A Lightweight Hash Function Resisting Birthday Attack and Meet-in-the-middle Attack*

Shenghui Su [1, 2], Tao Xie [3], Shuwang Lü [4]

[1] College of Computers, Beijing University of Technology
[2] College of Information Engineering, Yangzhou University
[3] School of Computers, National University of Defense Technology
[4] Graduate School, Chinese Academy of Sciences

Abstract: In this paper, to match a lightweight digital signing scheme of which the length of modulus is between 80 and 160 bits, a lightweight hash function is proposed. It is based on MPP and ASPP intractabilities, and regards a short message or a message digest as an input which is treated as only one block. The lightweight hash function contains two algorithms: an initialization algorithm and a compression algorithm, and converts a string of n bits into another of m bits, where $80 \le m \le n \le 4096$. The two algorithms are described, and their securities are analyzed from several aspects. The analysis shows that the lightweight hash is one-way, weakly collision-free, strongly collision-free along a proof, and especially resistant to birthday attack and meet-in-the-middle attack with the concrete security of $O(2^m)$ at present, and its compression algorithm has the time complexity of $O(nm^2)$ bit operations. Moreover, the lightweight hash with short input and small computation may be used to reform a classical hash with output of n bits and security of $O(2^{n/2})$ into a compact hash with output of $n/2$ bits and the equivalent security. Thus, it opens a door to convenience for utilization of lightweight digital signing schemes.

Keywords: Bit long-shadow; Lightweight hash function; Compression algorithm; Birthday attack; Multivariate permutation problem; Anomalous subset product problem

1 Introduction

In recent years, the ECC-160 digital signing scheme, an analogue of the ElGamal public key cryptosystem based on the discrete logarithm problem (DLP) in an ellipse curve group over a finite field [1][2], and some lightweight digital signing schemes are utilized for RFID (Radio-Frequency Identity) tags or non-RFID (non-Radio-Frequency Identity) tags [3][4][5]. A RFID tag contains an IC chip which is used to store signatures and other data, but an non-RFID tag contains no IC chip because a short signature from a lightweight or ultra-lightweight signing scheme may be symbolized in short length, and printed directly on the paper of a tag. Now, such tags are applied to identification, authentication, or anti-forgery of financial-notes, certificates, diplomas, and commodities, particularly including food and drug.

It is well understood that we first need to extract the digest of a message by employing a hash function before signing the message. A hash function ordinarily consists of a compression function and the Merkle-Damgård iterative structure [6][7]. Let $\hat{h}$ be a hash function, and generally, it has the following properties [8][9]:

① given a message w, it is very easy to calculate the message digest $d = \hat{h}(w)$, where d is also called a hash output;

② given any digest d, it is very hard to calculate the message w according to $d = \hat{h}(w)$, namely $\hat{h}$ is one-way;

③ given any message w, it is computationally infeasible to find another message w' such that $\hat{h}(w) = \hat{h}(w')$, namely $\hat{h}$ is weakly collision-free;

④ it is computationally infeasible to find any two distinct messages w and w' such that $\hat{h}(w) = \hat{h}(w')$, namely $\hat{h}$ is strongly collision-free.

The word "infeasible" means that some problem cannot be solved at least in polynomial time. Sometimes, ④ is optional with some users of a hash function because ①, ②, and ③ are enough for most of applications of the users.

At present, SHA-1, SHA-256, and SHA-384 announced by NIST are among the hash functions which are believed to be secure [8][10], though cannot resist birthday attack of which the time complexity is approximately $O(2^{m/2})$, where m is the bit-length of a message digest, namely a hash

* This work is supported by MOST with Project 2007CB311100 and 2009AA01Z441.

output. The bit-lengths of outputs of these functions are 160, 256, and 384 respectively. When any of the three is matched practically with a lightweight signing scheme of which the bit-length of modulus is between 80 and 160, the bit-length of output of the hash function must be adapted to the range between the bit-length of security and the bit-length of modulus of the singing scheme, where the bit-length of security represents the security of the signing scheme by the bit. Take the ECC-160 scheme, its security is 2^{80}, namely the best algorithm for cracking ECC-160 will need 2^{80} operation steps currently, and hence, the bit-length of its security is 80.

Assume that the bit-length of security and the bit-length of modulus of a lightweight signing scheme are both 80. When SHA-1 and the lightweight signing scheme are paired, the bit-length of output of SHA-1 must be compressed to 80 while the security of it remains unchanged. Again when SHA-256 and ECC-160 are paired, the bit-length of output of SHA-256 must be compressed to the range from 80 to 160 while the security of it should be at least 2^{80}. Notice that owing to birthday attack, the securities of SHA-1 and SHA-256 are commonly thought to be 2^{80} and 2^{128} separately, namely the bit-lengths of securities of the two functions are 80 and 128 separately.

Therefore, it is a problem how we compress a short message or a message digest from a classical hash function securely in order that we can employ a lightweight signing scheme in practice. We will discuss the problem in this paper.

In Section 2 of the paper, several relevant definitions are given. In Section 3, the two algorithms of a lightweight hash function called JUNA are described. In Section 4, the security of the lightweight hash function is analyzed. In Section 5, the time complexity of the compression algorithm is dissected. In Section 6, the reformation of a classical hash function is illustrated.

The paper has two dominant novelties: ① designing an initialization algorithm which makes the lightweight hash be capable of resisting birthday attack; ② designing a compression algorithm due to which the lightweight hash can resist existent various attacks, especially meet-in-the-middle attack. The significance of the paper lies in the thing that a lightweight hash function of which the bit-length of output and the bit-length of security may equal each other is first proposed by the authors while the bit-length of output of a classical hash function is double the bit-length of security of it, namely when the bit-length of output of the lightweight hash function is m, its security is also up to $O(2^m)$, but not $O(2^{m/2})$.

Throughout the paper, unless otherwise specified, an even number $n \ge 80$ is the bit-length of a short message (or a message digest) or the item-length of a sequence, the sign % denotes "modulo", $\bar{M}$ does "$M-1$" with M prime, $\lg x$ denotes a logarithm of x to the base 2, $\neg b_i$ does NOT operation of a bit b_i, $\bar{P}$ does the maximal prime allowed in coprime sequences, $|x|$ does the absolute value of a number x, $\|x\|$ does the order of x % M, $¦S¦$ does the size of a set S, and $\gcd(x, y)$ represents the greatest common divisor of two integers x and y. Without ambiguity, "% M" is usually omitted in expressions.

2 Several Definitions

Before the two algorithms of a lightweight hash function are described, three important definitions should be presented, although they are already given in [11].

2.1 A Coprime Sequence

Definition 1: If $A_1, \ldots, A_n$ are n pairwise distinct positive integers such that $\forall A_i, A_j$ $(i \ne j)$, either $\gcd(A_i, A_j) = 1$ or $\gcd(A_i, A_j) = F \ne 1$ with $(A_i / F) \nmid A_k$ and $(A_j / F) \nmid A_k$ $\forall k \ne i, j \in [1, n]$, these integers are called a coprime sequence, denoted by $\{A_1, \ldots, A_n\}$, and shortly $\{A_i\}$.

Notice that the elements of a coprime sequence are not necessarily pairwise coprime, but a sequence whose elements are pairwise coprime is a coprime sequence.

