All-or-Nothing Transform and Remotely Keyed Encryption Protocols

Sang Uk Shin, Weon Shin, and Kyung Hyune Rhee

Department of Computer Science, PuKyong National University,
599-1, Daeyeon-dong, Nam-gu, Pusan, 608-737, Korea
{shnsu,redcomet,khrhee}@woongbi.pknu.ac.kr
http://unicorn.pknu.ac.kr/~{soshin, redcomet and khrhee}

Abstract. We propose two new Remotely Keyed Encryption protocols; a length-increasing RKE and a length-preserving RKE. The proposed protocols have smaller computations than the existing schemes and provide sufficient security. In proposed protocols, we use the cryptographic transform with all-or-nothing properties proposed by Rivest as encryption operation in the host. By using all-or-nothing transform(AONT), RKE can be more dependent on the entire input and all encryption operation can be implemented only by using hash functions in the host. Moreover, we provide a length-preserving RKE protocol requires only one interaction between the host and the card.

1 Introduction

No cryptographic protocol is stronger than the mechanism protecting its secret keys. However, many computer and communication systems do not have a safe place which can store secret keys and perform the cryptographic computations. Especially,computers connected to the network such as Internet may be partly controlled by an adversary. Therefore, it is nature to consider adding an external, special-purpose device such as a smart card. Since such devices have one purpose and communicate via a limited set of functions, they can be made much more secure than general-purpose host machines. However, it is not practical to rely on such devices to perform all sensitive cryptographic operations. Such devices have limited bandwidth, memory, and processor speed.

A Remotely Keyed Encryption(RKE) focuses the fact that many high-band width applications need symmetric-key encryption schemes that store long -lived keys in low-bandwidth smart-cards. A RKE can do bulk encryption/decryption for high-bandwidth applications in a way that takes advantage of both the superior power of the host and the superior security of the smart-card. A RKE consists of two protocols, one for encryption and one for decryption. Given an l-bit input, either to encrypt or to decrypt, a protocol runs; the host sends a challenge value to the card depending on the input, and the card replies a response value depending on the challenge value and the key. During the run of a protocol, every challenge value may depend on the input and the previous response values, and the response values may depend on the key and the previous challenge values.

H. Imai, Y. Zheng (Eds.): PKC 2000, LNCS 1751, pp. 178–195, 2000.

In this paper, we propose two new remotely keyed encryption protocols; a length-increasing RKE and a length-preserving RKE. The proposed protocols have smaller computations than the existing schemes and provide sufficient security. And in proposed protocols, we use the cryptographic transform with all-or-nothing properties proposed by Rivest[10] as encryption operation in the host. As All-Or-Nothing Transform(AONT) with all-or-nothing properties, we use an improved version of the AON hashing-3 proposed by Shin, et al[11]. The improved AON hashing-3(IAON hashing) provides better security without additional computation overhead. By using the AONT, RKE can be more dependent on the entire input and all encryption operation can be implemented only by using hash functions in the host. Also, we suggest a solution for an open problem suggested by Blaze, Feigenbaum and Naor[4][5]:

> *Is there a secure, length-preserving RKE that requires only one round of interaction?*

The proposed length-preserving RKE requires only one interaction between the host and the card.

The remainder of this paper is organized as follows. In section 2, we describe AONT and propose an improved AON hashing. In section 3, we describe the definition and security requirements of RKE. And in section 4, we propose two new secure RKE protocols, a secure length-increasing RKE and a secure length-preserving RKE. Finally, we have conclusions in section 5.

2 All-or-Nothing Transform

In 1997, Rivest proposed an all-or-nothing encryption, a new encryption mode for block ciphers[10]. This mode has the property that one must decrypt the entire ciphertext before one can determine even one message block. This means that brute-force searches against all-or-nothing encryption are slowed down by a factor equal to the number of blocks in the ciphertext.

Rivest proposed the all-or-nothing transform which is referred to "package transform", as follows[10]:

(1) Let the input message be $m_1, m_2, \ldots, m_s$.
(2) Choose at random a key K for the package transform block cipher.
(3) Compute the output sequence $m'_1, m'_2, \ldots, m'_{s'}$ for $s' = s + 1$ as follows:
- $m'_i = m_i \oplus E(K, i)$ for $i = 1, 2, 3, \ldots, s$.
- Let

$$m'_{s'} = K \oplus h_1 \oplus \ldots \oplus h_s,$$

where

$$h_i = E(K_0, m'_i \oplus i) \ \ for \ \ i = 1, 2, \ldots, s,$$

where K_0 is a fixed, publicly-known encryption key.

The block cipher for the package transform does not use a secret key, and needs not be the same as the block cipher for encrypting the pseudo-message. We assume that the key space for the package transform block cipher is sufficiently large that brute-force searching for a key is infeasible. It is easy to see that the package transform is invertible:

$$K = m'_{s'} \oplus h_1 \oplus \ldots \oplus h_s,$$

$$m_i = m'_i \oplus E(K, i) \ \ for \ \ i = 1, 2, \ldots, s.$$

If any block of pseudo-message sequence is unknown, the K can not be computed, and so it is infeasible to compute any message block.

An all-or-nothing transform is merely a pre-processing step, and so it can be used with already-existing encryption software and device, without changing the encryption algorithm. The legitimate communicants pay a penalty of approximately a factor of three in the time it takes them to encrypt or decrypt in all-or-nothing mode, compared to an ordinary separable encryption mode. However, an adversary attempting a brute-force attack pays a penalty of a factor of t, where t is the number of blocks in the ciphertext.

