JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 17, 945-957 (2001)

Short Paper

A New Design of the Hash Functions With All-or-Nothing Property*

SANG UK SHIN+ AND KYUNG HYUNE RHEE++
+*Electronics and Telecommunications Research Institute*
Yusong-Gu, Daejon, 305-350, Korea
E-mail: shinsu@etri.re.kr
++*Division of Electronics, Computer and Telecommunications Engineering*
PuKyong National University
Nam-Gu, Pusan, 608-737, Korea
E-mail: khrhee@pknu.ac.kr

All-or-nothing property is a new encryption mode proposed by Rivest and has the property that one must decrypt the entire ciphertext to determine any plaintext block. In this paper, we propose hash functions with all-or-nothing property. The proposed schemes use the existing hash functions without changing their structures, and they are secure against known attacks. Moreover, the proposed methods can be easily extended to the MAC (Message Authentication Code) for providing message confidentiality as well as authentication.

Keywords: cryptography, hash function, message authentication code, all-or-nothing transform, authentication

1. INTRODUCTION

Cryptographic hash functions play an important role in modern cryptography as a tool for providing message integrity and authentication. The basic idea of hash functions is that a hash value serves as a compressed representative image of an input string and can be used as if it is uniquely identifiable with that string. Hash functions are classified into two classes [1]: unkeyed hash function with single parameter - a message, and keyed hash function with two distinct inputs - a message and secret key. Keyed hash functions are used to construct the MAC. The MAC is widely used to provide data integrity and data origin authentication.

Rivest proposed the new encryption mode, referred to the "all-or-nothing encryption mode" [2]. This mode has the property that one must decrypt the entire ciphertext

Received January 4, 2001; accepted July 9, 2001.
Communicated by Chi Sung Laih.
* The early version of this paper was presented at PKC'99, Second International Workshop on Practice and Theory in Public Key Cryptography, Kamakura, Japan, March, 1-3, 1999.

before one can determine even one message block. One of the design principles of a hash function is to make the hash value dependent on the entire input message and to make finding collision hard. For existing hash functions, collision may be found by modifying any blocks of the input message. In this paper, we propose secure hash functions with all-or-nothing property that is a kind of a new encryption mode proposed by Rivest. The proposed schemes use the existing hash functions without changing the hash algorithm, and make them secure against known attacks. Also the proposed schemes can be easily extended to the MAC, which can provide a message confidentiality as well as authentication.

The remainder of this paper is organized as follows. In Section 2, we summarize the hash function and in Section 3, all-or-nothing property is described. In Section 4, we propose and analyze new construction schemes of the hash function with all-or-nothing property. Finally, we have conclusions in Section 5.

2. HASH FUNCTIONS

Hash functions(more exactly cryptographic hash functions) are functions that map bitstrings of arbitrary finite length into strings of fixed length. This output is commonly called a hash value(sometimes called a message digest, or a fingerprint). Given a hash function h and an input x, computing $h(x)$ must be easy. A one-way hash function must satisfy the following properties [1]:

– *preimage resistance*: it is computationally infeasible to find any input which hashes to any pre-specified output. That is, given a y in the image of h, it is computationally infeasible to find an input x such that $h(x) = y$.
– *second preimage resistance*: it is computationally infeasible to find any second input which has the same output as any specified input. That is, given a y in the image of $h(x)$, it is computationally infeasible to find an input $x' \neq x$ such that $h(x') = y$.

A cryptographically useful hash function must satisfy the following additional property [1]:

– *collision resistance*: it is computationally infeasible to find a collision. That is, it is computationally infeasible to find a pair of two distinct inputs x and x' such that $h(x)=h(x')$.

Almost all hash functions are iterative processes which hash inputs of arbitrary length by processing successive fixed-size blocks of input. The input X is padded to a multiple of block length and subsequently divided into t blocks X_1 through X_t. The hash function h can then be described as follows:

$$H_0 = IV, \quad H_i = f(H_{i-1}, X_i), 1 \le i \le t, \quad h(X) = g(H_t) \tag{1}$$

where f is the *compression function* of h, H_{i-1} is the *chaining variable* between stage i-1 and stage i, and IV is the initial value. An output transformation g is used in a final step to map the last chaining variable to a result $g(H_t)$; g is often the identity mapping $g(H_t)=H_t$.