Property 1: Let $\{A_1, \ldots, A_n\}$ be a coprime sequence. If we randomly select $m \in [1, n]$ elements from $\{A_1, \ldots, A_n\}$, and construct a subset $\{A_{x_1}, \ldots, A_{x_m}\}$, the subset product $G = \prod_{i=1}^{m} A_{x_i} = A_{x_1} \ldots A_{x_m}$ is uniquely determined, namely the mapping from $\{A_{x_1}, \ldots, A_{x_m}\}$ to G is one-to-one.

Refer to [11] for its proof.

2.2 A Bit Shadow and a Bit Long-Shadow

Definition 2: Let $b_1 \ldots b_n \ne 0$ be a bit string. Then $\underline{b}_i$ with $i \in [1, n]$ is called a bit shadow if it comes from such a rule: (1) $\underline{b}_i = 0$ if $b_i = 0$; (2) $\underline{b}_i = 1 +$ the number of successive 0-bits before b_i if $b_i = 1$; or (3)

$\underline{b}_i$ = 1 + the number of successive 0-bits before b_i + the number of successive 0-bits after the rightmost 1-bit if b_i is the leftmost 1-bit.

Notice that (3) of this definition is slightly different from that in [11].

Fact 1: Let $\underline{b}_1 \ldots \underline{b}_n$ be the bit shadow string of $b_1 \ldots b_n \neq 0$. Then there is $\sum_{i=1}^{n} \underline{b}_i = n$.

Proof.

According to Definition 2, every bit of $b_1 \ldots b_n$ is considered into $\sum_{i=1}^{k} \underline{b}_{x_i}$, where $\underline{b}_{x_1}, \ldots, \underline{b}_{x_k}$ are 1-bit shadows in the string $\underline{b}_1 \ldots \underline{b}_n$, and there is $\sum_{i=1}^{k} \underline{b}_{x_i} = n$.

On the other hand, there is $\sum_{j=1}^{n-k} \underline{b}_{y_j} = 0$, where $\underline{b}_{y_1}, \ldots, \underline{b}_{y_{n-k}}$ are 0-bit shadows.

In total, there is $\sum_{i=1}^{n} \underline{b}_i = n$. □

Property 2: Let $\{A_1, \ldots, A_n\}$ be a coprime sequence, and $\underline{b}_1 \ldots \underline{b}_n$ be the bit shadow string of $b_1 \ldots b_n \neq 0$. Then the mapping from $b_1 \ldots b_n$ to $G = \prod_{i=1}^{n} A_i^{\underline{b}_i}$ is one-to-one.

Proof.

Firstly, let $b_1 \ldots b_n$ and $b'_1 \ldots b'_n$ be two different nonzero bit strings, and $\underline{b}_1 \ldots \underline{b}_n$ and $\underline{b}'_1 \ldots \underline{b}'_n$ be the two corresponding bit shadow strings.

If $\underline{b}_1 \ldots \underline{b}_n = \underline{b}'_1 \ldots \underline{b}'_n$, then by Definition 2, there is $b_1 \ldots b_n = b'_1 \ldots b'_n$.

In addition, for any arbitrary bit shadow $\underline{b}_1 \ldots \underline{b}_n$, there always exists a preimage $b_1 \ldots b_n$. Thus, the mapping from $b_1 \ldots b_n$ to $\underline{b}_1 \ldots \underline{b}_n$ is one-to-one.

Secondly, obviously the mapping from $\underline{b}_1 \ldots \underline{b}_n$ to $\prod_{i=1}^{n} A_i^{\underline{b}_i}$ is surjective.

Presuppose that $\prod_{i=1}^{n} A_i^{\underline{b}_i} = \prod_{i=1}^{n} A_i^{\underline{b}'_i}$ for $\underline{b}_1 \ldots \underline{b}_n \neq \underline{b}'_1 \ldots \underline{b}'_n$.

Since $\{A_1, \ldots, A_n\}$ is a coprime sequence, and $A_i^{\underline{b}_i}$ either equals 1 with $\underline{b}_i = 0$ or contains the same prime factors as those of A_i with $\underline{b}_i \neq 0$, we can obtain $\underline{b}_1 \ldots \underline{b}_n = \underline{b}'_1 \ldots \underline{b}'_n$ from $\prod_{i=1}^{n} A_i^{\underline{b}_i} = \prod_{i=1}^{n} A_i^{\underline{b}'_i}$, which is in direct contradiction to $\underline{b}_1 \ldots \underline{b}_n \neq \underline{b}'_1 \ldots \underline{b}'_n$.

Therefore, the mapping from $\underline{b}_1 \ldots \underline{b}_n$ to $\prod_{i=1}^{n} A_i^{\underline{b}_i}$ is injective [12].

In summary, the mapping from $\underline{b}_1 \ldots \underline{b}_n$ to $\prod_{i=1}^{n} A_i^{\underline{b}_i}$ is one-to-one, and further the mapping from $b_1 \ldots b_n$ to $\prod_{i=1}^{n} A_i^{\underline{b}_i}$ is also one-to-one. □

Definition 3: Let $\underline{b}_1 \ldots \underline{b}_n$ be a bit shadow string of $b_1 \ldots b_n \neq 0$. Then $\hat{b}_i = \underline{b}_i 2^{\partial_i}$ with $i \in [1, n]$ is called a bit long-shadow, where $\partial_i = b_{i+(-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)} = 0$ or 1.

Fact 2: Let $\hat{b}_1 \ldots \hat{b}_n$ be a bit long-shadow string of $b_1 \ldots b_n \neq 0$. Then there is $n \le \sum_{i=1}^{n} \hat{b}_i \le 2n$.

Proof.

By Definition 3 and Fact 1, we have

$$\sum_{i=1}^{n} \hat{b}_i = \sum_{i=1}^{n} \underline{b}_i 2^{\partial_i} \text{ and } \sum_{i=1}^{n} \underline{b}_i = n.$$

If every $b_i = 1$, namely every $\partial_i = 1$, then

$$\sum_{i=1}^{n} \hat{b}_i = \sum_{i=1}^{n} \underline{b}_i 2^{\partial_i} = 2\sum_{i=1}^{n} \underline{b}_i = 2n.$$

Again, by Definition 3, not every bit of $b_1 \ldots b_n$ is zero.

If there exists only a nonzero bit in $b_1 \ldots b_n$ — $b_x = 1$ with $x \in [1, n]$ for example, then

$$\sum_{i=1}^{n} \hat{b}_i = \sum_{i=1}^{n} \underline{b}_i 2^{\partial_i} = \underline{b}_x 2^{\partial_x} = \underline{b}_x = n,$$

where $\partial_x = b_{x+(-1)^{\lfloor 2(x-1)/n \rfloor}(n/2)} = 0$ due to b_x being the unique nonzero bit.

Thus, it holds that $n \le \sum_{i=1}^{n} \hat{b}_i \le 2n$. □

Property 3: Let $\hat{b}_1 \ldots \hat{b}_n$ be a bit long-shadow string of $b_1 \ldots b_n \neq 0$. Then the mapping from $b_1 \ldots b_n$ to $\hat{b}_1 \ldots \hat{b}_n$ is one-to-one.

