Regular Paper

Representative System and Security Message Transmission using Re-encryption Scheme Based on Symmetric-key Cryptography

Dai Watanabe[1,a)] Hisao Sakazaki[1] Kunihiko Miyazaki[2]

Received: March 31, 2016, Accepted: September 6, 2016

Abstract: In this paper we consider the (legal) representative in governmental ICT services and propose a secure private mail box system in which a message sent to the pupil is re-encrypted by the proxy server. This process enables the representative to decrypt the message. We also show its formal description of the protocols and evaluate the security by ProVerif model checking tool.

Keywords: cryptographic protocol, re-encryption, formal analysis, ProVerif

1. Introduction

There are miscellaneous governmental services such as book reservation in library, reservation of public facilities, preventive inoculation for nurslings, tax procedures, application for disability aid, and so on. These services are provided based on paper media but the progress of ICT and popularization of information terminal push forward to change these services to be online.

In order to enjoy these online governmental service, the user is required to register his/her contact address to the service providers which are sometimes the departments of the governmental office itself and are sometimes independent organizations. Besides the convenience of the online services, to update the contact addresses is cumbersome if each service manages the address book one by one. This issue is similar to what is happen in moving house. One should have experiences to be tired of updating his/her (postal) address at many service counters. Therefore it is desirable to put in a service provider which takes charge of the management of a pair of user IDs and the corresponding online contact addresses: This is a kind of proxy. In this system, other service providers do not store the contact addresses of users but IDs. When a service provider intends to send a message to a user, he sends the message (with ID) to the proxy server. Then the proxy server refers to the address book and sends forward the message to the contact address connected to the ID. It is obvious that it becomes quite easy for users to update their contact addresses. They only have to notify the change to the proxy server.

Then let us consider the above system in terms of information security. In many cases the governmental services deal with highly private information therefore the information protection is mandatory to realize the system. The service provider stores users' IDs and their private information which is necessary to provide the service, and sends messages to the users. The content of the message may be private. The proxy server maintains users' IDs and their contact addresses, but not their private information. It is desirable that he cannot access the users' private information in terms of information control. I.e., the message sent from the service provider to the user should be encrypted end-to-end in order to keep the content to be secret other than to the service provider and to the target user.

However, there is a difficulty to realize the above mentioned system if one considers including a representative system in the real world. The representative system allows a user (a pupil) to tie himself to a representative. The representative acts for the administrative procedures instead of the pupil. An issue arises when the service provider sends an encrypted message to the user. The representative cannot decrypt the message if he/she does not have the appropriate decryption key.

Of course there are several possible solutions: The simplest answer is that the user shares another key with the representative in advance. When he receives the encrypted message, decrypts it with the key shared with the service provider, re-encrypts with the key shared with the representative, and then sends it forward to the representative. The pupil plays the most important role in the protocol in this case. This is not appropriate approach if one considers why the pupil needs the representative. Another solution is that each service provider shares the key not only with the user but also with the representative and it sends a message both to the pupil and the representative. But the removal of the representative may be an issue in this solution; Service providers tie the pull and the representative on each database so that the pupil has to make requests of removal of the representative one by one.

In this paper we study yet another solution in which the proxy server plays a role to re-encrypt the message to the user in order

1 Center for Technology Innovation-Systems Engineering, Hitachi, Ltd., Yokohama, Kanagawa 244–0817, Japan
2 Hitachi China R&D, Beijing 100190, China
a) dai.watanabe.td@hitachi.com

for the representative to be able to decrypt it.

1.1 Our Contribution

In this paper we propose a secure message transmission system in which messages are encrypted by the sender (service provider) and the proxy server re-encrypts it with the registered re-encryption key in order for the representative to be able to decrypt it. In our proposed system, the proxy server has to re-encrypt the message even for the user. It may sound costly but the cost for re-encryption is minimal because the re-encryption is executed by only an XORing of short bit string. We call it *Symmetric Key Re-Encryption Protocol* (SK-REP) in this paper.

The security of SK-REP is evaluated by formal analysis. We use ProVerif [18], which is symbolic verification tool of cryptographic protocol. In order to check the "end-to-end" security, the evaluation includes the case of key leakage at the proxy server. Nowadays an ICT system consists of many entities in practice and the information leakage at the entity in the middle is serious concern, e.g. the security breach at Heartland Payment Systems in 2008 [19].

1.2 Related Works

Our earlier works of this paper are Refs. [14], [16]. Sakazaki et al. investigated the Japanese representative systems and made the security requirement clear in Ref. [14]. He also pointed out the treatment of representative may be an issue in governmental message transmission systems and proposed a proxy re-encryption system using a stream cipher. Watanabe et al. improved the Sakazaki's re-encryption scheme and evaluated its security by ProVerif in Ref. [16]. The paper Ref. [13] highlights the legal requirements and the consistency from the legacy (paper media based) system. This paper focuses mainly on the technical viewpoint of the system based on proxy re-encryption.

Homomorphic key encryption technology plays a central role in our proposal and there are amount of precedent works. Gentry proposed fully homomorphic encryption scheme [8]. His scheme theoretically allows any information processing against ciphertext at impractical computational cost. Compared to his work, proxy re-encryption requires very restricted processing and many schemes have been proposed in order to achieve several different security properties [2], [4], [5], [6], [9], [10], [11], [12]. They are based on asymmetric key encryption mechanisms and provide rich flexibility in operation, but their computational cost for processing is still quite large, therefore not suitable for large scale systems such as public services.