As an application of AONT, Shin, et al proposed hash functions and its application to the MAC(Message Authentication Code) with all-or-nothing properties in 1999[11]. The proposed AON hashings used the existing hash functions without changing their algorithm and were secure against the existing known attacks. Also they can be easily applied to the MAC which can provide both the authentication and the confidentiality for message by using only hash functions. Now we describe shortly the proposed AON hashing-3 which operates as following. For details of the algorithm, see [11]:

A. Sender

(1) Splitting an input message X into s blocks of n-bit each : $Y = (X_1, X_2, \ldots, X_s)$.
(2) Generating a random key K of n-bit.
(3) Compute a pseudo-message Y by all-or-nothing transform.

$$Y_0 = IV \ , \ X_0 = 0 \ ,$$

$$Y_i = X_i \oplus h_{Y_{i-1}}(K \oplus (\overline{X_{i-1}||i})) \ , \ \ i = 1, 2, \ldots, s.$$

(4) Compute the last pseudo-message block, Y_{s+1} (n-bit length).

$$MD = h_{K_p}(Y_1|| \ldots ||Y_s||h_{IV}(K_p)) \ ,$$

$$Y_{s+1} = K \oplus \{\overline{MD}\}.$$

(5) Send $(Y||h_{MD}(Y_{s+1}))$.

B. Receiver

(1) Receive $(Y||Z)$.

(2) Splitting the pseudo-message Y into $s+1$ blocks of n-bit each, $Y_1, Y_2, \ldots, Y_s$ and Y_{s+1}.
(3) Recover the random key K.

$$MD' = h_{K_p}(Y_1||\ldots||Y_s||h_{IV}(K_p)) ,$$
$$K = Y_{s+1} \oplus \{\overline{MD'}\}.$$

(4) Check if $Z = h_{MD'}(Y_{s+1})$.
(5) Recover the original message.

$$Y_0 = IV \ , \ X_0 = 0 \, ,$$
$$X_i = Y_i \oplus h_{Y_{i-1}}(K \oplus (\overline{X_{i-1}||i})) \ , \ i = 1, 2, \ldots, s.$$

We propose an improved AON hashing(IAON hashing) which provides better security without additional computation overhead. IAON hashing operates as following:

A. Sender

(1) Compute $K = h_{IV}(X)$.
(2) Splitting an input message X into s blocks of n-bit each : $Y = (X_1, X_2, \ldots, X_s)$.
(3) Compute a pseudo-message Y by all-or-nothing transform.

$$Y_0 = IV \ , \ X_0 = K \, ,$$
$$Y_i = X_i \oplus h_{X_{i-1}}(Y_{i-1}||(K \oplus i)) \ , \ i = 1, 2, \ldots, s.$$

(4) Compute the last pseudo-message block, Y_{s+1} (n-bit length).

$$MD = h_{K_p}(Y_1||\ldots||Y_s||h_{IV}(K_p \oplus (s+1))) ,$$
$$Y_{s+1} = K \oplus MD.$$

(5) Send $(Y||h_{MD}(Y_{s+1}))$.

B. Receiver

(1) Receive $(Y||Z)$.
(2) Splitting the pseudo-message Y into $s+1$ blocks of n-bit each, $Y_1, Y_2, \ldots, Y_{s+1}$.
(3) Recover K.

$$MD' = h_{K_p}(Y_1||\ldots||Y_s||h_{IV}(K_p \oplus (s+1))) ,$$
$$K = Y_{s+1} \oplus MD'.$$

(4) Check if $Z = h_{MD'}(Y_{s+1})$.
(5) Recover the original message.

$$Y_0 = IV \ , \ X_0 = K \, ,$$
$$X_i = Y_i \oplus h_{X_{i-1}}(Y_{i-1}||(K \oplus i)) \ , \ i = 1, 2, \ldots, s.$$

(6) Check if $K = h_{IV}(X_1 \dots X_s)$.

In IAON hashing, each pseudo-message block depends on the entire original message since K is a hash value of the entire original message, and it is generated as k-bit random number in AON hashing-3. Thus, IAON hashing does not require an additional computation.

To find a collision pair, in AON hashing-3, an adversary has to find the collision pair of the pseudo-message and then search the random key K mapping the input message pair to collision pseudo-message. However, IAON hashing imposes stronger restriction to find a collision pair since K is a hash value of the original input message. That is, an attacker has to find a collision pair of the pseudo-message and then check whether the modified input message is mapped to the collision pair of the pseudo-message using the hash value of the modified input message. At the worst case, AON hashing-3 requires $2^{n/2+k}$ operations but IAON hashing requires $2^{n/2+\beta}$ operations to find a collision pair(β is a length of the input message).

IAON hashing can be easily applied to the MAC (Message Authentication Code). In this case, it is possible to provide both the authentication and confidentiality for message. For this MAC application, two communication parties between sender and receiver have to securely keep publicly-known random constant K_p. This MAC construction may be considered as the variant of HMAC proposed by Bellare, et al[1] such like that

$$HMAC_k(x) = h(\overline{k} \oplus opad, h(\overline{k} \oplus ipad, x))$$

where k is a secret information which held by two parties.
Thus AON-MAC has the same security as that of HMAC. Furthermore, an attacker who does not know a random constant K_p cannot find the K that is needed to recover the entire original message. To find a K, an attacker has to find the K_p such that

$$MD = h_{K_p}(Y_1 || \dots || Y_s || h_{IV}(K_p \oplus (s+1))) \,,$$

The best known attack for this scheme is the divide-and-conquer attack proposed by Preneel and van Oorschot[9]. Also, to decrypt only one block of message, an attacker who does not know the K has to solve the following:

$$X_i = Y_i \oplus h_{X_{i-1}}(Y_{i-1} || (K \oplus i))$$

To solve it, he must guess $h_{X_{i-1}}(Y_{i-1} || (K \oplus i))$ or find K and X_{i-1}.

We can improve the security by adding some overhead to the above schemes. We can encrypt the last pseudo-message block and message digest pair (Y_{s+1}, $h_{MD}(Y_{s+1})$) using the block cipher with the secret key K_S, and send it to the recipient. This scheme can provide the confidentiality of message by encrypting only some blocks without encrypting the entire message. It is specially useful if the cipher is a public key cryptosystem, such as RSA or ElGamal. In this scheme, we can use RSA to securely encrypt message longer than the key size without

need for a symmetric cipher. An additional encryption overhead is $2n$ bits. If we use SHA-1 as a building block, the length of the encrypting block becomes a size of total of 360 bits. Therefore, the performance degradation can be negligible in this case. However an attacker must find the K_S and K_p for decrypting the entire message.