Proof.

On one hand, assume that $b_1 \ldots b_n \neq 0$ is known.

It is known from Definition 3 that $\hat{b}_i = \underline{b}_i 2^{\partial_i}$ for each i, where $\partial_i = b_{i+(-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)}$.

Because when $b_1 \ldots b_n$ is known, $\underline{b}_1 \ldots \underline{b}_n$ and $\partial_1 \ldots \partial_n$ are respectively determined, $\hat{b}_1 \ldots \hat{b}_n$ can also be determined uniquely.

On the other hand, assume that $\hat{b}_1 \ldots \hat{b}_n$ is known.

According to $\hat{b}_i = \underline{b}_i 2^{\partial_i}$ and $\hat{b}_i = 0$ with $\underline{b}_i = 0$, where $\partial_i = b_{i+(-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)}$, we can determinate b_i for i = 1, …, n as follows.

① Case of $\hat{b}_i = 0$

If $\hat{b}_i = 0$, then $\underline{b}_i = 0$, and set $b_i = 0$.

② Case of $\hat{b}_i \neq 0$

If $\hat{b}_i \neq 0$, then $\underline{b}_i \neq 0$, and set $b_i = 1$.

In this way, the value of every b_i can be determined uniquely.

In summary, the mapping from $b_1 \ldots b_n$ to $\acute{b}_1 \ldots \acute{b}_n$ is one-to-one. □

2.3 A Lever Function

The coming hash function consists of the two algorithms: initialization algorithm and compression algorithm, and employs the concepts of a private key and a public key.

In the initialization algorithm of the new hash function, $C_i \equiv (A_i W^{\ell(i)})^{\delta}$ (% M) for $i = 1, \ldots, n$ is a key transform from a private key to a public key, where $\ell(i)$ is an exponent.

***Definition 4**:* The secret parameter $\ell(i)$ in the key transform of a public key cryptosystem or a hash function over the prime field $\mathbb{GF}(M)$ is called a lever function, if it has the following features:

① $\ell(.)$ is an injection from the domain $\{1, \ldots, n\}$ to the codomain $\Omega \subset \{1, \ldots, \bar{M}\}$;

② the mapping between i and $\ell(i)$ is established randomly without an analytical expression;

③ an attacker has to be faced with all the permutations of elements in Ω when inferring a related private key from a public key;

④ the owner of the private key only need to considers the accumulative sum of elements in Ω when recovering a related plaintext from a ciphertext through the cryptosystem.

Feature ③ and ④ make it clear that if n is large enough, it is infeasible for the attacker to search all the permutations of elements in Ω exhaustively while the decryption of a normal ciphertext is feasible in time being polynomial in n. Thus, the amount of calculation on $\ell(.)$ at "a public terminal" is large, and the amount of calculation on $\ell(.)$ at "a private terminal" is small.

Notice that in the REESSE1+ cryptosystem [11], a public key is used for encryption, and a private key is used for decryption.

***Property 4* (Indeterminacy of $\ell(.)$):** Let $\delta = 1$ and $C_i \equiv A_i W^{\ell(i)}$ (% M) for $i = 1, \ldots, n$, where $\ell(i) \in \Omega = \{5, \ldots, n + 4\}$ and $A_i \in \Lambda = \{2, \ldots, \bar{P}\}$. Then $\forall\, W \in [1, \bar{M}]$ with $\|W\| \neq \bar{M}$, and $\forall\, x, y, z \in [1, n]$ with $z \neq x, y$,

① when $\ell(x) + \ell(y) = \ell(z)$, there is

$$\ell(x) + \|W\| + \ell(y) + \|W\| \neq \ell(z) + \|W\| \text{ (\% } \bar{M});$$

② when $\ell(x) + \ell(y) \neq \ell(z)$, there always exist

$$C_x \equiv A'_x W'^{\,\ell'(x)},\ C_y \equiv A'_y W'^{\,\ell'(y)}, \text{ and } C_z \equiv A'_z W'^{\,\ell'(z)} \text{ (\% } M)$$

such that $\ell'(x) + \ell'(y) \equiv \ell'(z)$ (% $\bar{M}$) with $A'_z \leq \bar{P}$.

Proof.

① It is easy to understand that

$$W^{\ell(x)} \equiv W^{\ell(x)+\|W\|},\ W^{\ell(y)} \equiv W^{\ell(y)+\|W\|}, \text{ and } W^{\ell(z)} \equiv W^{\ell(z)+\|W\|} \text{ (\% } M).$$

Due to $\|W\| \neq \bar{M}$, $2\|W\| \neq \|W\|$, and $\ell(x) + \ell(y) = \ell(z)$, it follows that

$$\ell(x) + \|W\| + \ell(y) + \|W\| \neq \ell(z) + \|W\| \text{ (\% } \bar{M}).$$

However, it should be noted that when $\|W\| = \bar{M}$, there is $\ell(x) + \|W\| + \ell(y) + \|W\| \equiv \ell(z) + \|W\|$ (% $\bar{M}$).

② Let $\bar{O}_{\text{d}}$ be an oracle on solving a discrete logarithm problem.

Suppose that $W' \in [1, \bar{M}]$ is a generator of $(\mathbb{Z}^*_M, \cdot)$.

In light of group theories, $\forall\, A'_z \in \{2, \ldots, \bar{P}\}$, the congruence

$$C_z \equiv A'_z W'^{\,\ell'(z)} \text{ (\% } M)$$

has a solution. Then, $\ell'(z)$ may be taken through $\bar{O}_{\text{d}}$.

$\forall\, \ell'(x) \in [1, \bar{M}]$, and let $\ell'(y) \equiv \ell'(z) - \ell'(x)$ (% $\bar{M}$).

Further, from the congruences $C_x \equiv A'_x W'^{\,\ell'(x)}$ (% M) and $C_y \equiv A'_y W'^{\,\ell'(y)}$ (% M), we can obtain many distinct pairs (A'_x, A'_y), where $A'_x, A'_y \in (1, M)$, and $\ell'(x) + \ell'(y) \equiv \ell'(z)$ (% $\bar{M}$).

In this way, Property 4.2 is proven. □

Notice that letting $\Omega = \{5, \ldots, n + 4\}$, namely every $\ell(i) \geq 5$ makes seeking W from $W^{\ell(i)} \equiv A_i^{-1} C_i$ (% M) face an unsolvable Galois group when A_i is guessed [13], and especially when Ω is any subset containing n elements of $\{1, \ldots, \bar{M}\}$, Property 4 still holds.

Property 4 manifests that continued fraction attack on $C_i \equiv A_i W^{\ell(i)}$ (% M) by theorem 12.19 in Section 12.3 of [14] will be utterly ineffectual as long as Ω are fitly selected [15].

3 Design of a Lightweight Hash Function

Assume that the bit-length of modulus of a lightweight signing scheme is m, the bit-length of a short message or a message digest from a classical hash function is n, and there is $80 \leq m \leq n \leq 4096$.

There does not exist the unified or standard definition of a lightweight hash function, and therefore, we say that a hash function is regarded as lightweight if it has short input, short output, and small computation simultaneously.