There are also some schemes based on symmetric key cryptography. Angelos et al. studied symmetric key cryptography based proxy re-encryption scheme in abstract level in Ref. [1] and mentioned a stream cipher being an instance. On the other hand, they did not consider how to generate (or share) the re-encryption keystream. It is well-known that reuse of a keystream is strongly prohibited in the usage of stream cipher because at least the difference between two messages is exposed. Therefore their study is not sufficient to realize our system. In order to solve this issue, we apply Angelos's idea not to encrypt the message but to encrypt the one-time key.

Apart from Angelos's idea, Syalim et al. proposed a re-encryption key scheme based on All-Or-Nothing Transform (AONT) [15]. Syalim's scheme is more complex than SK-REP and takes more cost for re-encryption. Watanabe et al. proposed key updating system also based on AONT [17], whose encryption mechanism is similar to our proposal. However, they consider the long-time data storage in clouds and the way to transmit the encrypted data to another user is out of discussion.

1.3 Organization of this Paper

The rest of this paper is organized as follows: In the beginning we sketch the target system in Section 2. The encryption mechanism which enables proxy re-encryption is introduced in Section 3. Next the specifications of key distribution protocols and message transmission protocol are introduced in Section 4. Then we explain the modeling of the protocols and the security evaluation using ProVerif in Section 5. Finally we conclude the discussion in Section 6.

2. Target System

In the following, we call the proposed scheme by Symmetric Key Re-Encryption scheme with Proxy (SK-REP).

The proposed system is close to a private mail box. The sender sends a message (in an envelope) to the recipient via a post office and the post office puts the envelope in the mail box of the recipient. The post office is not allowed to open the envelope. In addition to that, there is an entity (a legal representative) who can access the mail box of his pupil.

In ICT system, an encryption scheme (such as the AES) is used to realize the envelope. Though its security is considered to be sufficient, key establishment between the sender and the recipient is not an easy issue if the number of entities is getting larger. It is desirable if the number of keys which have to store is small. In Ref. [14] Sakazaki et al. proposed an encryption scheme (and its operation) in which the entities except the post office have to store only a key. That means the key management is entrusted to the post office. On the other hand, they intend to the scheme being secure even if the post office is compromised at one point.

2.1 Entities and Network

There are following four entities appear in SK-REP.

Sender who represents a governmental organization and it sends official announcements to citizens.

Proxy server who organizes the portal site and proceeds re-encryption.

Recipient (Pupil) who represents a citizen and he/she receives messages.

Legal Representative of a certain recipient can read the message sent to the recipient.

These four entities are connected each other via the proxy server and shapes star network in SK-REP (See **Fig. 1**).

2.2 Usability of the Service

As mentioned in Section 1, there are three possible solutions to provide proxy-viewing from end to end:

System X The recipient shares the secret decryption key with

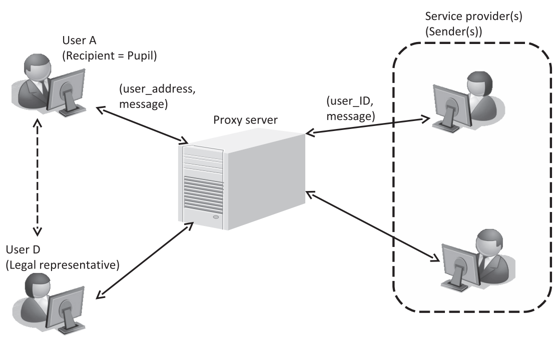

Fig. 1 Network structure of the proposed system.

his representative.

System Y The sender encrypts a message twice, i.e., with the key for the recipient and with the key for the representative.

System Z The sender encrypts a message with the key for the recipient. There is a proxy server who re-encrypts the encrypted message; the re-encrypted messages can be decrypted with the key for the representative.

In the system X when he receives the encrypted message, decrypts it with the key shared with the service provider, re-encrypts with the key shared with the representative, and then sends it forward to the representative. The pupil plays the most important role in the protocol in this case and it is not appropriate approach if one considers why the pupil needs the representative.

In the system Y, the removal of the representative may be an issue in this solution; Service providers tie the pupil and the representative on each database so that the pupil has to make requests of removal of the representative one by one.

The system Z requires a new role, a proxy server; therefore it is costly compared to other systems. On the other hand, it solves the above two problems.

2.3 Expected Adversary and Security Requirements

It is common to assume that an adversary can eavesdrop on messages, modifies them, and create/sends new message (Dolev-Yao model) in the Internet. However, Sakazaki et al. assumes that the transmitted messages are protected with standard cryptographic protocols such as IPsec and TLS. They also assume that the sender and the proxy are basically honest and do not try to threaten the security of the system. On the other hand, they seriously consider malware threats and assume that the proxy may be compromised at one point. (The incident of Heartland Payment System [19] showed that this kind of threats actually exists). After the malware infection, the proxy becomes malicious and may try to break the secrecy of transmitted messages.

In this report, we respect their assumption and focus on the security properties in data transmission phase, in which we assume Dolev-Yao model.