IAON-MAC is similar to the concept of "Scramble All, Encrypt Small" proposed by Jakobsson, et al[6].

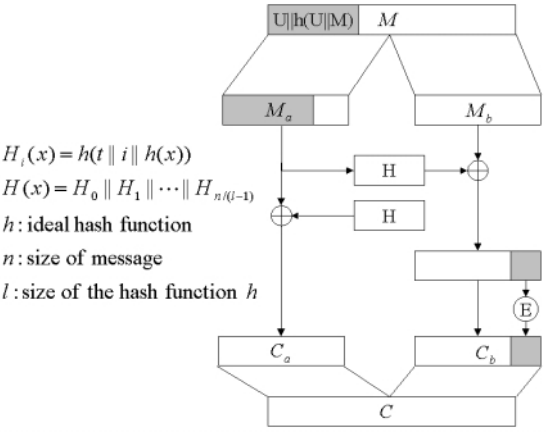

Fig. 1. Scramble All, Encrypt Small

In IAON-MAC, the process that transforms the original message into the pseudo-message is considered as the scrambling process in [6]. We compare both schemes in view point of computational amounts. Assuming a 160,000 bits message is processed, the Jakobsson's scheme requires 1001 computations of a hash function. That is, about 313,313 computations of a compression function of a hash function. However, IAON-MAC requires about 1,627 computations of a compression function. Thus, IAON-MAC has less computations than the Jakobsson's scheme.

3 Remotely Keyed Encryption Protocols

A remotely keyed encryption was first proposed by Blaze[3]. Blaze focuses the fact that many high-bandwidth applications need symmetric-key encryption schemes that store long-lived keys in low-bandwidth smart-cards. A RKE can do bulk encryption/decryption for high-bandwidth applications in a way that takes advantage of both the superior power of the host and the superior security of the smart-card. A RKE consists of two protocols, one for encryption and one for decryption. Given an l-bit input, either to encrypt or to decrypt, a protocol runs: the host sends a challenge value to the card depending on the input, and

the card replies a response value depending on the challenge value and the key. During the run of a protocol, every challenge value may depend on the input and the previous response values, and the response values may depend on the key and the previous challenge values.

3.1 Related Work

Blaze was the first to use the term "remotely keyed encryption" and proposed a specific scheme[3]. It is based on the idea of letting the host send the card one block which depends on the whole message. This block would be encrypted with the smart card secret key, and would also serve as a seed for the creation of a temporary secret key that will be used by the host in order to encrypt the rest of the message. However, in 1997 Lucks pointed that Blaze's scheme had problems in that allowed an adversary to forge a new valid plaintext/ciphertext pair after several interactions with the card[7]. Lucks suggested an alternative method[7] but was attacked by Blaze, Feigenbaum and Naor who further demonstrated the subtleties of this problem[4][5]. They showed that the encryption key used for the largest part of the message is deterministically derived from the two first blocks of the message, hence an adversary who takes control of the host will be able to decrypt a large set of messages with only one query. They derived a careful formal model and a scheme based on pseudorandom functions. In 1999, Lucks further extended their security model and proposed a faster one[8]. And Jakobsson, et al proposed "Scramble All, Encrypt Small" notion[6]. It scrambles the message in a publicly available method and then deprives an adversary of the ability to invert the scrambling by letting the smart card encrypt just a small part of the scrambled message.

3.2 Security Requirements

Lucks proposed that a RKE should have three properties[8]:

(i) Forgery security : If the adversary has controlled the host for m interactions, then it cannot produce $m + 1$ plaintext/ciphertext pairs.
(ii) Inversion security : Access to encryption should not imply the ability to decrypt and vice verse.
(iii) Pseudorandomness : The encryption function should behave randomly, for someone neither having access to the card, not knowing the secret key.

Requirements (i) and (ii) restrict the abilities of an adversary with access to the smart card. Requirement (iii) is only valid for outside adversaries, having no access to the card. If an adversary could compute forgeries or run inversion attacks, she could easily distinguish the encryption function from a random one.

Recently, Blaze, Feigenbaum and Naor found a better formalism to define the pseudorandomness of RKEs[4][5]. Their idea is based on the adversary gaining direct access to the card for a certain amount of time, making m interaction with the card. For the adversary having lost direct access to the card, the encryption function should behave randomly. An attacker is divided into two phases:

(i) The *host phase* : During the host phase, an attacker A may have a total of m interactions with the card. He may send any message to the card during one of these interaction and may deviate from the protocol. However, since m is an upper bound on the total number of interactions, A obtains at most m plaintexts and m ciphertexts during the host phase. After these m interactions with the card, A loses control of the host.

(ii) The *distinguishing phase* : An attacker chooses plaintexts or ciphertexts and asks for their encryptions or decryptions. The answer to these queries are either chosen randomly, or by honestly encrypting or decrypting according to the RKE. An adversary's task is to distinguish between the random case and honest encryption.

If, during the distinguishing phase, an adversary A queries the oracle about any of the m plaintexts and m ciphertexts that he obtained during the host phase, her task would be quite easy. Thus, in the distinguishing phase, it needs to filter texts that appeared in the host phase before. To do this, BFN introduced an "*arbiter*" algorithm B. The purpose of the arbiter B is to make sure that A does not ask during the distinguishing phase any of the queries that it asked during the host phase.

According to Lucks and BFN, the formal definitions of a RKE can be given as follows[4][5][8].