The computation of the hash value is dependent on the chaining variable. At the start of hashing, this chaining variable has a fixed initial value which is specified as a part of the algorithm. The compression function is then used to update the value of this chaining variable in a suitably complex way under the action and the influence of a part of the message being hashed. This process continues recursively, with the chaining variable being updated under the action of different parts of the message, until all the messages have been used. The final value of the chaining variable is then output as the hash value corresponding to these messages. Based on the construction of the internal compression function, hash functions can be classified as follows[1]:

- hash functions based on block ciphers
- hash functions based on modular arithmetic
- dedicated hash functions.

Dedicated hash functions have fast processing speed and are independent of other system subcomponents such as block ciphers and modular multiplication subcomponents. Most of existing dedicated hash functions have the structure similar to that of MD4 [3] which was designed by R. Rivest in 1990. The typical examples of the dedicated hash functions are the MD family hash functions such as MD5 [4], RIPEMD-160 [5], SHA-1 [6], HAVAL [7], and SMD[8]. On recent, a quite different class of dedicated hash functions based on cellular automata over GF(q) was reported in [9].

According to the theory of Merkle [10] and Damgård [11], *MD-strengthening* denotes appending an additional block at the end of the input string containing its length. It is possible to relate the security of hash function h to the security of compression function f and output function g according to the following theorem.

Theorem 1(cf. [9]). Let $h(\cdot)$ be an iterated hash function with MD-strengthening. Then preimage and collision attacks on $h(\cdot)$ (where an attacker can choose IV freely) have roughly the same complexity as the corresponding attacks on the compression function f and the output function g.

Theorem 1 gives a lower bound on the security of $h(\cdot)$. According to [12] the iterated hash functions based on the Davies-Meyer compression function given by the following:

$$f(M_i, H_{i-1}) = E_{M_i}(H_{i-1}) \oplus H_{i-1}, \tag{2}$$

where $E_K(\cdot)$ is a block cipher controlled by the key K, are believed to be as secure as the underlying cipher $E_K(\cdot)$ is.

As a direct extension of the results of security related to cipher block chaining and the assumption 2 from [9] (which is a standard one in cryptography today), we assume the following.

Assumption 2. Let the compression function f be the Davies-Meyer function and the employed cryptographic transformation be a secure one. Then finding collisions for f

requires about $2^{n/2}$ operations, and finding a preimage for f requires about 2^n operations, if we assume an n-bit hash result.

3. ALL-OR-NOTHING PROPERTY

In 1997, Rivest proposed an all-or-nothing encryption, a new encryption mode for block ciphers [2]. 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.

The problem with most popular encryption modes is that the adversary can obtain one block of the plaintext by decrypting just one block of the ciphertext. This makes the adversary's key-search problem relatively easy, since decrypting a single ciphertext block is enough to test a candidate key.

Let us say that an encryption mode for a block cipher is ***separable*** if it has the property that an adversary can determine one block of plaintext by decrypting just one block of ciphertext. Rivest defined ***strongly non-separable*** mode as follows [2]:

Definition 3. Suppose that a block cipher encryption mode transforms a sequence $m_1, m_2, \ldots, m_s$ of s message blocks into a sequence $c_1, c_2, \ldots, c_t$ of t ciphertext blocks for some $t (t \geq s)$. We say that the encryption mode is **strongly non-separable** if it is computationally infeasible to determine even one message block m_i (or any property of a particular message block m_i) without decrypting all t ciphertext blocks.

Rivest proposed the strongly non-separable modes as follows [2]:

- Transform the message sequence $m_1, m_2, \ldots, m_s$ into a "pseudo-message" sequence $m'_1, m'_2, \ldots, m'_{s'}$ (for some $s' \geq s$) with an "all-or-nothing transform".
- Encrypt the pseudo-message with an ordinary encryption mode with the given cryptographic key K to obtain the ciphertext sequence $c_1, c_2, \ldots, c_t$.