Under the assumptions, we claim that SK-REP satisfies the following properties:

Secrecy of transmitted message The adversary cannot recover the secret message.

Sender authentication (by Recipient) If the recipient recovers the secret message, the sender certainly has sent it before the protocol complete.

3. Re-encryption Mechanism

Angelos et al. studied proxy re-encryption scheme based on symmetric key encryption in the abstract and delivered requirements for the encryption algorithm. In Ref. [1] they also mentioned to use a stream cipher as its instance. In this section we briefly introduce Angelos' idea, and then propose the re-encryption scheme based on symmetric key encryption, the scheme which is used in our system.

3.1 Angelos's Proxy Re-encryption Scheme

Algorithm 1 shows the Angelos's idea to realize proxy re-encryption mechanism using stream cipher. This simple idea works with the help of homomorphic property of XORing.

Algorithm 1 Angelos's Proxy Re-Encryption using A Stream Cipher

```
Entities: Sender $S$, Recipient $A$, Proxy $P$,
$S$ encrypts message $M$ with keystream $KS_S$, sends the ciphertext $C_1 = M \oplus KS_S$ to $P$.
$P$ receives the ciphertext $C_1$ and re-encrypts with keystream $KS_S \oplus KS_A$, sends the ciphertext $C_2 = M \oplus KS_A$ to $A$.
$A$ receives the ciphertext $C_2$ and decrypts with keystream $KS_A$.
```

Besides, there is an issue to use this idea in practice. Angelos et al. gives only the idea of re-encryption in Ref. [1] and the way for the proxy to get (or generate) the keystream $KS_S \oplus KS_A$ is not described. Reuse a keystream to encrypt two different messages is strongly prohibited because it discloses some information about the keystream, therefore the suitable key updating mechanism must be considered.

3.2 Masked One-Time Encryption

The encryption scheme used in SK-REP is based on a symmetric key cryptography. The message is encrypted by any authenticated encryption with associated data (AEAD), which provides the secrecy of the encrypted message and the integrity of the message and the associated data, with randomly generated one-time key and then the key is masked with a fixed random string (long-term key). This construction avoids the "*do not a keystream twice*" rule by the masked data (one-time key) to be random. By intuition, only the information which the adversary can get from two ciphertext is the difference of random strings which is also random.

Definition 1

Let A be an agent and k_A be an associated random bit string of length l, $\mathtt{Enc}$ be a secure (deterministic) authenticated encryption scheme with associated data (AEAD). We define *Masked One-Time Encryption* (MOTE) by

$$\mathtt{MOTE}(m, a, k_A) = (k_A \oplus r_m)||\mathtt{Enc}(m, a, r_m),$$

where $x||y$ represents the concatenation of two data a and b, m is a message, a is an associated data (header), and r_m is a one-time encryption key, namely a randomly generated bit string of length l.

Note that the AEAD does not assures the secrecy of the associated data a, but its integrity. MOTE is secure AEAD if the underlying encryption scheme Enc is secure.

We do not describe the security proof (by hand) in this paper. Instead we point out that a similar approach of encryption has been proposed by Desai [7]. His scheme adopts All-Or-Nothing Transform (AONT) and is called All-Or-Nothing Encryption. Its security is studied from the viewpoint of computational theory and it is proved to be a secure encryption scheme. We can also adopt his scheme in SK-REP instead of MOTE.

3.3 Sketch of Re-encryption Mechanism

Let S, A, P are a sender, a recipient, and a proxy server respectively. Let k_S and k_A are the proper keys of the sender S and the recipient A. The proxy server stores the XORing of two keys $k_S \oplus k_A$. The message m is encrypted by the sender is $\mathtt{MOTE}(m, a, k_S)$. If the proxy receives the ciphertext, he XORs the stored key $k_A \oplus k_S$ to the first half of the ciphertext (namely $r_m \oplus k_S$). Then the value of the part is replaced by $r_m \oplus k_A$ so that the recipient A can decrypt the ciphertext.

4. SK-REP Specification

4.1 Key Derivation

Let N_a, N_d, N_s be the numbers of recipients, the number of recipients which a representative manages, and the number of senders respectively. If there is no key derivation mechanism, the number of keys which the representative has to manage is given by $N_a \times N_d \times N_s$. In the protocol specification in Section 4.2, we assume that each entity has only a key K and generates several keys, e.g., the key for the recipient to decrypt messages from the sender S is given by $h(k_A, S)$, where $h()$ is an arbitrary hash function.

4.2 Protocol Specifications

The proposed scheme consists of two phases: One is re-encryption key registration phase and the other is data transmission phase (**Fig. 2**). In the re-encryption key registration phase, the recipient A (and his legal representative D) registers the re-encryption key with a certain sender S. In the data transmission phase, the sender S encrypts a message and sends it to the recipient A via the proxy server P. There are two kinds of protocols in the re-encryption key registration phase: that for recipient and that for legal representative.

Figure 3 shows the rough sketch of the message sequence chart of the re-encryption key registration phase for recipient. The legal representative is abbreviated to "representative" in the above and following figures, and in the formal description given in Appendix. Remind that the communication channel is protected with a standard cryptographic protocol, but we omitted the wrap for easy understanding of the proposed scheme. The procedure of the protocol is as follows:

(1) (A recipient A and a server S generates his proper key k_A and k_S respectively before his first registration process.)