For example, the Chaum-van Heijst-Pfitzmann hash function, which is based on a discrete logarithm problem, and believed to be strongly collision-free presently (however, should be nonresistant to birthday attack because the security will be less than $2^{\lceil \lg p \rceil / 2}$), may be regarded as lightweight [16]. It is defined as follows:

$$\hat{h}: w_1, w_2 \mapsto \hat{h}(w_1, w_2) = \alpha^{w_1} \beta^{w_2} \,\%\, p \quad (\{0, ..., q-1\}^2 \to \mathbb{Z}_p - \{0\}),$$

where w_1 and w_2 are the two complementary parts of a short message, p and $q = (p-1)/2$ are two big primes, and α and β are two primitives of $\mathbb{Z}_p$. Evidently, it is difficult to know the value of $\log_\alpha \beta$.

The JUNA lightweight hash function contains two algorithms of which the securities are expected to be each $O(2^m)$ magnitude.

3.1 Initialization Algorithm

Let $\Lambda' = \{2, 3, \ldots, \mathcal{P} \mid \mathcal{P} \geq 2^{18}\}$.

Again let $\Omega' \subset \{\pm 5, \pm 7, \ldots, \pm(2n+3)\}$ and subjected to $x + y \neq 0 \; \forall\, x, y \in \Omega'$ with $|\Omega'| = n$, where "$\pm x$" means the coexistence of "$+x$" and "$-x$", which indicates that Ω' is one of 2^n potential sets.

This algorithm is employed by an authoritative third party or a digital signer, and only needs to be executed one time.

S1: Randomly generate a coprime sequence $\{A_1, \ldots, A_n\}$ with $A_i \in \Lambda'$.

S2: Find a prime M with $\lceil \lg M \rceil = m$ such that

$\gcd(q, \bar{M}/2) = 1 \; \forall q \in [1, 8n(n+1)]$.

S3: Pick $W, \delta \in (1, \bar{M})$ making $\gcd(\delta, \bar{M}) = 1$, $\|W\| \geq 2^{m-18}$.

S4: Randomly produce pairwise distinct $\ell(1), \ldots, \ell(n) \in \Omega'$.

S5: Compute $C_i \leftarrow (A_i W^{\ell(i)})^\delta \,\%\, M$ for $i = 1, \ldots, n$.

At last, $(\{C_i\}, M)$ is regarded as the initial value of a compression algorithm, and public to people. The private key $(\{A_i\}, \{\ell(i)\}, W, \delta)$ may be discarded, but must not be divulged.

By Definition 3, if there exists only a nonzero bit in $b_1 \ldots b_n$, there is $\sum_{i=1}^n \hat{b}_i = n$. If there exists only two nonzero bits — $b_x = b_y = 1$ with $x, y \in [1, n]$ and $y = x + n/2$ for example, there are

$$\textstyle\sum_{i=1}^n \hat{b}_i = \hat{b}_x + \hat{b}_y = \underline{b}_x 2^{\partial_x} + \underline{b}_y 2^{\partial_y} = 2(\underline{b}_x + \underline{b}_y) = 2n$$

and

$$\begin{aligned}\textstyle\sum_{i=1}^n \hat{b}_i \ell(i) &= \hat{b}_x \ell(x) + \hat{b}_y \ell(y) \leq \hat{b}_x(2n+3) + \hat{b}_y(2n-1) \\ &< (\hat{b}_x + \hat{b}_y)(2n+3) = 2n(2n+3).\end{aligned}$$

Hence at S2, the product $8n(n+1)$ is obtained (see Section 4.4.3) according to

$$\underline{k} - \underline{k}' = \textstyle\sum_{i=1}^n \hat{b}_i \ell(i) - \sum_{i=1}^n \hat{b}'_i \ell(i) < 2n(2n+3) + 2n(2n+1) = 8n(n+1).$$

Definition 5: Given the sequence $\{C_i\}$ and the prime M, seeking the original $\{A_i\}$, $\{\ell(i)\}$, W, δ from $C_i \equiv (A_i W^{\ell(i)})^\delta$ ($\%\, M$) with $A_i \in \{2, 3, \ldots, \mathcal{P} \mid \mathcal{P} \geq 2^{18}\}$ and $\ell(i) \in \{\pm 5, \pm 7, \ldots, \pm(2n+3)\}$ for $i = 1, \ldots, n$ (but $\ell(x) + \ell(y) \neq 0 \; \forall\, x, y \in [1, n]$) is referred as the multivariate permutation problem, shortly MPP.

Property 5: The MPP $C_i \equiv (A_i W^{\ell(i)})^\delta$ ($\%\, M$) with $A_i \in \{2, 3, \ldots, \mathcal{P} \mid \mathcal{P} \geq 2^{18}\}$ and $\ell(i) \in \{\pm 5, \pm 7, \ldots, \pm(2n+3)\}$ for $i = 1, \ldots, n$ (but $\ell(x) + \ell(y) \neq 0 \; \forall\, x, y \in [1, n]$) is computationally at least equivalent to DLP in the same prime field.

See Section 4.1 for its proof.

3.2 Compression Algorithm

Let $b_1 \ldots b_n \neq 0$ be a short message or a message digest from a classical hash function $\hat{h}$.

Assume that $(\{C_1, \ldots, C_n\}, M)$ is an initial value, where M is a prime whose bit-length is m with $80 \leq m \leq n \leq 4096$.

S1: Set $k \leftarrow 0$, $i \leftarrow 1$.

S2: If $b_i = 0$,

  S2.1: let $k \leftarrow k + 1$, $\underline{b}_i \leftarrow 0$;

 else

  S2.2: if $i = k + 1$, let $s \leftarrow i$;

  S2.3: let $\underline{b}_i \leftarrow k + 1$, $k \leftarrow 0$.

S3: Let $i \leftarrow i + 1$.

If $i \le n$, go to S2.

S4: Compute $\underline{b}_s \leftarrow \underline{b}_s + k$.

S5: Compute $\underline{d} \leftarrow \prod_{i=1}^{n} C_i^{\hat{b}_i} \% M$,

where $\hat{b}_i = \underline{b}_i 2^{\partial_i}$ with $\partial_i = b_{i + (-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)}$.

So, the digest $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ ($\% M$) of m bits is obtained.

It is not difficult to understand that there exists $max(\hat{b}_1, \ldots, \hat{b}_n) \le n$ since $b_1 \ldots b_n$ is a nonzero bit string, and there is $\hat{b}_i = \underline{b}_i 2^{\partial_i}$ with $\partial_i = b_{i+(-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)} = 0$ or 1.

Definition 6: Given the digest $\underline{d}$ and the prime M, seeking the original $\hat{b}_1 \ldots \hat{b}_n$ from $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ ($\% M$), where $\hat{b}_i = \underline{b}_i 2^{\partial_i}$ with $\partial_i = b_{i + (-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)}$ and $\underline{b}_i$ being a bit shadow is referred as the anomalous subset product problem, shortly ASPP.

Property 6: The ASPP $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ ($\% M$), where $\hat{b}_i = \underline{b}_i 2^{\partial_i}$ with $\partial_i = b_{i+(-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)}$ and $\underline{b}_i$ being a bit shadow is computationally at least equivalent to DLP in the same prime field.

See Section 4.3 for its proof.