We call encryption mode of this type "all-or-nothing encryption modes." To make this work, the all-or-nothing transform has to have certain properties as the following [2]:

Definition 4. A transform T mapping a message sequence $m_1, m_2, \ldots, m_s$ into a pseudo-message sequence $m'_1, m'_2, \ldots, m'_{s'}$ is said to be an **all-or-nothing transform** if

- The transform T is reversible: given the pseudo-message sequence, one can obtain the original message sequence.
- Both the transform T and its inverse are efficiently computable.
- It is computationally infeasible to compute any function of any message block if any one of the pseudo-message blocks is unknown.

An all-or-nothing encryption mode is strongly non-separable. The all-or-nothing transform is not itself encryption, since it makes no use of any secret key information.

The actual encryption is the operation that encrypts the pseudo-message. An all-or-nothing transform is a fixed public transform that anyone can perform on the message to obtain the pseudo-message, or invert given the pseudo-message to obtain the original message.

Rivest proposed the all-or-nothing transform which is referred to "package transform", as follows[2]:

(1) Let the input message be $m_1, m_2, \ldots, m_s$ (m_i is a b-bit string).
(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:
   A. $m'_i = m_i \oplus E_K(i)$ for $i=1,2,\ldots,s$ ($E_K(\cdot)$ is a b-bit block cipher with the key K).
   B. Let

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

   where

$$h_i = E_{K0}(m'_i \oplus i) \text{ for } i=1,2,\ldots,s,$$

   where $K0$ 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 computationally infeasible. It is easy to see that the package transform is invertible:

$$K = m'_{s'} \oplus h_1 \oplus \ldots \oplus h_s ,$$
$$m_i = m'_i \oplus E_K(i) \quad \text{for } i=1,2,\ldots,s.$$

If any block of pseudo-message sequence is unknown, the K cannot be computed, and so it is computationally 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 another all-or-nothing transform, it is possible to apply the bijective function. First, choose the bijective function $\tau(\cdot)$ which maps the entire input message to pseudo-message, and compute the pseudo-message $M' = \tau(M)$. Second, encrypt the pseudo-message M' using the block cipher. The receiver can obtain the original message M by applying to $\tau^{-1}(M')$ after decrypting the ciphertext. Here, $\tau^{-1}(\cdot)$ is the inverse of $\tau(\cdot)$. The selected bijective function must satisfy that computing inverse of the bijective function is computationally infeasible if any one block of the pseudo-message is unknown. Even though there are some bijective functions satisfying above property, the careful considerations will be needed for practical purposes.

In the following section, we propose construction schemes of hash functions with all-or-nothing property. Our aim of a new design of the hash function is to obtain a fixed hash value dependent on the pseudo-message which was originated from the entire message using all-or-nothing transform. Thus all-or-nothing property dependent on the entire message is suitable for constructing a hash function.

4. HASH FUNCTION WITH ALL-OR-NOTHING PROPERTY

Even though a lot of dedicated hash functions are published until now, as far as the authors are concerned, no hash functions with all-or-nothing property have been reported so far. This section points out some of them which are main results of this paper. All-or-nothing transform has been considered as a new encryption mode for the design of the block cipher[2], and it can be used to make the encryption mode strongly non-separable. This encryption mode is possible to enhance the security of the ciphertext blocks because an adversary attempting a brute-force attack on the keys has to decrypt all ciphertext blocks. This non-separability between a message block and all ciphertext blocks can be applied to the proposal of the hash functions with all-or-nothing property, which they authenticate a message by the hash value that has to be dependent on as many as possible in the original message. In this section, we propose three hash functions with all-or-nothing property which are using arbitrary member of a family of hash functions[13]. The proposed hash functions follow the Davies-Meyer principle(see relation (2) in section 2). Moreover, we describe their security analysis and design the MAC using proposed hash functions. For the concise design of hash functions, we restrict an optimal output function g in (1), section 2 as an identity mapping. Three design schemes are characterized by the usage of different all-or-nothing transforms.

We use the following notations:

- n is a length of an output and a chaining variable of a hash function.
- k is a length of an input block of a compression function of a hash function($n \le k$).
- X is an input message.
- X' is a pseudo-message resulting from all-or-nothing transform.
- K is a randomly chosen k -bit key.
- K_p is a fixed and publicly-known k-bit key.
- $h_x(y)$ is a hash function with an input y and an initial value x.
- IV is an initial value of a hash function.
- $\oplus$ denotes a bitwise exclusive or.
- $K \oplus i$ means bitwise exclusive or with respect to the binary representation of integers K and i.
- $||$ denotes a concatenation between two words.
- $\overline{Z}$ denotes a k -bit string by iterating an n-bit string Z, if necessary, it is adjustable to the length of k -bit.