(2) A queries the proxy server P to start the key registration between he and S.

(3) P generates a nonce nP and sends it to A.

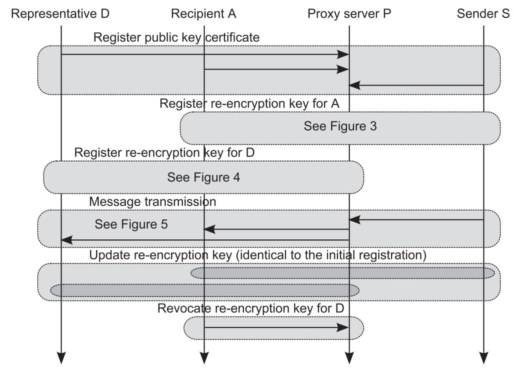
Representative D
Recipient A
Proxy server P
Sender S
Register public key certificate
Register re-encryption key for A
See Figure 3
Register re-encryption key for D
See Figure 4
Message transmission
See Figure 5
Update re-encryption key (identical to the initial registration)
Revocate re-encryption key for D

Fig. 2 Overview of the protocols used in the proxy re-encryption system.

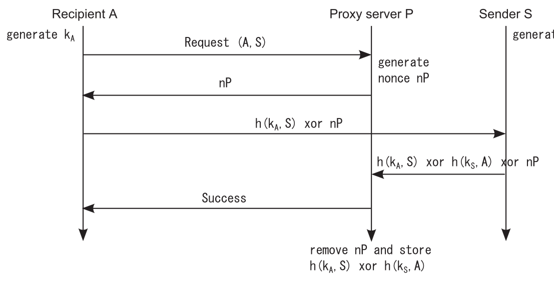

Fig. 3 The re-encryption key registration phase (for a recipient) overview.

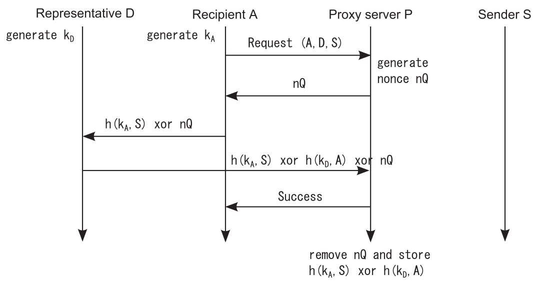

Fig. 4 The re-encryption key registration phase (for a legal representative) overview.

(4) A generates a piece of re-encryption key $h(k_A, S)$ where $h()$ is a hash function. Then he masks $h(k_A, S)$ with the nonce nP then sends it to S.

(5) S generates another piece of re-encryption key $h(k_S, A)$, XORs it to the received data $h(k_A, S) \oplus nP$ then sends it to P.

(6) P removes the mask nP and stores $(A, S, k(A, S))$ where $k(A, S) = h(k_A, S) \oplus h(k_S, A)$.

In the above we explained as if the network is a complete graph. An approach to match the assumption that the network is star shaped is to assume the existence of another trusted third party: key registration interagent I. A encrypts $h(k_A, S) \oplus nP$ (by the session key with K) and sends it to I. Then I decrypts and re-encrypts it by the session key with S, sends it to S. That means, the role of the post office is separated into two and the key registration interagent I is a part of them, and he is active only during the key registration phase. He may also play the proxy server role during the key registration phase.

Figure 4 shows the rough sketch of the message sequence chart of the re-encryption key registration phase. This protocol just replaces the sender by the legal representative in the previous one and the other things are the same. The proper keys for A can be the same as in Fig. 2.

Figure 5 shows the rough sketch of the message sequence chart

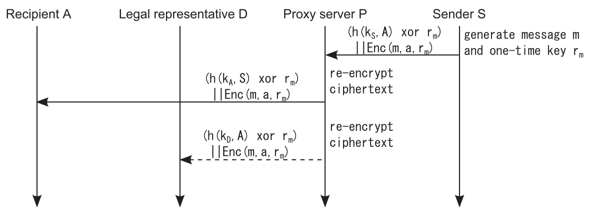
Recipient A
Legal representative D
Proxy server P
Sender S
(h(k_S, A) xor r_m) ||Enc(m, a, r_m)
generate message m and one-time key r_m
re-encrypt ciphertext
(h(k_A, S) xor r_m) ||Enc(m, a, r_m)
re-encrypt ciphertext
(h(k_D, A) xor r_m) ||Enc(m, a, r_m)

Fig. 5 Data transmission phase overview.

of the message transmission phase. The dot line (the message sent forward to the legal representative) is omitted if the recipient does not have a legal representative. The procedure of the protocol is as follows:

(1) The sender S generates a message m, a one-time encryption key r_m, and encrypts them with $h(k_S, A)$ and sends the ciphertext $\mathtt{MOTE}(m, a, k_S)$[*1] to the proxy P.
(2) P searches the re-encryption key $k(A, S)$, re-encrypts the received message with the key and sends the re-encrypted ciphertext $\mathtt{MOTE}(m, a, k_A)$ to the recipient A.
(3) A generates the decryption key $h(k_A, S)$ and decrypts the message with the key.
(4) (P searches the re-encryption key $k(D, S)$, re-encrypts the received message with the key and sends the message to the legal representative D.)
(5) D generates the decryption key $h(k_D, S)$ and decrypts the message with the key.

