Vol.2009-SE-164 No.7
Vol.2009-EMB-13 No.7
Vol.2009-CSEC-45 No.7
2009/5/28

Consideration of Lightweight Chameleon Hash Function (2)

Kazumasa Omote[†1]

Chameleon hash functions are trapdoor one-way hash functions, which prevent everyone except the holder of the trapdoor information from computing the collisions for a randomly given input. A lot of chameleon hash functions have been proposed so far. However, many existing chameleon hash functions have two drawbacks: (1) any chameleon hash function requires modulo operations with a large computational complexity; (2) anyone can calculate new collisions of most chameleon hash functions if a collision has already been generated once. In this paper, we construct a novel chameleon hash function using only hash function and XOR operation to solve the two problems listed above. Moreover, we propose an efficient k-times on-line/off-line signature scheme based on the proposed chameleon hash function. The proposed chameleon hash function is an improvement of SCIS2009.

1. Introduction

Chameleon hash functions are trapdoor one-way hash functions, which prevent everyone except the holder of the trapdoor information from computing the collisions for a randomly given input. A lot of chameleon hash functions have been proposed so far. Chameleon hash functions were first introduced by Krawczyk and Rabin[1], who proposed a chameleon hash function based on the discrete log assumption. In Crypto 2001, Shamir and Tauman[2] used a chameleon hash function to develop a new paradigm, named "*hash-sign-switch*", for designing an efficient on-line/off-line signature schemes. These chameleon hash functions were based on the discrete log assumption and the factoring assumption. Recently, some pairing-based chameleon hash functions[3)–5)], a double-trapdoor chameleon hash function[6], and some ID-based chameleon hash functions[3),5),7),8)] have been proposed . Especially, two chameleon hash functions[7),8)] are able to restrict the collision calculation of chameleon hash function using identities.

Many existent chameleon hash functions, however, have two drawbacks: (1) any chameleon hash function requires modulo operations with a large computational complexity; (2) anyone can calculate new collisions of most chameleon hash functions if a collision has already been generated once. As for (1), to the best of our knowledge, all chameleon hash functions used modulo operations. We describe about (2) in the following paragraph.

Ateniese and de Medeiros[9] first pointed out the key exposure problem of chameleon hashing: the private trapdoor key is exposed if a collision has already been generated once in some chameleon hash functions including the original one. As a result, anyone can also calculate new collisions of chameleon hash function using the exposed trapdoor key. Some chameleon hash functions[5),6)] have another problem: anyone can calculate new collisions if a collision has already been generated once, although the trapdoor key is not exposed. This feature is useful for a *chameleon signature*, but it can be a problem in "hash-sign-switch" paradigm. On the other hand, some schemes[2),7),8)] does not have the second problem listed above. One chameleon hash function[2] is based on the factoring assumption. Some chameleon hash functions[7),8)] can restrict the collision calculation by identities.

In this paper, we construct a novel chameleon hash function using only hash function and XOR operation to solve the two problems listed above. To the best of our knowledge, all chameleon hash functions used modulo operations. Moreover, we propose an efficient k-times on-line/off-line signature scheme based on the proposed chameleon hash function.

The rest of the paper is organized as follows. Some definitions are provided in Section 2. We propose an efficient chameleon hash function in Section 3, discuss the security and efficiency analysis of the proposed scheme in Section 4, and describe an efficient k-times on-line/off-line signature scheme in Section 5. We finally conclude this paper in Section 6.

2. Definition

First, we give some definitions used in our scheme.

Definition 1 (Cryptographic secure hash function) A function H is a

†1 Japan Advanced Institute of Science and Technology (JAIST)

cryptographic secure hash function if it satisfies the following properties[10]. We define that λ is a security parameter of H.

(1) Function H maps bit strings, either of an arbitrary length or a predetermined length, to strings of a fixed length λ ($H : \{0,1\}^* \mapsto \{0,1\}^\lambda$).

(2) One-wayness: Given x, it is easy to compute $H(x)$. Conversely, given $H(x)$, it is computationally infeasible to compute x.