4.1 Hash Function With All-or-Nothing Property – 1 (HAON-1)

The first hash function with all-or-nothing property operates as follows:

A. Sender

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

$$X'_0 = IV, X'_i = X_i \oplus h_{X'_{i-1}}(K \oplus i), i = 1,2,\cdots,s .$$

(4) Compute the last pseudo-message block, X'_{s+1} (k-bit length).

$$X'_{s+1} = K \oplus \{\overline{h_{X'_1}(K_p \oplus 1) \oplus \cdots \oplus h_{X'_s}(K_p \oplus s)}\} .$$

(5) Send ($X' \| h_{IV}(X')$).

B. Receiver

(1) Receive ($X' \| Z$) and check if $Z = h_{IV}(X')$.
(2) Split the pseudo-message X' into s blocks of n-bit each, $X' = (X'_1, X'_2, \ldots, X'_s)$ and X'_{s+1} of k-bit.
(3) Recover the random key K.

$$X'_{s+1} = K \oplus \{\overline{h_{X'_1}(K_p \oplus 1) \oplus \cdots \oplus h_{X'_s}(K_p \oplus s)}\} .$$

(4) Recover the original message.

$$X'_0 = IV, X_i = X'_i \oplus h_{X'_{i-1}}(K \oplus i), i = 1,2,\cdots,s .$$

HAON-1 uses all-or-nothing transform based on a hash function. This scheme has all properties of all-or-nothing transform proposed by Rivest. An adversary may try to attack after recovering the original message, however, the manipulation of the pseudo-message is not useful attack in this scheme. If an attacker does not know any one block of the pseudo-message, he can't recover the random key K and obtain the original message. For an effective attack, he has to find a collision of the pseudo-message with the same hash value in advance, and then searches the message and random key mapped to this pseudo-message. Moreover, the fact that the (i-1)th pseudo-message block is used to compute the ith pseudo-message block do increase the security. HAON-1 pays a penalty of approximately a factor of three in computational time compared to that of the original hash function. The simulation result shows that the performance of HAON-1 using the SHA-1 is about 10.05 Mbytes/sec, while that of package transform using RC6 block cipher[14] is 3.66 Mbytes/sec.

4.2 Hash Function With All-or-Nothing Property – 2 (HAON-2)

A. Sender

(1) Split an input message X into s blocks of n-bit each: $X = (X_1, X_2, \ldots, X_s)$.
(2) Generate a random key K of k-bit.

(3) Compute a pseudo-message *X'* by all-or-nothing transform.

$$X'_0 = IV, X_0 = 0, X'_i = X_i \oplus h_{X'_{i-1}}(K \oplus (\overline{X_{i-1} \| i})), i = 1,2,\cdots,s .$$

(4) Compute the last pseudo-message block, X'_{s+1} (*k*-bit length).

$$MD = h_{K_p}(X'_1 \| \cdots \| X'_s \| h_{IV}(K_p)), \quad X'_{s+1} = K \oplus \overline{MD} .$$

(5) Send $(X' \| h_{MD}(X'_{s+1}))$.

B. Receiver

(1) Receive $(X' \| Z)$.
(2) Split the pseudo-message *X'* into *s* blocks of *n*-bit each, $X' = (X'_1, X'_2, \ldots, X'_s)$ and X'_{s+1} of *k*-bit.
(3) Recover the random key *K*.

$$MD' = h_{K_p}(X'_1 \| \cdots \| X'_s \| h_{IV}(K_p)), \quad K = X'_{s+1} \oplus \overline{MD'} .$$

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

$$X'_0 = IV, X_0 = 0, X_i = X'_i \oplus h_{X'_{i-1}}(K \oplus (\overline{X_{i-1} \| i})), i = 1,2,\cdots,s .$$

This scheme is an improved version of HAON-1. All-or-nothing transform depends on the underlying hash function, and has all properties of HAON-1. It is more difficult to find the collision of pseudo-messages since all intermediate chaining variables are used in computing of the last pseudo-message block, X'_{s+1}. Moreover, we increase the security by using the hash value of the original message for computing the pseudo-message. While HAON-1 pays a penalty of a factor of three, HAON-2 improves the efficiency by applying the hash function twice. The simulation result shows that the performance of HAON-2 using the SHA-1 is about 15.07 Mbytes/sec. Compared to HAON-1 using the SHA-1, the performance is improved about 50%.