5. Modeling in ProVerif

In this section, we present a brief formal security evaluation of SK-REP with ProVerif model checking tool. We choose ProVerif version 1.88 [18] as an evaluation tool. ProVerif is a tool for the automatic verification of security protocols. Please refer to the web page of ProVerif for the technical background. Our model is evaluated by ProVerif without any options and it means that the evaluation level corresponds to PAL4 in Ref. [20].

5.1 Formal Verification of Cryptographic Protocol

Formal verification of the symbolic model is common approach in the security analysis of cryptographic protocols. In the symbolic model, each cryptographic primitive are considered ideal. Let $C = Enc(K, P)$ be an encryption function where K is a key, P is a plaintext and C is the corresponding ciphertext. The adversary can decrypt the plaintext P if and only if he has C and K. Different from the computational complexity theory which is typical in the analysis of cryptographic primitives, one does not consider the probabilistic event such as birthday attack so that some cryptographic properties may be lost in the analysis. Nevertheless, the symbolic analysis is powerful approach with help of automatic verification tool.

ProVerif is formal verification tool developed by Blanchet et al. [18]. It has its own description language and the model is firstly translated into Horn clauses then analyzed if there a trace satisfies all clauses.

5.2 Modeling in ProVerif Language

Here we roughly explain our model. The full description is given in Appendix A.1.

The communication channel under Delev-Yao model is declared in the following sentence:

```
free ch: channel.
```

The sent message which should be kept secret to the adversary is declared in the following sentence:

```
free s: bitstring [private].
```

`hostA, hostD, host P, hostS` denote the four entities Recipient A, Legal representative D, Proxy server P, Sender S.

The function `addr()` concatenates the name of the sender and the receiver. Two functions `sender()` and `receiver()` extract each from data of type `addr(a, b)`.

The functions `enc()` and `dec()` denote the symmetric key encryption/decryption functions[*2] `mac_gen()` and `mac_verify()` denote MAC generation/verification function. We model AEAD as a generic composition `Enc-then-Mac` as follows:

```
reduc forall m:bitstring, k:bitstring, ad:bitstring;
      aead_enc(m, k, ad) =
        (enc(m, k), mac_gen((ad, enc(m, k)), k)).
reduc forall m:bitstring, k:bitstring, ad:bitstring;
      aead_dec((enc(m,k), mac_gen((ad,enc(m,k)),k)),
        k, ad)
        = m.
```

`Enc-then-Mac` is known to be secure AEAD if the underlying encryption and message authentication algorithms are secure [3]. The encryption algorithm is deterministic in our model, i.e., the encryption algorithm does not take a nonce as its input. This is because the encryption key is always one-time in SK-REP, therefore the key also plays a nonce role.

The function `kdf()` denotes the hash function $h()$ in Section 4.1 where `kdf` stands for *key derivation function*.

The key registration protocols (for k_{SA} and k_{SAD}) are not modeled as processes, but as the key registration to the tables `reEncryptionKeysA` and `reEncryptionKeysD` as follows:

```
insert reEncryptionKeysA(hostS, hostA,
        xor(kdf(kA, hostS), kdf(kS, hostA)));
insert reEncryptionKeysD(hostS, hostA, hostD,
        xor(kdf(kD, hostA), kdf(kA, hostS)));
```

Note that the adversary is not allowed to access the tables. On the other hand, we assume Dolev-Yao model in transmission phase. This assumption relaxes the adversary model in Ref. [14] where the attack on the communication channel is not taken in account. If ProVerif does not find any attack in the evaluation of our model, it means that there is no attack even under the original adversary model.

The function `xor()` denotes XORing of two elements. ProVerif deals with a symbolic model and cannot model precisely algebraic operations such as XORing. In this paper we modeled only the commutative property of the operation by two reduction `xor_reEnc` and `xor_dec` as follows:

[*1] We assume that addresses of the sender and the recipient are included in the associated data a. On the other hand, a nonce is not necessary in SK-REP because the key for `MOTE` is one-time.

[*2] Note that `Enc()` in the message sequence charts denotes an AEAD while `enc()` in the formal model denotes an encryption scheme without integrity check.

```
fun xor(bitstring, bitstring): bitstring.
  (* reduction rules *)
reduc forall x:bitstring, y:bitstring, z:bitstring;
      xor_reEnc(xor(x, y), xor(y, z)) = xor(x, z).
reduc forall x:bitstring, y:bitstring;
      xor_dec(xor(x, y), x) = y.
```

For entities, we assumed that the legal representative is honest because he can do everything what his pupil can do. If we include the legal representative in the model, we will find amount of attacks, and this possible weakness reflects the power of the legal representative in the real world. On the other hand, we assume the proxy may leak re-encryption keys at the end of his process as follows:

```
(* re-enc. keys leakage which represents
the behavior of compromised P *)
out (ch, kSA);
out (ch, kSAD);
```

5.2.1 Possibility to Describe Security Properties

- It is easy to see that *injective agreement* of the message is not satisfied because no nonce from the recipient is used in the protocol. It is better to include the expiration date of the message for example in the associated data in order to reinforce the security.
- If the legal representative leaks his key $h(k_D, A)$ then the proxy easily recovers $h(k_A, S)$ and $h(k_S, A)$. In other words, the proposed scheme is vulnerable against such a key leakage and a collusion attack. On the other hand, the proposal paper assumes that the legal representative is not considered to be malicious and this assumption matches to the real world mechanism.