(3) Collision resistance: For any given x, it is computationally infeasible to find y ($\neq x$) such that $H(x) = H(y)$

Definition 2 (One-way hash chain) Select a cryptographic secure hash function H defined by Definition 1 with security parameter λ, $H : \{0,1\}^* \mapsto \{0,1\}^\lambda$. Pick a seed c_0 randomly and apply H recursively k times to the initial seed c_0 to generate a one-way hash chain, c_0, c_1, ..., c_k ($c_i = H(c_{i-1})$, $1 \leq i \leq k$). Note that the one-way hash chain is also denoted as $c_i = H^{i-1}(c_0)$ ($1 \leq i \leq k$). One-way hash chain is a widely-used cryptographic primitive. One of the first uses of one-way chain was for one-time passwords by Lamport[11].

We introduce the basic notion of chameleon hash family by Shamir and Tauman[2]. Every trapdoor hash function is associated with a pair of public key and private key, referred to as the hash key HK and the trapdoor key TK, respectively.

Definition 3 (Chameleon hash family) A chameleon hash family consists of a pair $(\mathcal{I}, \mathcal{H})$:

- $\mathcal{I}$ is a probabilistic polynomial-time key generation algorithm that on input 1^λ, outputs a pair (HK, TK) such that the sizes of HK, TK are polynomially related to λ.
- $\mathcal{H}$ is a family of randomized hash functions. Every hash function in $\mathcal{H}$ is associated with a hash key HK, and is applied to a message from a space $\mathcal{M}$ and a random element from a finite space $\mathcal{R}$. The output of the hash function H_{HK} does not depend on TK.

A chameleon hash family $(\mathcal{I}, \mathcal{H})$ has the following properties:

(1) *Efficiency*: Given a hash key HK and a pair $(m, r) \in \mathcal{M} \times \mathcal{R}$, $H_{HK}(m, r)$ is computable in polynomial time.

(2) *Collision resistance*: There is no probabilistic polynomial time algorithm $\mathcal{A}$ that on input HK outputs, with a probability which is not negligible, two pairs (m_1, r_1), $(m_2, r_2) \in \mathcal{M} \times \mathcal{R}$ that satisfy $m_1 \neq m_2$ and $H_{HK}(m_1, r_1) = H_{HK}(m_2, r_2)$ (the probability is over HK, where $(HK, TK) \leftarrow \mathcal{I}(1^\lambda)$, and over the random coin tosses of algorithm $\mathcal{A}$).

(3) *Trapdoor collisions*: There exists a probabilistic polynomial time algorithm that given a pair $(HK, TK) \leftarrow \mathcal{I}(1^\lambda)$, a pair $(m_1, r_1) \in \mathcal{M} \times \mathcal{R}$, and an additional message $m_2 \in \mathcal{M}$, outputs a value $r_2 \in \mathcal{R}$ such that:

- $H_{HK}(m_1, r_1) = H_{HK}(m_2, r_2)$.
- If r_1 is uniformly distributed in $\mathcal{R}$ then the distribution of r_2 is computationally indistinguishable from uniform in $\mathcal{R}$.

3. The proposed chameleon hash function

Many existent chameleon hash functions have two drawbacks: (1) any chameleon hash function requires modulo operations with a large computational complexity; (2) anyone can calculate new collisions of chameleon hash function if a collision has already been generated once. These two problems are addressed by the proposed chameleon hash function described in this section. To the best of our knowledge, we firstly propose an efficient chameleon hash function without modulo operations.

3.1 System parameters generation

Let $H_1 : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ be a hash function which satisfies only one-wayness of a cryptographic secure hash function, and λ be a security parameter. At first, we pick a seed $c_0 \in_R \{0,1\}^\lambda$ randomly and calculate a hash chain $\{c_0, c_1, \ldots, c_k\}$ ($c_i = H_1^i(c_0)$) ($1 \leq i \leq k$). Then, we define a public hash key $HK_p = (c_k, c_p)$, a private trapdoor key $TK = c_0$ and a hash chain space $\mathcal{C} = \{c_0, \ldots, c_k\}$.

A part of the hash key c_p in the proposed scheme is updated every time a collision is generated. If c_p is not updated, we can easily calculate a new collision. The c_p denotes the revealed latest c_i. We use c_p from $p \leftarrow k$ to $p \leftarrow 0$ one by one. Note that only a user who knows TK can update c_p in HK_p.