Definition 1 (length-preserving RKE). *A* length-preserving RKE *is a pair of protocols, one for encryption and one for decryption, to be executed by a host and a card. The length of a ciphertext must be the same as that of the corresponding plaintext. Let B be an algorithm, the "arbiter algorithm", which is initialized with a transcript of the communication between the host and the card during the host phase. During the host phase, an attacker A may play the role of the host in a total of q_h interactions with the card. During this phase, A may send any message to the card and does not necessarily follow the encryption or decryption protocol. During the distinguishing phase, A choose up to q_d texts T as queries and asks for the corresponding encryptions or decryptions. W.l.o.g., we prohibit A to ask equivalent queries, i.e., to ask twice for the encryption of T, to ask twice for the decryption of T, or ask once for the encryption of a T and some time before or after this for the decryption of the corresponding ciphertext. Before the distinguishing phase starts, a switch S is randomly set either to 0 or to 1. If the arbiter B acts, A's query s answered according to the RKE. If B does not act on, the answers are generated depending on S. Consider A asking for the encryption or decryption of a text $T \in \{0,1\}^\beta$ with $\beta > b$. If $S = 0$, the response is evaluated according to the RKE. If $S = 1$, the response is a random value in $\{0,1\}^\beta$. After the distinguishing phase, A's task is to guess S. The difference between the probability that A accepts on a continuation of the RKE and the probability that A accepts on a switch to a random function pair must be negligible.*

Length-increasing RKEs would be easier to construct than length-preserving RKEs. However, we can require additional security properties of length-in-

creasing RKEs that are not achievable in the length-preserving case. In the length-increasing case, each plaintext may correspond to multiple ciphertexts because the ciphertext space is bigger than the plaintext space. For a length-increasing RKE, BFN introduced the concept of a "random, self-validating black box"[5]. Given that an arbiter is filtering based on a transcript of the host phase communication, after the host phase, the adversary cannot tell whether he is interacting with real protocols or with a "random, self-validating black box." A "random, self-validating black box" contains an encryption box and a decryption box. On any input of the appropriate plaintext length, the encryption box outputs a random string of the appropriate ciphertext length. The decryption box outputs "invalid" on all inputs, except those that were previously output by the encryption box, and on those it outputs the input string on which the encryption box gave this output.

Definition 2 (length-increasing RKE). *A* length-increasing RKE *is a pair of protocols, one for encryption and one for decryption, to be executed by a host and a card. The length of a ciphertext is greater than the length of the corresponding plaintext. If its input is a ciphertext that has previously been output by the encryption protocol, the decryption protocol outputs the corresponding plaintext; otherwise, it may output "invalid". The RKE is secure if there is a polynomial-time* arbiter *B that can enforce the following restriction on any adversary A: During the host phase, an attacker A may play the role of the host in a total of q_h interactions with the card. During this phase, A may send any message to the card and does not necessarily follow the encryption or decryption protocol. Between the host phase and the distinguishing phase, a choice is made between continuing to use the RKE or switching to a random, self-validating black box. B gets as input the transcript of the host phase communication between the host and the card. During the distinguishing phase, A choose up to q_d texts T as queries and asks for the corresponding encryptions or decryptions. Before the distinguishing phase starts, a switch S is randomly set either to 0 or to 1. If the arbiter B acts, A's query s answered according to the RKE. If B does not act on, the answers are generated depending on S. Consider A asking for the encryption or decryption of a text $T \in \{0,1\}^{\beta}$ with $\beta > b$. If $S = 0$, the response is evaluated according to the RKE. If $S = 1$, the response is evaluated according to a random, self-validating black box. The difference between the probability that A accepts on a continuation of the RKE and the probability that A accepts on a switch to a random, self-validating black box must be negligible.*

4 New Secure Remotely Keyed Encryption Schemes

4.1 Building blocks

In this section, we describe the building blocks that we use for new secure remotely keyed encryption(SRKE) schemes. As will be proven below, our schemes are secure if their building blocks are secure. The following is a description of the building blocks.

- X and Y denote the plaintext and the ciphertext, respectively. Usually, both are given in blocks and hence are denoted $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_n)$ where each of X_i and Y_i is in $\{0,1\}^b$.
- E and D denote the encryption and decryption functions of a block cipher. $E_k(X_i)$ denotes the encryption of plaintext block X_i with encryption key k, i.e., $E_k, D_k : \{0,1\}^b \rightarrow \{0,1\}^b$ and $D_k(E_k(X_i)) = X_i$. The required security property is that it should be a strong pseudorandom permutation. For $k \neq k'$, the permutations E_k and $E_{k'}$ are independent.
- $F : \{0,1\}^b \rightarrow \{0,1\}^b$ is a pseudorandom function. It may or may not be identical to the encryption function E of the block cipher. In situation that never require the function to be inverted, we use F_s rather than E_s. Similarly, two random functions F_k and $F_{k'}$ depending on independently chosen keys k and k' are independent.
- G_K denotes the encryption of an n-block plaintext $(X_1, \ldots, X_n)$ using encryption key K. The corresponding decryption function is denoted $\hat{G}_K$. The security requirement for G_K is that if K is chosen uniformly at random, then $G_K(X_1, \ldots, X_n)$ is pseudorandom for any $(X_1, \ldots, X_n)$. A possible example of G_K is to apply a pseudorandom generator to K and Xor the resulting sequence with $(X_1, \ldots, X_n)$. Another example is to use E_K with some sort of chaining, e.g., CBC[2].
- $h : \{0,1\}^* \rightarrow \{0,1\}^b$ is a collision-resistant hash function.

For the analysis, we assume the used building blocks(such as block ciphers) to behave like their ideal counterparts(such as random permutations).

4.2 A Secure, Length-Increasing RKE

Using the above building blocks, we first propose a secure, length-increasing RKE(LI_SRKE) protocol. The private key stored in the smart card is denoted k_1, k_2 and k_3. The encryption protocol works as follows:
LI_SRKE encryption protocol : input $X = (X_1, \ldots, X_l)$; output $t, Y_0, Y = (Y_1, \ldots, Y_l)$

(1) Host : $md_i \leftarrow h(X_1, \ldots, X_l)$, $K_0 \leftarrow md_i$
(2) Host : $Y \leftarrow G_{K_0}(X)$
(3) Host : $md_o \leftarrow h(Y)$
(4) Host $\rightarrow$ Card : K_0, md_o
(5) Card : $Y_0 \leftarrow E_{k_1}(K_0 \oplus F_{k_2}(md_o))$
(6) Card : $t \leftarrow F_{k_2}(md_o) \oplus F_{k_3}(K_0)$
(7) Card $\rightarrow$ Host : Y_0, t