5.3 Evaluation Results

The following description is to check the secrecy of the message sent over the public communication channel, where `attacker` is the reserved predicate which implies the adversary, and `attacker(s)` means that the adversary knows `s`.

```
query attacker(s).
```

And the following description checks the authenticity of the message, which means if the event `begin_received_A(a,b,x)` is carried out then the event `begin_sender(a,b,x)` has been carried out before it.

```
query a:host,b:host,x:bitstring;
      event(message_received_A(a,b,x))
        ==>event(begin_sender(a,b,x)).
query a:host,b:host,x:bitstring;
      event(message_received_D(a,b,x))
        ==>event(begin_sender(a,b,x)).
```

With the model described in Section 5.2 we confirm that both security claims are satisfied.

Note that we do not claim the secrecy of re-encryption keys. Consider if the adversary replace the MOTE ciphertext $(k_A \oplus r_m)||\mathsf{Enc}(m, a, r_m)$ by $0\ldots0||\mathsf{Enc}(m, a, r_m)$, where $0\ldots0$ means a consecutive 0's of key length. After the re-encryption by the proxy server the adversary gets $(k_S \oplus k_A)||\mathsf{Enc}(m, a, r_m)$, i.e., he/she gets the re-encryption key. However, our evaluation takes the leakage of re-encryption keys into account, so that this attack does not threatens the secrecy of the message.

6. Conclusion

In this paper we proposed a secure message transmission system for governmental services. The encryption scheme used in the system is based on stream cipher which has been indicated by Angelos et al. and we append a small technique to make it useful in practical system. We also evaluate the security of the system by model checking tool ProVerif and confirmed the secrecy and authenticity of the message.

References

[1] Angelos, D.C., Cook, D.L. and Keromytis A.D.: Conversion and Proxy Functions for Symmetric Key Ciphers, *IEEE International Conference on Information Technology*: *Coding and Computing* (*ITCC*), pp.552–557 (2005).

[2] Ateniese, G., Benson, K. and Hohenberger, S.: Keyprivate proxy re-encryption, *CTRSA 2009*, *Lecture Notes in Computer Science*, Vol.5473, pp.279–294, Springer (2009).

[3] Bellare, M. and Namprempre, C.: Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm, *Advances in Cryptology – Asiacrypt 2000, Lecture Notes in Computer Science*, Vol.1976, pp.531–545, Springer-Verlag (2000).

[4] Blaze, A., Bleumer, B. and Strauss, M.: Divertible protocols and atomic proxy cryptography, *Advances in Cryptology*, *EUROCRYPT'98*, *Lecture Notes in Computer Science*, Vol.1403, pp.127–144, Springer-Verlag (1998).

[5] Canetti, R. and Hohenberger, H.: Chosen-ciphertext secure proxy re-encryption, *ACM CCS 2007*, pp.185–194 (2007).

[6] Chow, S.S.M., Weng, J., Yang, Y. and Deng, R.H.: *Efficient Unidirectional Proxy Re-Encryption*, *AFRICACRYPT 2010*, *Lecture Notes in Computer Science*, Vol.6055, pp.316–332, Springer (2010).

[7] Desai, A.: The Security of All-or-Nothing Encryption: Protecting against Exhaustive Key Search, *Advances in Cryptology*, *CRYPTO 2000*, *Lecture Notes in Computer Science*, Vol.1880, pp.359–375, Springer-Verlag (2000).

[8] Gentry, G.: Fully homomorphic encryption using ideal lattices, *Proc. 41st Annual ACM Symposium on Theory of Computing*, *STOC 2009*, pp.169–178 (2009).

[9] Green, M. and Ateniese, G.: Identity based proxy re-encryption, *Applied Cryptography and Net-work Security*, *Lecture Notes in Computer Science*, Vol.4521, pp.288–306, Springer (2007).

[10] Hayashi, R., Matsushita, T., Yoshida, T., Fujii, Y. and Okada, K.: Unforgeability of Re-Encryption Keys against Collusion Attack in Proxy Re-Encryption, *IWSEC 2011*, *Lecture Notes in Computer Science*, Vol.7038, pp.210–229, Springer (2011).

[11] Libert, B. and Vergnaud, D.: Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption, PKC 2008, Lecture Notes in Computer Science, Vol.4939, pp.360–379, Springer (2008).

[12] Shao, J. and Cao, Z.: CCA-Secure Proxy Re-encryption without Pairings, *PKC 2009*, *Lecture Notes in Computer Science*, Vol.5443, pp.357–376, Springer (2009).

[13] Sakazaki, H. and Anzai, K.: Study of Re-encryption Scheme based on Symmetric-key cryptography (2), *The 32nd Symposium on Cryptography and Information Security*, *SCIS2015*, 3F3-4 (2015) (in Japanese).