3.2 A chameleon hash function

Given the hash key HK_p and a pair $(m, r) \in \mathcal{M} \times \mathcal{R}$, the proposed chameleon hash function $H_{HK_p} : \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ is defined as follows:

$$H_{HK_p}(m, r) = m \oplus r \oplus c_p. \quad (1)$$

Note that there exists v such that $c_k = H_1^v(c_p)$. Given the hash key HK_p, a pair

(m_1, r_1) and an additional message m_2, we want to find a collision r_2 such that $H_{HK_p}(m_1, r_1) = H_{HK_{p-1}}(m_2, r_2)$. It is necessary to update the hash key from H_{HK_p} to $H_{HK_{p-1}}$ to obtain a new hash collision r_2.

3.3 Example

We show an example of the calculation of the proposed chameleon hash value and the collision. Let U be a only user who knows the trapdoor key TK, $\mathcal{C} = \{c_0, c_1, c_2, c_3, c_4, c_5\}$ $(k = 5)$ be a hash chain space, and $HK_3 = (c_5, c_3)$ be a public hash key. The values of c_5, c_4, c_3 are revealed because c_3 is included in HK_3. We pick a seed $r \in_R \{0,1\}^\lambda$, and calculate the chameleon hash value $H_{HK_3}(m, r) = m \oplus r \oplus c_3$ of a message m. Given an additional message m', U can find a collision r' as follows. At first, U calculates $c_2 = H_1^2(c_0)$ using $TK = c_0$, and updates the hash key from $H_{HK_3} = (c_5, c_3)$ to $H_{HK_2} = (c_5, c_2)$. Then U calculates a collision $r' = m \oplus m' \oplus r \oplus c_3 \oplus c_2$ such that $H_{HK_3}(m, r) = H_{HK_2}(m', r')$.

3.4 New property

A user who knows TK can easily compute new collisions owing to a feature of trapdoor collision. Moreover, most chameleon hash functions have a problem that anyone can calculate new collisions even without knowing know TK, if a collision has already been generated once. We restrict the collision calculation of the proposed chameleon hash function: anyone can calculate only k-times collisions. In this paper, we define k-times trapdoor collisions as a new property of chameleon hash function.

Definition 4 (k-times trapdoor collisions)
A probabilistic polynomial time algorithm outputs $r_2 \in \mathcal{R}$ only k times in Definition 3.

4. Analysis

In this section, we discuss security and efficiency about the proposed chameleon hash function. Refer to the previous scheme[12] for the comparison of features of chameleon hash functions.

4.1 Security

We show that the chameleon hash function H_{HK_p} in Equation (1) satisfies three kinds of properties: collision resistance, trapdoor collisions and k-times trapdoor collisions. We can give proofs of Theorem 1 and Theorem 2 in a similar fashion to the proofs for existing schemes[2),6)].

Theorem 1 (Collision resistance) The proposed chameleon hash function satisfies collision resistance if the hash function H_1 satisfies only one-wayness of a cryptographic secure hash function.

Proof. Assume to the contrary, there exists a polynomial time algorithm $\mathcal{A}_1$ that on input HK_p outputs, with a probability which is not negligible, two pairs (m_1, r_1), $(m_2, r_2) \in \mathcal{M} \times \mathcal{R}$ that satisfy $m_1 \neq m_2$ and $H_{HK}(m_1, r_1) = H_{HK}(m_2, r_2)$. Then, we can use $\mathcal{A}_1$ to break the one-wayness of H_1 as follows: For a randomly given instance a, define $c_p = a = H_1(b)$ (The value of b is not disclosed). $\mathcal{A}_1$ can output two pairs (m_1, r_1), (m_2, r_2) that satisfy $m_1 \neq m_2$ and $H_{HK_p}(m_1, r_1) = H_{HK_{p-1}}(m_2, r_2)$ when the current hash key is HK_p. We can compute $b = c_{p-1} = a \oplus m_1 \oplus m_2 \oplus r_1 \oplus r_2$. Therefore, we can break the one-wayness of H_1. On the other hand, if the hash function H_1 does not satisfy collision resistance, we can compute a' such that $H_1(a) = H_1(a')$. However, we cannot eventually compute c_{p-1} from a and a' because of one-wayness, that is to say, we cannot update HK_p. Hence the proposed chameleon hash function satisfies collision resistance even if H_1 does not satisfy collision resistance. ∎

Theorem 2 (Trapdoor collisions) The proposed chameleon hash function satisfies trapdoor collisions.