4.3 Hash Function With All-or-Nothing Property – 3 (HAON-3)

We propose an improved version of HAON-2. HAON-3 provides better security without additional computation overhead than HAON-2. Here we assume K_p is a publicly-known constant of *n*-bit. HAON-3 operates as follows:

A. Sender

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

$$X'_0 = IV, X_0 = K, X'_i = X_i \oplus h_{X_{i-1}}(X'_{i-1} \| (K \oplus i)), i = 1,2,\cdots,s$$

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

$$MD = h_{K_p}(X'_1 \| \cdots \| X'_s \| h_{IV}(K_p \oplus (s+1))), \quad X'_{s+1} = K \oplus MD .$$

(5) Send ($X' \| h_{MD}(X'_{s+1})$).

B. Receiver

(1) Receive ($X' \| Z$).

(2) Splitting the pseudo-message X' into s +1 blocks of n-bit each, $X' = (X'_1, X'_2, \ldots, X'_s, X'_{s+1})$.

(3) Recover K.

$$MD' = h_{K_p}(X'_1 \| \cdots \| X'_s \| h_{IV}(K_p \oplus (s+1))), \quad K = X'_{s+1} \oplus MD .$$

(4) Recover the original message.

$$X'_0 = IV, X_0 = K, X_i = X'_i \oplus h_{X_{i-1}}(X'_{i-1} \| (K \oplus i)), i = 1,2,\cdots,s .$$

(5) Check if $K = h_{IV}(X_1 X_2 \ldots X_s)$.

HAON-3 is an improved version of HAON-2 and has all properties of HAON-2. In HAON-3, each pseudo-message block depends on the entire original message since K is a hash value of the entire original message. Furthermore, HAON-3 does not require an additional computation because the hash value K corresponds to the k-bit random number in HAON-2.

4.4 Security Analysis of the Proposed Schemes

Note that according to the Theorem 1, a lower bound on security of the proposed hash function is determined by the characteristics of its compression and output functions. Since we have only confined to the identity mapping as an output function g, the security will be considered through the security of the underlying hash function h. The facts and discussions which are given in this section imply that the proposed schemes have ideal security, i.e., given a hash output, producing each of a preimage or second preimage requires approximately 2^n operations and producing a collision requires approximately $2^{n/2}$ operations. If n is greater than or equal to 160 bits, it can provide a sufficient security against the birthday attack so far. In particular, to find collisions, the existing known attacks like those of MD4, MD5 and RIPEMD, can't be directly applicable to HAONs due to the following reason.

Most of the existing known attacks against hash functions depend on manipulation of the message blocks. Since HAONs send the pseudo-message instead of the original message, an adversary must intercept the pseudo-message and recover the original mes-

sage. The manipulation of pseudo-message is not effective for the correct recovery of original message by the receiver because the random key K cannot be recovered exactly. Under this condition, if an attacker recover the original message and modify it(so the resulting of pseudo-message is altered), then the message digest would be different from the original one. Thus an adversary has to find a different pseudo-message pair(i.e., a collision pseudo-message pair) with the same hash value in advance, and then compute the input message pair corresponding to the collision pseudo-message one, or find a K which maps the input message to the collision pseudo-message pair. That is, an attacker has to search a collision of the pseudo-message and find a random key K which maps the input message to the pseudo-message, or find an input message mapped to the pseudo-message with the key K. This requires the search of a random key K or an input message, as well as finding a collision pair of the pseudo-message. Therefore, to find a collision pair, one must find a collision of pseudo-message with the same hash value in advance, which requires $2^{n/2}$ operations for HAON-1, and then find original message pair mapped to the pseudo-image pair. Finding an input message pair mapped to the pseudo-image pair requires $2^{n-s/2}$ or 2^k operations(s is the number of n-bit blocks of original message). Hence total of $2^{n/2\cdot(1+s)}$ or $2^{(n+k)/2}$ operations are needed to find a collision pair in HAON-1. Similarly, for HAON-2, it requires total of $2^{n/2\cdot(1+s)}$ or $2^{(n+k)/2}$ operations to succeed above attack. However, HAON-3 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. To do this, HAON-3 requires $2^{n/2+\beta}$ operations to find a collision pair at the worst case(β is a length of the input message).