4 Security Analysis of the Lightweight Hash Function

Because a hash function must be one-way, weakly collision-free, and sometimes required to be strongly collision-free, the lightweight hash function of a round of iteration should also be at least one-way and weakly collision-free.

It is should be noted that $\lceil \lg M \rceil = m$, but not n, is the security dominant parameter of the lightweight hash function.

Definition 7: Let A and B be two computational problems. A is said to reduce to B in polynomial time, written as $A \le_{\mathrm{T}}^{\mathrm{P}} B$, if there is an algorithm for solving A which calls, as a subroutine, a hypothetical algorithm for solving B, and runs in polynomial time, excluding the time of the algorithm for solving B [8][17].

The hypothetical algorithm for solving B is called an oracle. It is easy to understand that no matter what the running time of the oracle is, it does not influence the result of the comparison.

$A \le_{\mathrm{T}}^{\mathrm{P}} B$ means that the difficulty of A is not greater than that of B, namely the running time of the fastest algorithm for solving A is not greater than that of the fastest algorithm for solving B when all polynomial times are treated as being pairwise equivalent. Concretely speaking, if A cannot be solved in polynomial or subexponential time, correspondingly B cannot also be solved in polynomial or subexponential time; and if B can be solved in polynomial or subexponential time, correspondingly A can also be solved in polynomial or subexponential time.

Obviously, Definition 7 gives a partial order relation among the complexities or hardnesses of problems [18].

In addition, for convenience sake, let $\hat{H}(y = f(x))$ represent the complexity or hardness of solving a problem $y = f(x)$ for x [19].

4.1 Proof of Property 5 on MPP

In Section 3.1, MPP is defined as $C_i \equiv (A_i W^{\ell(i)})^{\delta}$ ($\% M$) with $A_i \in \Lambda' = \{2, 3, \ldots, Ᵽ \mid Ᵽ \ge 2^{18}\}$ and $\ell(i) \in \Omega' \subset \{\pm 5, \pm 7, \ldots, \pm(2n+3)\}$ for $i = 1, \ldots, n$. Considering that $\Omega' \subset \{\pm 5, \pm 7, \ldots, \pm(2n+3)\}$ is indeterminate and different from $\Omega = \{5, 7, \ldots, 2n+3\}$ in [11], and the value of $Ᵽ$ is larger than the old one in [11], we specially give the proof of property 5.

Proof.

Firstly, systematically consider $C_i \equiv (A_i W^{\ell(i)})^{\delta}$ ($\% M$) for $i = 1, \ldots, n$.

Assume that each $g_i \equiv A_i W^{\ell(i)}$ ($\% M$) is a constant, where $\ell(i) \in \{\pm 5, \pm 7, \ldots, \pm(2n+3)\}$ with $\ell(x) + \ell(y) \ne 0 \; \forall \, x, y \in [1, n]$.

Let

$$g_i \equiv g^{x_i} \; (\% M), \text{ and } z_i \equiv \delta x_i \; (\% \bar{M}),$$

where $g \in \mathbb{Z}_M^*$ be a generator.

Then, there is

$$C_i \equiv g_i^{\delta} \equiv g^{\delta x_i} \; (\% M) \text{ for } i = 1, \ldots, n.$$

Again let $\delta x_i \equiv z_i$ ($\% \bar{M}$). Then

$$C_i \equiv g^{z_i} \; (\% M) \text{ for } i = 1, \ldots, n.$$

The above expression corresponds to the fact that in the ElGamal cryptosystem with many users sharing a modulus and a key generator, User 1 acquires a private key z_1 and a public key C_1, …, User n acquires a private key z_n and a public key C_n. It is well known that in this case, the attack of adversaries is still faced with DLP, namely seeking z_i from $C_i \equiv g^{z_i}$ (% M) for $i = 1, \ldots, n$ is equivalent to DLP [8].

Thus, when every g_i is weakened to a constant, seeking δ from $C_i \equiv g_i^{\delta}$ (% M) for $i = 1, \ldots, n$ is equivalent to DLP, which indicates that when every g_i is not a constant, seeking g_i and δ from $C_i \equiv g_i^{\delta}$ (% M) for $i = 1, \ldots, n$ is at least equivalent to DLP.

Secondly, singly consider a certain C_i, where the subscript i is designated.

Assume that $\bar{O}_{\mathrm{m}}(C_i, M, \underline{R})$ is an oracle on solving $C_i \equiv g_i^{\delta}$ (% M) for g_i and δ, where i is in $\{1, \ldots, n\}$, and $\underline{R}$ is a constraint on g_i such that the original g_i and δ can be found.

Let $y \equiv g^x$ (% M) be of DLP. Then, by calling $\bar{O}_{\mathrm{m}}(y, M, g)$, x can be obtained.

According to Definition 7, there is

$$\hat{H}(y \equiv g^x \text{ (\% } M)) \le_{\mathrm{T}}^{\mathrm{P}} \hat{H}(C_i \equiv g_i^{\delta} \text{ (\% } M)),$$

which means that when only a certain g_i is known, seeking g_i and δ from $C_i \equiv g_i^{\delta}$ (% M) is at least equivalent to DLP.

Integrally, seeking the original $\{A_i\}$, $\{\ell(i)\}$, W, and δ from $C_i \equiv (A_i W^{\ell(i)})^{\delta}$ (% M) for $i = 1, \ldots, n$ is computationally at least equivalent to DLP in the same prime field. □

The above proof illuminates that the distinctness of elements in the sets Ω and Ω' and the enlarging of value of $\mathcal{P}$ do not influence the correctness of Property 5.

4.2 Security of the Initialization Algorithm

Clearly, the security of the initialization algorithm depends on the security of the MPP $C_i \equiv (A_i W^{\ell(i)})^{\delta}$ (% M) with $A_i \in \Lambda' = \{2, 3, \ldots, \mathcal{P} \mid \mathcal{P} \ge 2^{18}\}$ and $\ell(i) \in \Omega' \subset \{\pm 5, \pm 7, \ldots, \pm(2n + 3)\}$ for $i = 1, \ldots, n$.

In [11], we analyze the security of the MPP $C_i \equiv (A_i W^{\ell(i)})^{\delta}$ (% M) with $A_i \in \Lambda = \{2, 3, \ldots, \mathcal{P} \mid \mathcal{P} \le 1201\}$ and $\ell(i) \in \Omega = \{5, 7, \ldots, (2n + 3)\}$ for $i = 1, \ldots, n$ from the three aspects, discover no subexponential time solution to it, and contrarily, find some evidence which inclines people to believe that MPP is computationally harder than DLP.

Likewise, considering that Ω' is different from Ω in [11], and the value of $\mathcal{P}$ is larger than the old one in [11], we will analyze the security of $C_i \equiv (A_i W^{\ell(i)})^{\delta}$ (% M) with $A_i \in \Lambda'$ and $\ell(i) \in \Omega'$, which is a supplement to the analysis in [11]. The supplemental analysis and the existent analysis tell us that $C_i \equiv (A_i W^{\ell(i)})^{\delta}$ (% M) with $A_i \in \Lambda'$ and $\ell(i) \in \Omega'$ has also no subexponential time solution at present.