Proof. Assume that we are given the hash and trapdoor key pair (HK_p, TK), a pair $(m_1, r_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$, and an additional message $m_2 \in \{0,1\}^\lambda$, we want to find $r_2 \in \{0,1\}^\lambda$ such that $H_{HK_p}(m_1, r_1) = H_{HK_{p-1}}(m_2, r_2)$. We compute $c_{p-1} = H_1^{p-1}(TK)$ to update the hash key from H_{HK_p} to $H_{HK_{p-1}}$. A collision r_2 can be computed in polynomial time as follows: $r_2 = r_1 \oplus m_1 \oplus m_2 \oplus c_p \oplus c_{p-1}$. Also, if r_1 is uniformly distributed in $\mathcal{R}$ then the distribution of r_2 is computationally indistinguishable from uniform in $\mathcal{R}$. ∎

Theorem 3 (k-times trapdoor collisions) The proposed chameleon hash function satisfies k-times trapdoor collisions if the hash function H_1 satisfies only one-wayness of a cryptographic secure hash function.

Proof. Assume to the contrary, there exists a polynomial time algorithm $\mathcal{A}_2$ that on input (HK_p, TK) (an initial p is set to k), a pair $(m_0, r_0) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$, and $(k+1)$ additional messages $\{m_1, \ldots, m_{k+1}\}$ outputs, with a probability which

Vol.2009-SE-164 No.7
Vol.2009-EMB-13 No.7
Vol.2009-CSEC-45 No.7
2009/5/28

Table 1 Comparison of efficiency

	Comp. cost of H_{HK}	Comp. cost of a collision	$\lvert H_{HK}\rvert$	Assumption
1), 2)	$exp.$	$mlt. + inv.$	$\lvert\mathbb{G}\rvert$	DLP
2)	$exp.$	$0.1mlt.$	$\lvert\mathbb{G}\rvert$	Factoring
3)	$2e + mlt. + h$	$2mlt. + 2h$	$\lvert\mathbb{G}\rvert$	Pairing
4)	$2e + exp. + mlt.$	$3exp. + mlt.$	$\lvert\mathbb{G}\rvert$	Pairing
5)	$2e + exp. + h$	$4exp. + 2mlt. + 2h$	$\lvert\mathbb{G}\rvert$	Pairing
6)	$exp. + h$	$2mlt. + inv + 2h$	$\lvert\mathbb{G}\rvert$	ECDLP
7)	$exp. + h$	$exp. + mlt.$	$\lvert\mathbb{G}\rvert$	Factoring
8)	$exp. + 2mlt.$	$2exp. + 2mlt. + inv$	$\lvert\mathbb{G}\rvert$	DLP
Ours	(Negligible)	$\frac{(k-1)}{2}h$	$\lvert H\rvert$	Hash

is not negligible, r_{i+1} $(i = 0, \ldots, k)$ that satisfy $m_0 \neq m_{i+1}$ and $H_{HK_k}(m_0, r_0) = H_{HK_{k-i-1}}(m_{i+1}, r_{i+1})$ $(i = 0, \ldots, k)$. Then, we can use $\mathcal{A}_2$ to break the one-wayness of H_1 as follows: For a randomly given instance a, define $TK = c_0 = a = H_1(b)$ and $HK_p = (H_1^k(a), H_1^p(a))$. Note that the value of b is not revealed. $\mathcal{A}_2$ can compute $(r_1, \ldots, r_k)$ by Theorem 2. Moreover, $\mathcal{A}_2$ outputs r_{k+1} that satisfy $m_0 \neq m_{k+1}$ and $H_{HK_k}(m_0, r_0) = H_{HK_{-1}}$ (m_{k+1}, r_{k+1}) such that $HK_{-1} = (c_k, c_{-1})$ and $c_0 = H_1(c_{-1})$. $\mathcal{A}_2$ is allowed to output c_{-1} because we do not know the value of $TK = c_0$. We can compute $b = c_{-1} = c_k \oplus m_0 \oplus m_{k+1} \oplus r_0 \oplus r_{k+1}$. Therefore, we can break the one-wayness of H_1. As for collision resistance, H_1 need not satisfy collision resistance to satisfy k-times trapdoor collisions in a similar fashion to Theorem 1. ∎