To find a collision message of a hash function, one may try to find message pair having the same message digest independent of the intermediate chaining variables. HAON-1 uses the (i-1)th pseudo-message block to generate the ith pseudo-message block, and HAON-2 also uses the (i-1)th pseudo-message block and (i-1)th original message block to compute the ith pseudo-message block. Also in HAON-3, the ith pseudo-message block depends on the entire original input message and the (i-1)th pseudo-message block. Thus, the proposed schemes are secure against attacks by the manipulation of the message block.

4.5 The MAC Application of HAONs

The proposed schemes can be easily applied to the MAC. By applying HAONs to the MAC, it is possible to provide both the authentication and the confidentiality for a message. In this case, two communication parties between the sender and the receiver have to securely keep publicly-known random constant K_p in the MAC application. This MAC construction(HAON-MAC) may be considered as the variant of HMAC proposed by Bellare, et. al. [15] such like that

$$HMAC_k(x) = h(\bar{k} \oplus opad, h(\bar{k} \oplus ipad, x)) \tag{3}$$

where k is a secret information which held by two parties.

The proposed HAON-MAC operates as follows: First, it generates a pseudo-message by applying an HAON h and next, h is applied to the pseudo-message once again. Hence the generation of pseudo-message from an original message is considered as the inner hashing process of HMAC. Thus HAON-MAC has similar security properties as that of HMAC[15]. HAON-MAC requires $2^{(n+k)/2}$ or $2^{n/2+\beta}$ operations to find a collision pair. Therefore, the proposed MAC is more secure than existing schemes. Moreover, the proposed MAC can provide the message confidentiality as well as authentication. An attacker who does not know K_p may try to find the random key K for recovering the entire original message. To find the random key K, in case of HAON-1, it is required to the finding of K'_p having the same values as $h_{X'_i}(K_p \oplus i)$. For HAON-2, one must find the K'_p which generates the same values as $MD' = h_{K_p}(X'_1 \| \cdots \| X'_s \| h_{IV}(K_p))$. And in the case of HAON-3, it requires the finding of the K'_p having the same value as $MD' = h_{K_p}(X'_1 \| \cdots \| X'_s \| h_{IV}(K_p \oplus (s+1)))$. It corresponds to the envelope method which is one of the MAC construction using the hash function. The best known attack for this scheme is the divide-and-conquer attack proposed by Preneel and van Oorschot[16]. If we use SHA-1 or RIPEMD-160 which is considered as a secure hash function, this attack is computationally infeasible. For an adversary who tries to decrypt the only one block, the proposed schemes require 2^n operations. If n is more than or equal to 160-bit length, it is computationally infeasible.

We can improve the security by adding some overheads to the above schemes. We can encrypt the last pseudo-message block and message digest pair (X'_{s+1}, MD) 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. If we use SHA-1, X'_{s+1} and MD are 160-bit length in the case of HAON-2 and HAON-3. Therefore, the total length of the encrypting block becomes total of 320 bits. The degradation of the performance is negligible in this case. However an attacker must find the keys K_S and K_p for decrypting the entire message. The confidentiality of the proposed MAC has the property that one must decrypt the entire ciphertext before one can determine even one message block. It can provide both the authentication and confidentiality by using only hash functions. The proposed HAONs are constructed by using only the hash functions. They are more efficient than all-or-nothing transform using the block cipher such as the package transform proposed by Rivest, and they can avoid the patent and the export restrictions of some algorithms. The proposed schemes pay a penalty of approximately a factor of two in the time compared to an ordinary hash function, but they can be performed efficiently in parallel.

5. CONCLUSIONS

This paper addresses the applicability of some hash functions with the concept of All-Or-Nothing property which was proposed by Rivest in 1997. The aim of our paper is to apply AON concept to cryptographic hash functions since the hash values of hash

functions must dependent on the entire message to prevent from the collision attacks. The proposed schemes use existing hash functions without changing their structures and make more secure against known attacks by applying a minor overhead. Moreover, we also proposed the MAC, which can provide both the authentication and the confidentiality of messages.