4.2.1 Suppose that $\ell(x_1) + \ell(x_2) = \ell(y_1) + \ell(y_2)$

An adversary may eliminate W through judging $\ell(x_1) + \ell(x_2) = \ell(y_1) + \ell(y_2)$.

Because of $\Omega' \subset \{\pm 5, \pm 7, \ldots, \pm(2n + 3)\}$ and being subjected to $x + y \ne 0 \ \forall\ x, y \in \Omega'$, when the absolute values $|\ell(x_1)|$, $|\ell(x_2)|$, $|\ell(y_1)|$, $|\ell(y_2)|$ are deterministic, the value $\ell(x_1) + \ell(x_2) - (\ell(y_1) + \ell(y_2))$ has $2^4 = 16$ possible cases, which implies that there exists indeterminacy in the judgment of $\ell(x_1) + \ell(x_2) = \ell(y_1) + \ell(y_2)$ that interlaps with the indeterminacy of the lever function $\ell(i)$.

Refer to Section 4.2.1 of [11] for the rest of the analysis.

The running time of such an attack task is about 2^n [11], and it is not less than 2^m.

4.2.2 Suppose that $\|W\|$ Is Guessed

An adversary may eliminate W through the $\|W\|$-th power.

Raising either side of $C_i \equiv (A_i W^{\ell(i)})^{\delta}$ (% M) to the $\|W\|$-th power yields

$$C_i^{\|W\|} \equiv (A_i)^{\delta \|W\|} \text{ \% } M.$$

Suppose that the value of A_i is guessed, or the possible values of A_i are traversed.

Let $C_i \equiv \mathrm{g}^{u_i}$ (% M), and $A_i \equiv \mathrm{g}^{v_i}$ (% M), where g is a generator of $(\mathbb{Z}_M^*, \cdot)$. Then

$$u_i \|W\| \equiv v_i \|W\| \delta \text{ (\% } \bar{M})$$

for $i = 1, \ldots, n$. Notice that $u_i \ne v_i \delta$ (% $\bar{M}$), and $\{v_1, \ldots, v_n\}$ is not a super increasing sequence.

Obviously, if the adversary guesses the value of v_i, the equation $u_i \|W\| \equiv v_i \|W\| \delta$ (% $\bar{M}$) can be solved for δ. However the number of all the potential values of δ will be up 2^m due to $\|W\| \ge 2^{m-18}$ and $A_i \in \Lambda' = \{2, 3, \ldots, \mathcal{P} \mid \mathcal{P} \ge 2^{18}\}$.

Refer to Section 4.2.2 of [11] for the rest of the analysis.

The running time of such an attack task is greater than 2^n [11], and it is not less than 2^m.

4.3 Proof of Property 6 on ASPP

In Section 3.2, ASPP is defined as $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ (% M), where $\hat{b}_i = \underline{b}_i 2^{\partial_i}$ with $\partial_i = b_{i+(-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)}$ and $\underline{b}_i$ a bit shadow. Considering that a bit long-shadow $\hat{b}_i$ is different from a bit shadow $\underline{b}_i$ [11], we specially give the proof of Property 6 in Section 3.2.

Proof.

Assume that $\bar{O}_a(\underline{d}, C_1, \ldots, C_n, M)$ is an oracle on solving $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ (% M) for $\hat{b}_1 \ldots \hat{b}_n$, where $\hat{b}_1 \ldots \hat{b}_n$ is the bit long-shadow string of $b_1 \ldots b_n$.

Particularly, when $C_1 = \ldots = C_n = C$, define

$$\underline{d} \equiv \prod_{i=1}^{n} C^{(n+1)^{n-i}\hat{b}_i} \equiv \prod_{i=1}^{n} (C^{(n+1)^{n-i}})^{\hat{b}_i} \ (\%\ M)$$

due to $max(\hat{b}_1, \ldots, \hat{b}_n) \le n$, and then define the above oracle as $\bar{O}_a(\underline{d}, C^{(n+1)^{n-1}}, \ldots, C^{(n+1)^0}, M)$.

Let $\bar{G}_1 \equiv \prod_{i=1}^{n} C_i^{b_i}$ (% M) be of the subset product problem, shortly SPP [11][20][21].

Since there is $0 \le b_i \le \hat{b}_i$, and the mapping from $\hat{b}_1 \ldots \hat{b}_n$ to $b_1 \ldots b_n$ is one-to-one, by calling $\bar{O}_a(\bar{G}_1, C_1, \ldots, C_n, M)$, $b_1 \ldots b_n$ can be found.

By Definition 7, there is

$$\hat{H}(\bar{G}_1 \equiv \prod_{i=1}^{n} C_i^{b_i} \ (\%\ M)) \le_T^P \hat{H}(\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i} \ (\%\ M)).$$

In terms of Property 5 in [11], there is

$$\hat{H}(y \equiv g^x \ (\%\ M)) \le_T^P \hat{H}(\bar{G}_1 \equiv \prod_{i=1}^{n} C_i^{b_i} \ (\%\ M)).$$

Further by transitivity, there is

$$\hat{H}(y \equiv g^x \ (\%\ M)) \le_T^P \hat{H}(\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i} \ (\%\ M)).$$

Therefore, solving $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ (% M) for $\hat{b}_1 \ldots \hat{b}_n$ is at least equivalent to DLP in the same prime field in computational complexity. □

4.4 Security of the Compression Algorithm

Because the lightweight hash function contains only a round of iteration, the compression algorithm is the main body of it. Clearly, the security of the compression algorithm depends on the security of the ASPP $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ (% M), where $\hat{b}_i = \underline{b}_i 2^{\partial_i}$ with $\partial_i = b_{i+(-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)}$ and $\underline{b}_i$ a bit shadow.

In [11], we analyze the security of the ASPP $\bar{G} \equiv \prod_{i=1}^{n} C_i^{\underline{b}_i}$ (% M) from the three aspects, discover no subexponential time solution to it, and contrarily, find some evidence which inclines people to believe that $\bar{G} \equiv \prod_{i=1}^{n} C_i^{\underline{b}_i}$ (% M) is computationally harder than DLP. Due to $\hat{b}_i = \underline{b}_i 2^{\partial_i} \ge \underline{b}_i$, the security conclusion about $\bar{G} \equiv \prod_{i=1}^{n} C_i^{\underline{b}_i}$ (% M) is also suitable for $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ (% M) which is just another form of ASPP, namely $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ (% M) has no subexponential time solution at present.

In what follows, we specially analyze whether the compression formula $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ (% M) satisfies the four properties of a hash function, and resists the three classical attacks.

4.4.1 Lightweight Hash Is Computationally One-way

According to Section 3.2, apparently, given a short message $b_1 \ldots b_n \ne 0$, it is easy to calculate a related digest $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ (% M). Then see the contrary.

Let $C_1 \equiv g^{u_1}$ (% M), …, $C_n \equiv g^{u_n}$ (% M), $\underline{d} \equiv g^v$ (% M), where g is a generator of the group $(\mathbb{Z}_M^*, \cdot)$, and is easily found when $\lceil \lg M \rceil < 1024$.