4.2 Efficiency

Given HK_p and (m, r), the proposed chameleon hash value $H_{HK_p}(m, r)$ is computable using hash function H_1 and XOR operation in polynomial time. Hence the proposed chameleon hash function satisfies efficiency in Definition 3.

We compare the efficiency of our scheme with that of related schemes (see Table 1). We denote by $mlt.$ "modulo multiplication", by $exp.$ "modulo exponential", by $inv.$ "modulo inverse", by e the "bilinear pairing", and by h the hash operation. Also, we denote by $|\mathbb{G}|$ the size of a group and by $|H|$ the output size of hash function.

In Table 1, the proposed chameleon hash function is much superior to the others in the computation cost of H_{HK} and the size of H_{HK} ($|H_{HK}|$). The computation cost of H_{HK} in our scheme is negligible, since the computation of H_{HK} requires only XOR operation. When the current hash key is (c_k, c_i) in our scheme, we have to compute c_{i-1} to generate a collision. So, the computation cost of a collision requires $(i - 1)$ times iteratively hashing c_0 $(1 \leq i \leq k)$, that is, on average $(k - 1)/2$ times hash operations. Hence the computation cost of a collision depends on the length of the hash chain. It can be, however, said that this influence is not necessarily large since it is more efficient than modulo operations such as $exp.$ or $inv.$ if the length of the hash chain is less than about 1000[13]. Therefore, the proposed chameleon hash function is comparatively efficient.

5. The proposed k-times on-line/off-line signature

In this section, we apply the "hash-sign-switch" paradigm[2] to propose an efficient on-line/off-line signature scheme based on the proposed chameleon hash function. On the other hand, the concept of k-times signature is introduced by Hwang[14], in which it is effective when the manager wants to restrict the number of signature generation to k times for each user.

The proposed chameleon hash function satisfies a property of k-times trapdoor collisions. The proposed on-line/off-line signature scheme innately has a feature of restricting the number of signature generation because it is based on the proposed chameleon hash function. So, our scheme is the k-times on-line/off-line signature scheme. We now introduce our method for combining the proposed trapdoor hash family $(\mathcal{I}, \mathcal{H})$ and any signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ to get an on-line/off-line signature scheme. For a security parameter λ, we construct an on-line/off-line k-times signature scheme $(\mathcal{G}', \mathcal{S}', \mathcal{V}')$.

5.1 An on-line/off-line signature scheme

The update procedure of the hash key in the proposed chameleon hash function is different from that in the existing scheme. We will adapt the proposed chameleon hash function to an existing generic construction[2].

- Key Generation Algorithm $(\mathcal{G}')$:
  - (1) Generate a pair (SK, VK) of signing key and verification key, by applying $\mathcal{G}$ to the input 1^λ (where $\mathcal{G}$ is the key generation algorithm of the original scheme).
  - (2) Generate a pair (HK_p, TK) of hash key and trapdoor key, by applying $\mathcal{I}$ to the input 1^λ (where $\mathcal{I}$ is the key generation algorithm of the

trapdoor hash family).

The signing key is (SK, HK_p, TK) and the verification key is (VK, HK_p).