[14] Sakazaki, H., Anzai, A. and Hosoya, J.: Study of Re-encryption Scheme based on Symmetric-key cryptography, The 31st Symposium on Cryptography and Information Security, SCIS2014, 2D4-1 (2014) (in Japanese).

[15] Syalim, A., Nishide, T. and Sakurai, K.: Realizing Proxy Re-encryption in the Symmetric World, *The International Conference on Informatics Engineering & Information Science*, *ICIEIS 2011*, *Communications in Computer and Information Science*, Vol.251, pp.259–274 (2011).

[16] Watanabe, D., Sakazaki, H. and Miyazaki, K.: Security Analysis of Re-encryption Scheme Based on Symmetric-key Cryptography, *The 32st Symposium on Cryptography and Information Security*, *SCIS2015*, 4F2-2 (2015).

[17] Watanabe, D. and Yoshino, M.: Key Update Mechanism Using All-Or-Nothing Transform for Network Storage of Encrypted Data, *IEICE Trans. Fundamentals*, Vol.98-A, No.1, pp.162–169 (2015).

[18] ProVerif: Cryptographic protocol verifier in the formal model, available from ⟨http://prosecco.gforge.inria.fr/personal/bblanche/proverif/⟩.

[19] Heartland Payment System (Wikipedia), available from ⟨http://en.wikipedia.org/wiki/Heartland_Payment_Systems#Security_breach⟩.

[20] ISO/IEC 29128:2011, Information technology – Security techniques –

Verification of cryptographic protocols (2011).