The decryption protocol works as follows:
LI_SRKE decryption protocol : input $t, Y_0, Y = (Y_1, \ldots, Y_l)$; output $X = (X_1, \ldots, X_l)$ or "invalid"

(1) Host : $md_o \leftarrow h(Y)$
(2) Host $\rightarrow$ Card : t, Y_0, md_o

(3) Card : $K_0 \leftarrow D_{k_1}(Y_0) \oplus F_{k_2}(md_o)$
(4) Card : if $t \neq F_{k_2}(md_o) \oplus F_{k_3}(K_0)$ then $K_0 \leftarrow$ "invalid"
(5) Card $\rightarrow$ Host : K_0
(6) Host : If K_0 = "invalid" then output "invalid"
else $X \leftarrow \hat{G}_{K_0}(Y)$ if K_0=$h(X)$ then output X

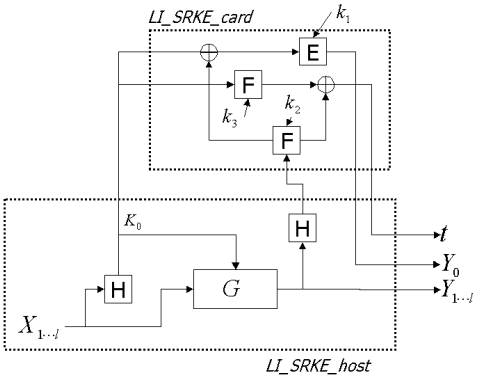

Fig. 2. LI_SRKE encryption protocol

As required by Definition 4, the arbiter B does not filter queries of $(X_1, \ldots, X_l)$ during the distinguishing phase. If no switch was made between phases it just sends them to the encryption protocol and to a random, self-validating black box if a switch was made. On the queries of $(Y_1, \ldots, Y_l)$, B computes $h(Y_1, \ldots, Y_l)$ and checks whether $(h(Y_1, \ldots, Y_l), Y_0, t)$ occurs in the transcript of the host phase. If it does, then B sends the query to the decryption protocol, regardless of whether a switch was made between phases. If it doesn't, then B sends it either to the decryption protocol or to the random, self-validating black box, depending on whether a switch was made.

We analyze the security of LI_SRKE based on the analysis of BFN's length-increasing RKE scheme[5] as following:

Theorem 1. *LI_SRKE is a secure, length-increasing RKE protocol.*

Proof. The definition of the cryptographic building blocks E, F and G imply that any sequence of encryptions is indistinguishable from a random one. Consider the case of decryption queries. Suppose that a query $(Y_1, \ldots, Y_l)$ does not corrspond to an encryption query $(X_1, \ldots, X_l)$ that occurred earlier in the distinguishing phase and $(md_o = h(Y_1, \ldots, Y_l), Y_0, t)$ did not appear in the host phase. A random, self-validating black box will answer such a query by saying "invalid." The real protocol will also answer "invalid" if $t \neq F_{k_2}(md_o) \oplus F_{k_3}(K_0)$ ($K_0 = D_{k_1}(Y_0) \oplus F_{k_2}(md_o)$), but it will produce a decryption if $t = F_{k_2}(md_o) \oplus F_{k_3}(K_0)$.

Thus an attacker can tell whether a switch was made between phases only if it can find $(t, Y_0, Y(= Y_1, \ldots, Y_l))$ such that

$$t = F_{k_2}(md_o) \oplus F_{k_3}(D_{k_1}(Y_0) \oplus F_{k_2}(md_o))$$

The collision resistance of h implies that the adversary cannot find $(Y_1, \ldots, Y_l) \neq (Y_1', \ldots, Y_l')$ such that $h(Y_1, \ldots, Y_l) = h(Y_1', \ldots, Y_l')$. Therefore, an attacker has to find collision pairs $(Y_0, h(Y_1, \ldots, Y_l)) \neq (Y_0', h(Y_1', \ldots, Y_l'))$ such that

$$\begin{aligned} &F_{k_2}(h(Y_1, \ldots, Y_l)) \oplus F_{k_3}(D_{k_1}(Y_0) \oplus F_{k_2}(h(Y_1, \ldots, Y_l))) \\ &= F_{k_2}(h(Y_1', \ldots, Y_l')) \oplus F_{k_3}(D_{k_1}(Y_0') \oplus F_{k_2}(h(Y_1', \ldots, Y_l'))) \end{aligned}$$

Suppose that F_{k_2} and F_{k_3} are truly random functions. Then all the values $F_{k_2}(h(Y_1, \ldots, Y_l))$ and $F_{k_3}(D_{k_1}(Y_0) \oplus F_{k_2}(h(Y_1, \ldots, Y_l)))$ are random values. Thus, the probability that $(Y_0, h(Y_1, \ldots, Y_l)) \neq (Y_0', h(Y_1', \ldots, Y_l'))$ but

$$\begin{aligned} &F_{k_2}(h(Y_1, \ldots, Y_l)) \oplus F_{k_3}(D_{k_1}(Y_0) \oplus F_{k_2}(h(Y_1, \ldots, Y_l))) \\ &= F_{k_2}(h(Y_1', \ldots, Y_l')) \oplus F_{k_3}(D_{k_1}(Y_0') \oplus F_{k_2}(h(Y_1', \ldots, Y_l'))) \end{aligned}$$

is $1/2^b$. There are $(q_h + q_d)^2$ possible pairs. Therefore, with the probability of $\frac{(q_h+q_d)^2}{2^b}$, the adversary can distinguish between a random, self-validating black box and the original encryption algorithm. ♠

The implementation of the function G

A function G which is used in the host of LI_SRKE algorithm has to satisfy the pseudorandomness. Possible realization of G are a pseudorandom generator and a block cipher with some sort of chaining. As an alternative scheme, we propose a IAON_G function which is a modified version of IAON hashing described in section 2. IAON_G works as follows:

IAON_G encryption : input $X = (X_1, \ldots, X_l)$; output $Y = (Y_1, \ldots, Y_l)$

$$\begin{aligned} &K_0 = h(X) \\ &X_0 = K_0, Y_0 = h(K_0) \\ &\text{for } \ i = 1 \ldots l \\ &\qquad K_i = h_{K_{i-1}}(Y_{i-1} || (K_{i-1} \oplus i)) \\ &\qquad H_1 = h_{X_{i-1}}(Y_{i-1} || (K_i \oplus i)) \\ &\qquad Y_i = X_i \oplus H_1 \oplus K_i \end{aligned}$$

IAON_G decryption : input $K_0, Y = (Y_1, \ldots, Y_l)$; output $X = (X_1, \ldots, X_l)$

$$\begin{aligned} &X_0 = K_0, Y_0 = h(K_0) \\ &\text{for } \ i = 1 \ldots l \\ &\qquad K_i = h_{K_{i-1}}(Y_{i-1} || (K_{i-1} \oplus i)) \\ &\qquad H_1 = h_{X_{i-1}}(Y_{i-1} || (K_i \oplus i)) \\ &\qquad X_i = Y_i \oplus H_1 \oplus K_i \end{aligned}$$

IAON_G is a modified version of AONT using hash functions, which transforms the original input message into the pseudo-message. An attacker who does

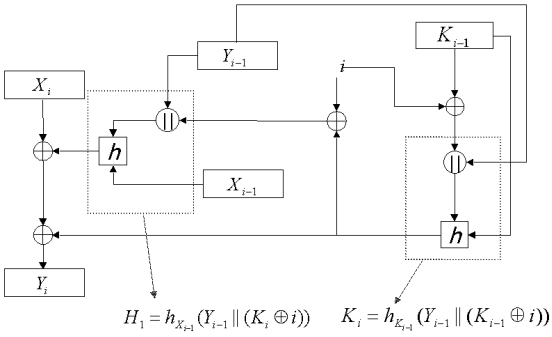

Fig. 3. IAON_G encryption process

not know K_0 cannot compute K_i and H_1. Thus he cannot recover X_i. This function is considered as a pseduorandom generator using hash functions. Since hash functions have pseudoranomness, K_0 is considered as pseudorandom value, and H_1 and K_i are also pseudorandom values, respectively. Therefore IAON_G has pseudorandomness stated by BFN. Also by using IAON_G, the output of the proposed RKE becomes more dependent on the entire input message and all encryption operation on the host can be implemented only by using the hash function.

We compare the proposed LI_SRKE with BFN's length-increasing RKE[5]. BFN's length-increasing RKE operates as Fig. 4.

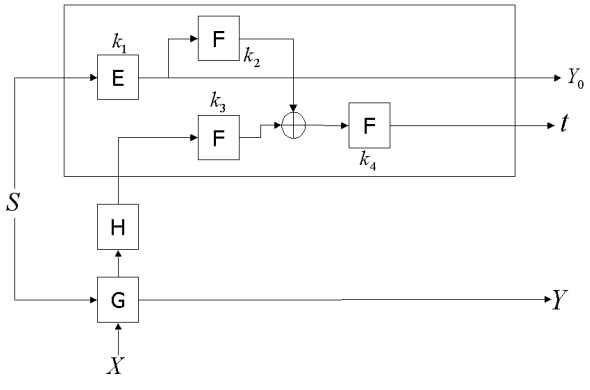

Fig. 4. BFN's length-increasing RKE

On the part of the host, the proposed scheme requires one more computation of a hash function but it is considered as the computation of a random number S

in BFN's scheme. The communication between the host and the card is exactly the same for both protocols. Inside the card, LI_SRKE needs three evaluations of cryptographic functions(such as E and F). In contrast to this, BFN's scheme needs four evaluations of cryptographic functions. In the case of BFN's scheme, to decrypt the ciphertext without interactions with the card, an attacker has to compute $S = D_{k_1}(Y_0)$. However, LI_SRKE requires the computation of $K_0 = D_{k_1}(Y_0) \oplus F_{k_2}(md_o)$. Moreover, since LI_SRKE uses IAON_G with all-or-nothing properties, LI_SRKE is more dependent on the input message and more secure against key search attacks such as brute-force attack.

4.3 A Secure, Length-Preserving RKE

In this section, we propose a secure, length-preserving RKE(LP_SRKE) protocol. The private key stored in the smart card is denoted k_1, k_2 and k_3. The encryption protocol works as follows:
LP_SRKE encryption protocol : input $X = (X_1, \ldots, X_l)$; output $Y = (Y_1, \ldots, Y_l)$

(1) Host : $K_0 \leftarrow X_1 \oplus h(X_2, \ldots, X_l)$
(2) Host : $(Y_2, \ldots, Y_l) \leftarrow G_{K_0}(X_2, \ldots, X_l)$
(3) Host : $t \leftarrow h(Y_2, \ldots, Y_l)$
(4) Host $\rightarrow$ Card : K_0, t
(5) Card : $Y_1 \leftarrow E_{k_1}(K_0 \oplus F_{k_3}(t)) \oplus F_{k_2}(t)$
(6) Card $\rightarrow$ Host : Y_1

The decryption protocol works as follows:
LP_SRKE decryption protocol: input $Y = (Y_1, \ldots, Y_l)$; output $X = (X_1, \ldots, X_l)$

(1) Host : $t \leftarrow h(Y_2, \ldots, Y_l)$
(2) Host $\rightarrow$ Card : Y_1, t
(3) Card : $K_0 \leftarrow D_{k_1}(Y_1 \oplus F_{k_2}(t)) \oplus F_{k_3}(t)$
(4) Card $\rightarrow$ Host : K_0
(5) Host : $(X_2, \ldots, X_l) \leftarrow \hat{G}_{K_0}(Y_2, \ldots, Y_l)$
(6) Host : $X_1 \leftarrow K_0 \oplus h(X_2, \ldots, X_l)$

We uses the IAON_G function as the LI_SRKE protocol. We analyze the security of LP_SRKE based on the security analysis of BFN's length-preserving RKE[4][5] and Lucks's ARKE(Accelerated RKE)[8]. By X^i, Y^i, K_0^i, t^i, we denote the challenge and response values of the i-th protocol execution. The protocol can either be executed in the host phase indicated by $i \in \{1, \ldots, q_h\}$, or in the distinguishing phase indicated by $i \in \{q_h + 1, \ldots, q\}$.