The security of the proposed HAONs was analyzed, which was based on the so far published known attacks against hash functions. The analysis implies that the proposed HAONs are more secure than existing schemes, i.e., given an n-bit hash value, producing of collision requires testing of approximately over than $2^{(n+k)/2}$ or $2^{n/2+\beta}$ (β is a length of the input message) hypothesis, which is more secure than $2^{n/2}$ of the original dedicated hash functions. Of course, some overheads are necessary for producing a pseudo-message, at least twice of operations of a hash function.

REFERENCES

1. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
2. R. L. Rivest, "All-or-nothing encryption and the package transform," in *Proceedings of the 1997 Fast Software Encryption Conference*, LNCS, Vol. 1267, 1997, pp. 210-218.
3. R. L. Rivest, "The MD4 message-digest algorithm," *Advances in Cryptology-CRYPTO 90*, LNCS, Vol. 537, 1991, pp. 303-311.
4. RFC 1321, "The MD5 message-digest algorithm," Internet request for comments 1321, R. L. Rivest, April 1992.
5. H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD," *Fast Software Encryption-Cambridge Workshop*, LNCS, Vol.1039, 1996, pp. 71-82.
6. FIPS 180-1, "Secure hash standard," Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce / NIST, 1995.
7. Y. Zheng, J. Pieprzyk, and J. Sebery, "HAVAL – a one-way hashing algorithm with variable length of output," *Advances in Cryptology-AUSCRYPT 92*, LNCS, Vol. 718, 1993, pp. 83-104.
8. S. U. Shin, K. H. Rhee, D. H. Ryu, and S. J. Lee, "A new hash function based on MDx-family and its application to MAC," *PKC'98(International Workshop on Practice and Theory in Public Key Cryptography)*, LNCS, Vol. 1431, 1998, pp. 234-246.
9. M. Mihaljevic, Y. Zheng, and H. Imai, "A family of fast dedicated one-way hash functions based on linear cellular automata over GF(q)," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E82-A, No. 1, 1999, pp. 40-47.
10. R. Merkle, "One way hash functions and DES," *Advances in Cryptology-CRYPTO 89*, LNCS, Vol. 435, 1990, pp. 428-446.
11. I. B. Damgård, "A design principle for hash functions," *Advances in Cryptology-CRYPTO 89*, LNCS, Vol. 435, 1990, pp. 416-427.
12. L. Knudsen, B. Preneel, "Fast and secure hashing based on codes," *Advances in Cryptology-CRYPTO 97*, LNCS, Vol. 1294, 1997, pp. 485-498.

13. S. U. Shin, K. H. Rhee, and J. W. Yoon, "Hash functions and the MAC using all-or-nothing property," *PKC'99(International Workshop on Practice and Theory in Public Key Cryptography)*, LNCS, Vol. 1560, 1999, pp. 263-275.
14. R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The RC6 block cipher," a block cipher submitted for consideration as the new AES.
15. M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," *Advances in Cryptology-CRYPTO 96*, LNCS, Vol. 1109, 1996, pp. 1-15.
16. B. Preneel and P. van Oorschot, "MDx-MAC and building fast MACs from hash functions," *Advances in Cryptology-CRYPTO 95*, LNCS, Vol. 963, 1995, pp. 1-14.

Sang Uk Shin received his M.S. and Ph.D. degrees from PuKyong National University, Pusan Korea in 1997 and 2000, respectively. He now works as a senior researcher in Electronics and Telecommunications Research Institute, Daejeon, Korea. His research interests center on design of hash functions and their applications, and security analysis of cryptographic algorithms.

Kyung Hyune Rhee received his M.S. and Ph.D. degrees from Korea Advanced Institute of Science and Technology, Daejeon Korea in 1985 and 1992, respectively. He worked as a senior researcher in Electronics and Telecommunications Research Institute, Daejeon Korea from 1985 to 1993. He is currently an associate professor in the Division of Electronic, Computer and Telecommunication Engineering of PuKyong National University. His research interests center on key management and its applications, mobile communications security and security evaluation of cryptographic algorithms.