Then, solving $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i}$ (% M) for $\hat{b}_1 \ldots \hat{b}_n$, namely $b_1 \ldots b_n$, is equivalent to solving

$$\hat{b}_1 u_1 + \ldots + \hat{b}_n u_n \equiv v \ (\%\ \bar{M}),$$

which is called the anomalous subset sum problem, shortly ASSP [11], and computationally at least equivalent to the subset sum problem due to $\hat{b}_i = \underline{b}_i 2^{\partial_i} \ge \underline{b}_i \ge b_i \in [0, 1]$.

It has been proved that SSP is NP-complete in its feasibility recognition form, and the computational version, especially the high-density version, is NP-hard [8][22]. Hence, solving ASSP is at least NP-hard.

Moreover in the lightweight hash function, there is $n \ge m = \lceil \lg M \rceil$ and $n \ge \hat{b}_i \ge b_i \in [0, 1]$. The knapsack density relevant to the ASSP $\hat{b}_1 u_1 + \ldots + \hat{b}_n u_n \equiv v$ (% $\bar{M}$) roughly equals

$$\textstyle\sum_{i=1}^{n} \lceil \lg n \rceil / \lceil \lg M \rceil = n \lceil \lg n \rceil / m > \lceil \lg n \rceil > 1,$$

which means that there exists many solutions to $\hat{b}_1 u_1 + \ldots + \hat{b}_n u_n \equiv v$ (% $\bar{M}$), namely the original solution cannot be determined, or will not occur in the reduced lattice base.

Hence, the L^3 lattice base reduction attack on ASSP [23][24] is utterly ineffectual, which illustrates that even although DLP with the bit-length of the modulus less than 1024 can be solved, the original $\bar{b}_1 \dots \bar{b}_n$ cannot be found yet in DLP subexponential time, namely $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\bar{b}_i}$ (% M) is computationally one-way.

4.4.2 Lightweight Hash Is Weakly Collision-free

Assume that $b_1 \dots b_n$ is a short message or an output of a hash function which contains at least two nonzero bits. Consequently, we easily understand that $\bar{b}_i = \underline{b}_i 2^{\partial_i} \le n \ \forall i \in [1, n]$.

Let $b_1 \dots b_n$ be a given short message, and $b'_1 \dots b'_n$ be another short message to need to be found.

Let $\underline{b}_1 \dots \underline{b}_n$ be the bit shadow string of $b_1 \dots b_n$, and $\underline{b}'_1 \dots \underline{b}'_n$ be the bit shadow string of $b'_1 \dots b'_n$.

Let $l\hat{h}$ be the compression algorithm of the lightweight hash function described in Section 3.2. Hence, we have

$$\underline{d} = l\hat{h}(b_1 \dots b_n) = \prod_{i=1}^{n} C_i^{\bar{b}_i} \,\%\, M,$$

and

$$\underline{d}' = l\hat{h}(b'_1 \dots b'_n) = \prod_{i=1}^{n} C_i^{\bar{b}'_i} \,\%\, M,$$

where $\bar{b}_i = \underline{b}_i 2^{\partial_i}$ with $\partial_i = b_{i + (-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)}$, and $\bar{b}'_i = \underline{b}'_i 2^{\partial'_i}$ with $\partial'_i = b'_{i + (-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)}$.

If $\underline{d} = \underline{d}'$, there is $\prod_{i=1}^{n} C_i^{\bar{b}_i} \equiv \prod_{i=1}^{n} C_i^{\bar{b}'_i}$ (% M).

Firstly, observe an extreme case.

Let $C_1 = \dots = C_n = C$.

Presume that $\underline{d} = \underline{d}'$. Then according to Section 4.3, $max(\bar{b}_1, \dots, \bar{b}_n) \le n$, and $max(\bar{b}'_1, \dots, \bar{b}'_n) \le n$,

$$\prod_{i=1}^{n} C^{(n+1)^{n-i}\bar{b}_i} \equiv \prod_{i=1}^{n} C^{(n+1)^{n-i}\bar{b}'_i} \ (\%\, M),$$

namely

$$C^{\sum_{i=1}^{n}(n+1)^{n-i}\bar{b}_i} \equiv C^{\sum_{i=1}^{n}(n+1)^{n-i}\bar{b}'_i} \ (\%\, M).$$

Let $z \equiv \sum_{i=1}^{n} \bar{b}_i (n+1)^{n-i}$ (% $\bar{M}$), and $z' \equiv \sum_{i=1}^{n} \bar{b}'_i (n+1)^{n-i}$ (% $\bar{M}$).

Correspondingly,

$$C^{z} \equiv C^{z'} \ (\%\, M).$$

We need to solve the above equation for z'.

If the order $\|C\|$ is known, then let $z' = z + k\|C\|$, where $k \ge 1$ is an integer, and there will be $C^z \equiv C^{z'}$ (% M). However, seeking $\|C\|$ is the integer factorization problem (IFP) at present because the prime factors of $\bar{M}$ must be known.

In practice, it is completely possible to make $C_1, \dots, C_n$ be pairwise unequal when $C_1, \dots, C_n$ are generated through the algorithm in Section 3.1, which implies that for any given short message $b_1 \dots b_n$, seeking another short message $b'_1 \dots b'_n$ such that $\prod_{i=1}^{n} C_i^{\bar{b}_i} \equiv \prod_{i=1}^{n} C_i^{\bar{b}'_i}$ (% M) is harder than IFP in computational complexity, namely $b'_1 \dots b'_n$ for $l\hat{h}(b_1 \dots b_n) = l\hat{h}(b'_1 \dots b'_n)$ cannot be found in IFP subexponential time.

Therefore, we say that the lightweight hash function is weakly collision-free.

Similarly, the lightweight hash function is resistant to single-block differential attack [25].

4.4.3 Lightweight Hash Can Resist Birthday Attack

Birthday attack is widely exploited in finding a collision w' of a message w such that $\hat{h}(w) = \hat{h}(w')$, where $\hat{h}$ is a hash function, $\hat{h}(w)$ is a related digest [26]. If the bit-length of the digest is m, an adversary can find the collision w' with non-negligible probability by using the birthday attack in roughly $1.25 \times 2^{m/2}$ running steps [27].

However, to the lightweight hash, the result will be utterly dissimilar.

Let $b_1 \dots b_n$ and $b'_1 \dots b'_n$ be two arbitrary different short messages, and $\bar{b}_1 \dots \bar{b}_n$ and $\bar{b}'_1 \dots \bar{b}'_n$ be two related bit long-shadow strings.

Suppose that $\underline{d} = \underline{d}'$, namely $\prod_{i=1}^{n} C_i^{\bar{b}_i} \equiv \prod_{i=1}^{n} C_i^{\bar{b}'_i}$ (% M).

Then, there is

$$\prod_{i=1}^{n} (A_i W^{\ell(i)})^{\delta \bar{b}_i} \equiv \prod_{i=1}^{n} (A_i W^{\ell(i)})^{\delta \bar{b}'_i} \ (\%\, M).$$

Further, there is

$$W^{\underline{k}\delta} \prod_{i=1}^{n} (A_i)^{\delta \bar{b}_i} \equiv W^{\underline{k}'\delta} \prod_{i=1}^{n} (A_i)^{\delta \bar{b}'_i} \ (\%\, M),$$

where $\underline{k} = \sum_{i=1}^{n} \bar{b}_i \ell(i)$, and $\underline{k}' = \sum_{i=1}^{n} \bar{b}'_i \ell(i) \,\%\, \bar{M}$.