Theorem 2. *LP_SRKE is a secure, length-preserving RKE protocol.*

Proof. For the proof, we first define the arbiter algorithm B, and then we bound the advantage of the adversary. The arbiter algorithm B operates as follows:

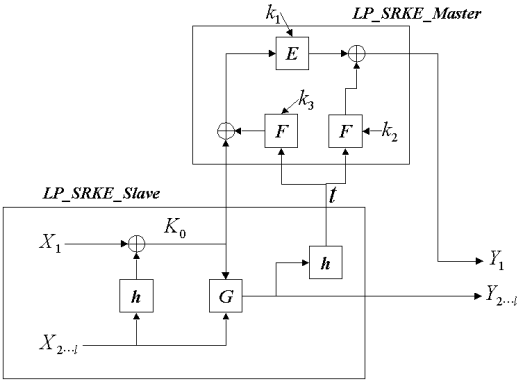

Fig. 5. LP_SRKE encryption protocol

The arbiter algorithm B

For $i \in \{1, \ldots, q_h\}$, B compiles a list L_1 of all the pairs (K_0^i, t^i) and another list L_2 of all (Y_1^i, t^j). These can be deduced from the transcript. If an attacker A asks for the encryption of a plaintext X^j with $j \in \{q_h + 1, \ldots, q\}$ in the distinguishing phase, the first challenge value for the card is the pair (K_0^j, t^j) where $K_0^j = X_1^j \oplus h(X_2^i, \ldots, X_l^i)$. If the pair (K_0^j, t^j) is contained in the list L_1, B acts on that query, and the answer is generated according to the encryption protocol. Similarly, if A asks for the decryption of a ciphertext Y^j in the distinguishing phase, and if the corresponding challenge value (Y_1^j, t^j) is contained in L_2, B acts and the answer is generated according to the decryption protocol. B does not act on more than q_h queries. Due to the collision resistance of h, A does not know more than one value $(X_2^i, \ldots, X_l^i)$ with $K_0^i = X_1^i \oplus h(X_2^i, \ldots, X_l^i)$. For the same reason, A know only one value $(Y_2^i, \ldots, Y_l^i)$ with $t^i = h(Y_2^i, \ldots, Y_l^i)$. Hence, for every $i \in \{1, \ldots, q_h\}$, A can ask no more than one plaintext X^j to be encrypted during the distinguishing phase, where the corresponding pair (K_0^j, t^j) would be found in the list L_1. Similarly, we argue for decryption queries. Asking for the encryption of $T = (X_1^j, X_2^j, \ldots, X_l^j)$ is equivalent to asking for the decryption of $T' = (Y_1^j, Y_2^j, \ldots, Y_l^j)$, and we only regard non-equivalent queries. ♣

Let $k > q_h$ and A be asking for the encryption of a plaintext $(X_1^k, X_2^k, \ldots, X_l^k)$. We assume $(K_0^k, t^k) \notin \{(K_0^1, t^1), \ldots, (K_0^k - 1, t^k - 1)\}$. If $j \in \{1, \ldots, q_h\}$ and $(K_0^j, t^j) = (K_0^k, t^k)$, then B acts and the answer to this query does not depend on the switch S. If $j \in \{q_h + 1, \ldots, k - 1\}$, then $(X_1^j, X_2^j, \ldots, X_l^j) \neq (X_1^k, X_2^k, \ldots, X_l^k)$, because the j-th and the k-th query are not equivalent. If $(X_1^j, X_2^j, \ldots, X_l^j) \neq (X_1^k, X_2^k, \ldots, X_l^k)$, then $(K_0^j, t^j) = (K_0^k, t^k)$ implies a collision $(X_2^j, \ldots, X_l^j) \neq (X_2^k, \ldots, X_l^k)$ for h.

Depending on previous protocol runs and responses, we define the set $U_k \subseteq \{0,1\}^b \times \{0,1\}^{\beta-b}$ of ciphertexts(β is a length of a plaintext):

$$U_k \ := \ \left(\{0,1\}^b - \{Y_1^1, \ldots, Y_1^{k-1}\}\right) \times \left(\{0,1\}^{\beta-b}\right)$$

For $S = 1$, the ciphertext $(Y_1^j, Y_2^j, \ldots, Y_l^j)$ is a uniformly distributed random value in $\{0,1\}^b \times \{0,1\}^{\beta-b}$. We define what it means for the k-th query to be Σ_k, $\Sigma_k \ :\Leftrightarrow Y_1^k \in \{Y_1^1, \ldots, Y_1^{k-1}\}$. Obviously, if not Σ_k, then $(Y_1^k, Y_2^k, \ldots, Y_l^k)$ is a uniformly distributed random value in U_k. Further:

$$\mathrm{P}\left[\, \Sigma_k \mid S = 1 \,\right] \ \leq \frac{k-1}{2^b}$$

Consider $S = 0$. Obviously, if not Σ_k, then $(Y_1^k, Y_2^k, \ldots, Y_l^k) \in U_k$. Also, if not Σ_k, then K_0^k is not in $\{K_0^1, \ldots, K_0^{k-1}\}$, and then $G_{K_0^k}(Y_2^k, \ldots, Y_l^k)$ is a uniformly distributed random value in $\{0,1\}^{\beta-b}$. Furthermore, if $t^j = t^k$, $Y_1^j \neq Y_1^k$ due to $K_0^j \neq K_0^k$.