- The Signing Algorithm ($\mathcal{S}'$): Given a signing key (SK, HK_p, TK), the signing algorithm operates as follows.
  - (1) Off-line phase:
    - Choose at random $(m', r') \in_R \mathcal{M} \times \mathcal{R}$, and compute $H_{HK_p}(m', r')$ using HK_p.
    - Run the signing algorithm $\mathcal{S}$ with the signing key SK to sign the message $H_{HK_p}(m', r')$. Denote the outputs $\mathcal{S}_{SK}(H_{HK_p}(m', r'))$ by Σ.
    - Store the pair (m', r'), the hash value $H_{HK_p}(m', r')$, and the signature Σ. (The hash value $H_{HK_p}(m', r')$ is stored only to avoid its recomputation in the on-line phase).
  - (2) On-line phase: Given a message m, the on-line phase proceeds as follows.
    - Retrieve from memory the pair (m', r'), the hash value $H_{HK_p}(m', r')$, and the signature Σ.
    - Find $r \in \mathcal{R}$ such that $H_{HK_{p-1}}(m, r) = H_{HK_p}(m', r')$
    - Send (r, Σ) as a signature of m.
- The Verification Algorithm: ($\mathcal{V}'$): To verify that the pair (r, Σ) is indeed a signature of the message m, with respect to the verification key (VK, HK_p), compute $H_{HK_{p-1}}(m, r)$ and use the verification algorithm $\mathcal{V}$ (of the original signature scheme) to check that Σ is indeed a signature fo the hash value $H_{HK_{p-1}}(m, r)$ with the verification key VK.

We now analyze the security of the resultant on-line/off-line signature scheme.

5.2 Security

The most general known security notion of a signature scheme is security against existential forgery on adaptively chosen message attacks. The scheme[2] defined the security notion about on-line/off-line signature based on chameleon hash function.

We can give a proof Theorem 4 in a similar way to the proof in the previous scheme[2] except for the update procedure of the hash key, because the update procedure of the proposed chameleon hash function is different from that in the previous scheme[2].

Theorem 4 Let ($\mathcal{G}$, $\mathcal{S}$, $\mathcal{V}$) be a signature scheme and let ($\mathcal{I}$, $\mathcal{H}$) be a trapdoor hash family. Let ($\mathcal{G}'$, $\mathcal{S}'$, $\mathcal{V}'$) be the resultant on-line/off-line signature scheme. Suppose that ($\mathcal{G}'$, $\mathcal{S}'$, $\mathcal{V}'$) is existentially forgeable by a Q-adaptive chosen message attack in time T with success probability ϵ. Then one of the following cases holds:

(1) There exists a probabilistic algorithm that, given a hash key HK_p, finds collisions of H_{HK_p} in time $T + T_{\mathcal{G}} + Q(T_{\mathcal{H}} + T_{\mathcal{S}})$ with success probability $\geq \frac{\epsilon}{2}$ (where T_G is the running time of $\mathcal{G}$, $T_{\mathcal{H}}$ is the running time required to compute functions in $\mathcal{H}$, and $T_{\mathcal{S}}$ is the running time of $\mathcal{S}$).

(2) The original signature scheme ($\mathcal{G}$, $\mathcal{S}$, $\mathcal{V}$) is existentially forgeable by a generic Q-chosen message attack in time $T + Q(T_{\mathcal{H}} + T_{COL}) + T_{\mathcal{I}}$ with success probability $\geq \frac{\epsilon}{2}$ (where T_{COL} is the time required to find collisions of the trapdoor hash function given the hash key and the trapdoor key, and $T_{\mathcal{I}}$ is the running time of algorithm $\mathcal{I}$).

Proof. Suppose that $\mathcal{F}'$ is a probabilistic algorithm that, given a verification key (HK, VK), forges a signature with respect to the signature scheme ($\mathcal{G}'$, $\mathcal{S}'$, $\mathcal{V}'$) by a Q-chosen message attack in time T with success probability ϵ. Let $\{m_i\}_{i=1}^{Q}$ denote the Q queries that the forger $\mathcal{F}'$ sends to the signing oracle, and let $\{(r_i, \Sigma_i)\}_{i=1}^{Q}$ denote the corresponding signatures produced by the oracle. Let m, (r, Σ) denote the output of $\mathcal{F}'$. We can give a proof of case 1 in a similar fashion to that in the existing scheme[2]. We only give the difference of proof in case 2 described below. If case 2 holds, we define a probabilistic algorithm $\mathcal{F}$ that forges a signature with respect to $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ by a generic Q-chosen message attack like that in the scheme[2]. The main difference between the previous proof and this proof is the hash update procedure. This hash update is used when $\mathcal{F}$ simulates the forger $\mathcal{F}'$ on input (VK, HK). Hence we have only to show that $\mathcal{F}$ can simulate the forger $\mathcal{F}'$ on input (VK, HK). When $\mathcal{F}'$ queries the oracle with a message m_i, $\mathcal{F}$ finds $r_i \in \mathcal{R}$ such that $H_{HK_{p-1}}(m_i, r_i) = H_{HK_p}(m_i', r_i')$ and proceeds as if the signature obtained by the signing oracles S' was (r_i, Σ_i). We underline that $\mathcal{F}$ can calculate the next hash key HK_{p-1} from the current hash key H_{HK_p} since $\mathcal{F}$ knows the trapdoor key TK. The rest of the proof can be done in a similar fashion to that in scheme[2]. ∎