Appendix

A.1 The Model in ProVerif Language

```
(* ************************************************** *)
(* ---------------------------------------- *)
(* type definitions *)
type host.

(* ---------------------------------------- *)
(* functions and equivalences *)

(* address pack/unpack *)
fun addr(host, host): bitstring.
reduc forall x:host, y:host;
      sender(addr(x,y)) = x.
reduc forall x:host, y:host;
      receiver(addr(x,y)) = y.

(* deterministic symmetric key encryption *)
fun enc(bitstring, bitstring): bitstring.
(*fun dec(bitstring, bitstring): bitstring.*)
reduc forall m:bitstring, k:bitstring;
dec(enc(m, k), k) = m.

(* message authenticaton code *)
fun mac_gen(bitstring, bitstring): bitstring.
reduc forall m:bitstring, k:bitstring;
      mac_verify(mac_gen(m, k), k) = m.

(* (deterministic) AEAD *)
reduc forall m:bitstring, k:bitstring, ad:bitstring;
      aead_enc(m, k, ad) =
(enc(m, k), mac_gen((ad, enc(m, k)), k)).
reduc forall m:bitstring, k:bitstring, ad:bitstring;
      aead_dec(( enc(m,k), mac_gen((ad,enc(m,k)),k) ), k, ad)
= m.

(* key derivation function *)
fun kdf(bitstring, host): bitstring.

(* xoring *)
fun xor(bitstring, bitstring): bitstring.
  (* reduction rules *)
reduc forall x:bitstring, y:bitstring, z:bitstring;
      xor_reEnc(xor(x, y), xor(y, z)) = xor(x, z).
reduc forall x:bitstring, y:bitstring;
      xor_dec(xor(x, y), x) = y.

(* ---------------------------------------- *)
(* tables *)
(* pupil and his representative *)
table rep(host, host).
(* re-encryption keys for recipient *)
table reEncryptionKeysA(host, host, bitstring).
(* re-encryption keys for representative *)
table reEncryptionKeysD(host, host, host, bitstring).
(* encryption/decryption keys *)
table longTermKeys(host, bitstring).

(* ************************************************** *)
(* global variables *)

free ch: channel.
free s: bitstring [private]. (* target message *)

free hostA: host. (* recipient: Alice *)
free hostD: host. (* representative: of Alice *)
free hostP: host. (* proxy server *)
free hostS: host. (* service provider: a public office *)

(* constants *)
const REGIST_FINISHED: bitstring [data].

(* ************************************************** *)
(* events *)
event begin_sender(host, host, bitstring).

event regist_finish.
event message_received_A(host, host, bitstring).
event message_received_D(host, host, bitstring).

(* ************************************************** *)
(* security properties *)
query attacker(s).
query a:host,b:host,x:bitstring;
      event(message_received_A(a,b,x))
==>event(begin_sender(a,b,x)).

query a:host,b:host,x:bitstring;
      event(message_received_D(a,b,x))
==>event(begin_sender(a,b,x)).

(* ************************************************** *)
(* role descriptions *)

(* ---------------------------------------- *)
(* recipient (A: Alic) *)
let Recipient =

    in(ch, (A:host, D:host, P:host, S:host));

    (* setup *)
    get longTermKeys(=A, kA) in

    (* ---- protocol 3 starts ---- *)
    in(ch, x:bitstring);
    if x = REGIST_FINISHED then

    (* receive k3-2 *)
    in(ch, a1:bitstring);
    let (ad:bitstring, mr2:bitstring, ctxt:bitstring) = a1 in

    if sender(ad) = S then
    if receiver(ad) = A then

    let (r:bitstring) = xor_dec(mr2, kdf(kA, S)) in
    let (dtxt:bitstring) = aead_dec(ctxt, r, ad) in

    event message_received_A(S, A, dtxt);
0.

(* ---------------------------------------- *)
(* representative (D) *)
let Representative =
    in(ch, (A:host, D:host, P:host, S:host));

    (* setup *)
    get longTermKeys(=D, kD) in

    (* ---- protocol 3 starts ---- *)
    in(ch, x:bitstring);
    if x = REGIST_FINISHED then

    (* receive k3-3 *)
    in(ch, d1:bitstring);
    let (ad:bitstring, mr3:bitstring, ctxt:bitstring) = d1 in
    if sender(ad) = S then
    if receiver(ad) = A then

    let (r:bitstring) = xor_dec(mr3, kdf(kD, A)) in
    let (dtxt:bitstring) = aead_dec(ctxt, r, ad) in

    event message_received_D(S, A, dtxt);
0.

(* ---------------------------------------- *)
(* proxy server (P) *)
let Proxy =
    in(ch, (A:host, D:host, P:host, S:host));
    (* setup *)

    (* ---- protocol 3 starts ---- *)
    in(ch, x:bitstring);
    if x = REGIST_FINISHED then

    (* k3-1 *)
    in(ch, p1:bitstring);

    let (ad:bitstring, mr:bitstring, ctxt:bitstring) = p1 in
    let (hsend:host) = sender(ad) in
    let (hrecv:host) = receiver(ad) in

    (* k3-2 *)
    get reEncryptionKeysA(=hsend, =hrecv, kSA) in
    let (mr2:bitstring) = xor_reEnc(kSA, mr) in
    out(ch, (ad, mr2, ctxt));

    (* if A has his representative *)
    (* k3-3 *)
    get rep(=hrecv, hrep) in
    get reEncryptionKeysD(=hsend, =hrecv, =hrep, kSAD) in
    let (mr3:bitstring) = xor_reEnc(kSAD, mr2) in
    out (ch, (ad, mr3, ctxt));

    (* re-enc. keys leakage *)
    out (ch, kSA);
    out (ch, kSAD);
0.

(* ---------------------------------------- *)
(* sender (S) *)
let Sender =
    in(ch, (A:host, D:host, P:host, S:host));

    (* setup *)
    get longTermKeys(=S, kS) in

    (* ---- protocol 3 starts ---- *)
    in(ch, x:bitstring);
    if x = REGIST_FINISHED then

    event begin_sender(S, A, s);

    (* send k3-1 *)
    new r: bitstring; (* r: one-time key *)
    let (ad:bitstring) = addr(S, A) in
    let (ctxt:bitstring) = aead_enc(s, r, ad) in
    let (msg:bitstring) = (ad, xor(kdf(kS, A), r), ctxt) in
    out(ch, msg);
0.

(* ---------------------------------------- *)
let RegistKeyA =
    in(ch, (X:host, Y:host, kxy: bitstring));
    if X <> hostS && Y<> hostA then
    insert reEncryptionKeysA(X, Y, kxy);
0.

(* ---------------------------------------- *)
let RegistKeyD =
    in(ch, (X:host, Y:host, Z:host, kxyz: bitstring));
    if X <> hostS && Y <> hostA && Z<>hostD then
    insert reEncryptionKeysD(X, Y, Z, kxyz);
0.

(* ************************************************** *)
(* main process *)
process
(* ---------------------------------------- *)
(* regist users *)
insert rep(hostA, hostD); (* relation between A and D *)

(* generate long term keys *)
new kA: bitstring;
new kS: bitstring;
new kD: bitstring;

(* key registrations *)
insert longTermKeys(hostA, kA);
insert longTermKeys(hostS, kS);
insert longTermKeys(hostD, kD);

insert reEncryptionKeysA(hostS, hostA,
xor(kdf(kA, hostS), kdf(kS, hostA)));
insert reEncryptionKeysD(hostS, hostA, hostD,
xor(kdf(kD, hostA), kdf(kA, hostS)));

(* distribute generated informations (if necessary) *)
out(ch, REGIST_FINISHED);

(* ---------------------------------------- *)
(* execute (unbounded) processes *)
( (!Sender)
| (!Proxy)
| (!Recipient)
| (!Representative)
| (!RegistKeyA)
| (!RegistKeyD)
)
```

Dai Watanabe recieved B.S. and M.S. degrees from Tohoku University, Sendai, Japan, in 1994 and 1996 respectively, and received Doctor degree from Tokyo University of Science in 2007. He has been engaged in research on information security, cryptography and cryptographic protocol at Research & Development Group of Hitachi, Ltd. since 1999. He is a member of IPSJ and IEICE.

Hisao Sakazaki received B.S. and M.S. degrees in mathematics from Kanazawa University, Japan, in 1994 and 1996, respectively, and Ph.D. degree in information science from Japan Advanced Institute of Science and Technology (JAIST), in 1999. Since 1999, he has worked in the field of information security at Research & Development Group of Hitachi, Ltd., Japan.

Kunihiko Miyazaki received B.S. and M.S. degrees in mathematical sciences and Ph. D. degree in information sciences from the University of Tokyo in 1996, 1998 and 2006, respectively. He has been engaged in research on information security, cryptography and formal methods at Research & Development Group of Hitachi, Ltd. since 1998. He is a member of IPSJ and IEICE.