If $K_0^j \neq K_0^k$, then $E_{k_1}(K_0^j)$ and $E_{k_1}(K_0^k)$ are two independent random values in $\{0,1\}^b$. If $(K_0^k, t^k) \notin \{(K_0^1, t^1), \ldots, (K_0^k - 1, t^k - 1)\}$, for every $j \in \{1, \ldots, k-1\}$, $K_0^j \neq K_0^k$ if $t^j = t^k$. Hence, $\mathrm{P}\left[\, Y_1^j = Y_1^k \mid K_0^j \neq K_0^k \,\right] \leq 2^{-b}$.

$$\mathrm{P}\left[\, \Sigma_k \mid S = 0 \,\right] \ \leq \frac{k-1}{2^b}$$

We write Σ^* if any query in k in the distinguishing phase is Σ_k.

$$\mathrm{P}\left[\, \Sigma^* \mid S = 1 \,\right] \ \leq \sum_{k=q_h+1}^{q} \mathrm{P}\left[\, \Sigma_k \mid S = 1 \,\right] \ \leq \frac{1}{2}\frac{q^2}{2^b}$$

$$\mathrm{P}\left[\, \Sigma^* \mid S = 0 \,\right] \ \leq \sum_{k=q_h+1}^{q} \mathrm{P}\left[\, \Sigma_k \mid S = 0 \,\right] \ \leq \frac{q^2}{2^b}$$

If A asks for the decryption of the ciphertext $(Y_1^k, Y_2^k, \ldots, Y_l^k)$ as the k-th query instead of asking for an encryption, the same argument applies. Therefore, the advantage of A is

$$\mathrm{Adv}_A \leq \frac{q^2}{2^b}$$

♠

We compare the proposed LP_SRKE protocol with BFN's length-preserving RKE and Lucks's ARKE. BFN's scheme and ARKE operate as Fig.6 and Fig.7, respectively. On the part of the host, the cryptographic calculations are exactly the same for three protocol. While BFN's scheme and ARKE require two interactions between the host and the card, LP_SRKE requires only one interaction. Inside the card, BFN's scheme requires six evaluations of cryptographic functions and ARKE requires four evaluations. However, LP_SRKE requires three evaluations of cryptographic functions. Therefore, the proposed LP_SRKE has smaller computations and requires only one interaction between the host and the card. This scheme can be a solution for the open problem suggested by BFN[4][5]:

Is there a secure, length-preserving RKE that requires only one round of interaction?

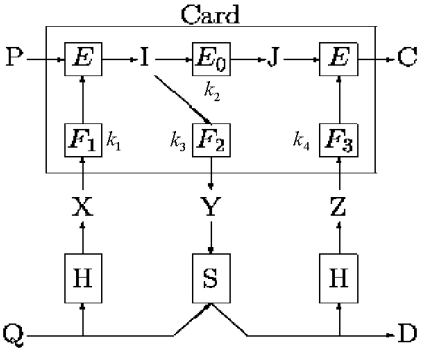

Fig. 6. BFN's length-preserving RKE protocol

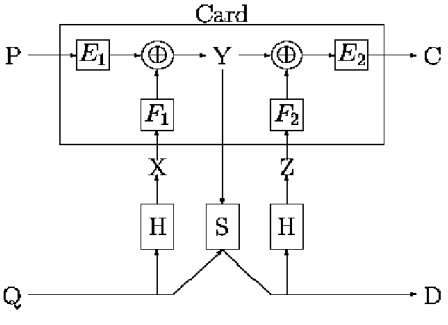

Fig. 7. ARKE protocol

5 Conclusions

In this paper, we proposed new secure remotely keyed encryption protocols; a secure, length-increasing RKE and a secure, length-preserving RKE. The proposed protocols have smaller computations than the existing RKE protocols and provide sufficient security. Especially, because the proposed length-preserving RKE requires only one interaction between the host and the card, we suggested the proposed length-preserving scheme as the solution for the open problem(*Is there*

a secure, length-preserving RKE that requires only one round of interaction?) suggested by BFN. Also, as a cryptographic function G in the host of the proposed protocols, we used the IAON_G with all-or-nothing properties proposed by Rivest. The IAON_G is an improved version of AON hashing-3 proposed by Shin, et al[11], and provides better security without additional computation overhead.

References

1. M. Bellare, R. Canetti, H. Krawczyk, "Keying Hash Functions for Message Authentication," in Advances in Cryptology - Crypto'96, LNCS, vol.1109, pp.1-15, 1996 182
2. M. Bellare, J. Kilian, R. Rogaway, "The Security of Cipher Block Chaining," in Advanced in Cryptography - Crypto'94, LNCS, vol.839, Springer, pp.341-358, 1994 187
3. M. Blaze, "High-Bandwidth Encryption with Low-Bandwidth Smartcards," in Proceedings of the Fast Software Encryption Workshop, LNCS, vol.1039, Springer, pp.33-40, 1996 183, 184
4. M. Blaze, J. Feigenbaum, M. Naor, "A Formal Treatment of Remotely Keyed Encryption," in Advanced in Cryptography - Eurocrypt'98, LNCS, vol.1403, pp.33-40, 1998 179, 184, 185, 191, 193
5. M. Blaze, J. Feigenbaum, M. Naor, "A Formal Treatment of Remotely Keyed Encryption," Full version of Eurocrypt'98, 1999 179, 184, 185, 186, 188, 190, 191, 193
6. M. Jakobsson, J.P. Stern, M. Yung, "Scramble All, Encrypt Samll," in Proceedings of the Fast Software Encryption Workshop, LNCS, vol.1636, 1999 183, 184
7. S. Lucks, "On the security of Remotely Keyed Encryption," in Proceedings of the Fast Software Encryption Workshop, LNCS, vol.1267, Springer, pp.219-229, 1997 184
8. S. Lucks, Accelerated Remotely Keyed Encryption," in Proceedings of the Fast Software Encryption Workshop, LNCS, vol.1636, 1999 184, 185, 191
9. B. Preneel, P. van Oorschot, "MDx-MAC and Building Fast MACs from Hash Functions," in Advances in Cryptology - Crypto'95, LNCS, vol.963, pp.1-14, 1995 182
10. R. L. Rivest, "All-Or-Nothing Encryption and The Package Transform," in Proceedings of the Fast Software Encryption Workshop, LNCS, vol.1267, pp.210-218, 1997 179
11. Sang Uk Shin, Kyung Hyune Rhee, Jae Woo Yoon, "Hash Functions and the MAC Using All-Or-Nothing Property," in Proceedings of PKC'99(International Workshop on Practice and Theory in Public Key Cryptography), LNCS, vol.1560, pp.263-275, 1999 179, 180, 195