5.3 Efficiency

The proposed chameleon hash function is much more efficient than existing schemes because there are no modulo operations. Our on-line/off-line signature scheme become consequently more efficient. The on-line phase becomes especially much more efficient.

6. Conclusion

In this paper, we firstly introduce a lightweight chameleon hash function using only hash function and XOR operations, which has a property of k-times trapdoor collisions to restrict the calculation frequency of the collisions to k times. Compared with other chameleon hash functions, the advantages of our scheme are the lower computation cost of hashing, and the small output size of chameleon hash function. We also apply the "hash-sign-switch" paradigm to propose a much more efficient generic on-line/off-line signature scheme based on our chameleon hash function.

References

1) H.Krawczyk and T.Rabin, "Chameleon signatures," Proc. 7th Network and Distributed System Security – NDSS'00, pp.143–154, 2000.
2) A.Shamir and Y.Tauman, "Improved online/offline signature schemes," Advances in Cryptology – CRYPTO'01, LNCS 2139, pp.355–367, Springer-Verlag, 2001.
3) F.Zhang, R.S.Naini, W.Susilo, "ID-Based Chameleon Hashes from Bilinear Pairings," Cryptology ePrint Archive: Report 2003/208, 2004 (revised).
4) C. Ma, J. Ao, J. Li, "Chameleon-Based Deniable Authenticated Key Agreement Protocol Secure Against Forgery," Proc. 2nd Online Communities and Social Computing – OCSC'07, LNCS 4564, pp.124–133, Springer-Verlag, 2007.
5) Q. Ren, Y. Mu, W. Susilo, "Mitigating phishing with ID-based online/offline authentication," Proc. 6th Australasian conference on Information security – AISC'08, pp.59–64, ACM, 2008.
6) X. Chen, F. Zhang, W. Susilo and Y. Mu, "Efficient generic on-line/off-line signatures without key exposure," Proc. Applied Cryptography and Network Security – ACNS'07, LNCS 4521, pp.18–30, Springer-Verlag, 2007.
7) G. Ateniese, B. Medeiros, "Identity-Based Chameleon Hash and Applications," Proc. 8th Financial Cryptography – FC'04, LNCS 3110, pp.164–180, Springer-Verlag, 2004.
8) X.Chen, F.Zhang and K.Kim, "Chameleon hashing without key exposure," Proc. 7th Information Security Conference – ISC'04, LNCS 3225, pp.87–98, Springer-Verlag, 2004.
9) G. Ateniese, "On the Key Exposure Problem in Chameleon Hashes," Proc. 4th Security in Communication Networks – SCN'04, LNCS 3352, pp.165–179, Springer-Verlag, 2005.
10) NIST, "Secure hash standard," National Institute for Standards and Technology, Gaithersburg, MD, USA, 1995.
11) L.Lamport, "Password authentication with insecure communication," Communications of the ACM, vol.24, no.11, pp.770–772, November 1981.
12) K. Omote, "Consideration of Lightweight Chameleon Hash Function," SCIS'09, 2009.
13) W.Dai, "Speed Comparison of Popular Crypto Algorithms," http://www.cryptopp.com/benchmarks.html, 2007.
14) J.Y.Hwang, H.J.Kim, D.H.Lee and J.Lim, "Digital signature schemes with restriction on signing capability," Proc. Information Security and Privacy – ACISP'03, LNCS 2727, pp.324–335, Springer-Verlag, 2003.