R. Dojen, "A Novel Security Protocol Attack Detection Logic with Unique Fault Discovery Capability for Freshness Attacks and Interleaving Session Attacks", In: IEEE Transactions on Dependable and Secure Computing, IEEE Xplore, Print ISSN: 1545-5971, Online ISSN: 1545-5971, 10.1109/TDSC.2017.2725831, available under the "Early Access" on IEEEXplore, July 2017.
31. R. Dojen, and T. Coffey, "Layered Proving Trees: A Novel Approach to the Automation of Logic-Based Security Protocol Verification", ACM Transactions on Information and System Security (TISSEC), vol. 8, Issue 3, pp. 287-311, 2005.
32. L. Malina, J. Hajny, R. Fudiak, J. Hosek, "On Perspective of Security and Privacy-Preserving Solutions for the Internet of Things", Computer Networks, vol. 19, pp. 83-95, 2016.