Raising either side of the above congruence to the δ^{-1}-th power yields

$$W^{\underline{k}} \prod_{i=1}^{n} A_i^{\hat{b}_i} \equiv W^{\underline{k}'} \prod_{i=1}^{n} A_i^{\hat{b}'_i} \ (\% \ M).$$

Without loss of generality, let $\underline{k} \ge \underline{k}'$. Because $(\mathbb{Z}^*_M, \cdot)$ is an Abelian group, we have

$$W^{\underline{k}-\underline{k}'} \equiv \prod_{i=1}^{n} A_i^{\hat{b}'_i} (\prod_{i=1}^{n} A_i^{\hat{b}_i})^{-1} \ (\% \ M).$$

Due to $\gcd(q, \bar{M} / 2) = 1 \ \forall q \in [1, 8n(n+1)]$ and $\underline{k} - \underline{k}' \le 8n(n+1)$, there is

$$W \equiv (\prod_{i=1}^{n} A_i^{\hat{b}'_i} (\prod_{i=1}^{n} A_i^{\hat{b}_i})^{-1})^{(\underline{k}-\underline{k}')^{-1}} \ (\% \ M), \tag{1}$$

or

$$W^{2^k} \equiv (\prod_{i=1}^{n} A_i^{\hat{b}'_i} (\prod_{i=1}^{n} A_i^{\hat{b}_i})^{-1})^{((\underline{k}-\underline{k}') / 2^k)^{-1}} \ (\% \ M), \tag{2}$$

where $k \ge 1$ is a positive integer. Since $\bar{M}$ contains only one 2-factor, (2) has only two solutions. Therefore, if $\hat{b}_1 \dots \hat{b}_n$ and $\hat{b}'_1 \dots \hat{b}'_n$ satisfy (1) or (2), there will be $\underline{d} = \underline{d}'$.

Nevertheless, because $W \in (1, \bar{M})$ as a component of a private key is determinate, and $b_1 \dots b_n$ and $b'_1 \dots b'_n$, namely $\hat{b}_1 \dots \hat{b}_n$ and $\hat{b}'_1 \dots \hat{b}'_n$ are arbitrarily picked, the probability that $\hat{b}_1 \dots \hat{b}_n$ and $\hat{b}'_1 \dots \hat{b}'_n$ nicely satisfy (1) or (2) is only $1 / 2^m$, but not $1 / 2^{m/2}$ or so.

Moreover, because a private key $(\{A_i\}, \{\ell(i)\}, W, \delta)$ is unknown for an adversary, and MPP is one-way, it is also impossible that the adversary finds specific $b_1 \dots b_n$ and $b'_1 \dots b'_n$ satisfying (1) or (2) by utilizing the private key.

The above analysis shows that the lightweight hash is resistant to the birthday attack.

4.4.4 Lightweight Hash Can Resist Meet-in-the-middle Attack

Meet-in-the-middle dichotomy was first developed as an attack on an intended expansion of a block cipher by Diffie and Hellman in 1977 [28]. Section 3.10 of [8] brings forth a meet-in-the-middle attack algorithm for solving the subset sum problem.

INPUT: a set of positive integers $\{C_1, C_2, \dots, C_n\}$ and a positive integer s.

OUTPUT: $b_i \in \{0, 1\}$, $1 \le i \le n$, such that $\sum_{i=1}^{n} C_i b_i = s$, provided such b_i exist.

S1: Set $t \leftarrow \lfloor n / 2 \rfloor$.

S2: Construct a table with entries $(\sum_{i=1}^{t} C_i b_i, (b_1, b_2, \dots, b_t))$ for $(b_1, b_2, \dots, b_t) \in (\mathbb{Z}_2)^t$.
  Sort this table by the first component.

S3: For each $(b_{t+1}, b_{t+2}, \dots, b_n) \in (\mathbb{Z}_2)^{n-t}$, do the following:

  S3.1: Compute $r = s - \sum_{i=t+1}^{n} C_i b_i$ and check, using a binary search,
    whether r is the first component of some entry in the table;

  S3.2: If $r = \sum_{i=1}^{t} C_i b_i$, then return (a solution is $(b_1, b_2, \dots, b_n)$).

S4: Return (no solution exists).

It is not difficult to understand that the time complexity of the above algorithm is $O(n 2^{n/2})$.

Let $b_1 \dots b_n$ be a short message, its digest be $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i} \ (\% \ M)$.

If $n = m$, and $b_{n/2} = b_n = 1$ (thus, any bit shadow on the left of the middle has no relation with bits on the right), an adversary may attempt to attack the ASPP $\underline{d} \equiv \prod_{i=1}^{n} C_i^{\hat{b}_i} \ (\% \ M)$ by the meet-in-the-middle method.

However, owing to $\hat{b}_i = \underline{b}_i 2^{\partial_i}$ with $\partial_i = b_{i+(-1)^{\lfloor 2(i-1)/n \rfloor}(n/2)}$ for every $i \in [1, n]$, when i is from 1 to $n / 2$, there exists

$$\hat{b}_1 \dots \hat{b}_{n/2} = (\underline{b}_1 2^{b_{1+n/2}}) \dots (\underline{b}_{n/2} 2^{b_n}),$$

which involves all the bits of the short message, namely a reasonable middle does not exist.

If a fork is selected in proportion to $(n / 3 : 2n / 3)$ or $(n / 4 : 3n / 4)$, the right of the fork substantially still involves all the bits $b_1, \dots, b_n$.

For instance, let $n = 12$, a short message (a bit string) $= b_1 \dots b_{12}$, and a fork be to $(4 : 8)$, then

$$\hat{b}_5 \dots \hat{b}_{12} = (\underline{b}_5 2^{b_{11}})(\underline{b}_6 2^{b_{12}})(\underline{b}_7 2^{b_1})(\underline{b}_8 2^{b_2})(\underline{b}_9 2^{b_3})(\underline{b}_{10} 2^{b_4})(\underline{b}_{11} 2^{b_5})(\underline{b}_{12} 2^{b_6})$$

involves all the bits $b_1, \dots, b_{12}$.

The above dissection manifests that the meet-in-the-middle attack is essentially ineffectual on the lightweight hash function. Therefore, even if $n = m$, namely the input length = the output length of the function, the time complexity of the attack task is still $O(2^m)$ at present, but not $O(m 2^{m/2})$.

Besides, unlike $\sum_{i=1}^{n} C_i = \sum_{i=1}^{n} b_i C_i + \sum_{i=1}^{n} \neg b_i C_i$ in SSP, there is not

$$\prod_{i=1}^{n} C_i = \prod_{i=1}^{n} C_i^{\hat{b}_i} \prod_{i=1}^{n} C_i^{\neg \hat{b}_i} \ (\% \ M)$$

in ASPP, where $\neg \hat{b}_i$ is the bit long-shadow of $\neg b_i$, which implies there does not exist an easy relation between ASPP and dichotomy.

4.4.5 Lightweight Hash Can Resist Multi-block Differential Attack

[29] and [30] show multi-block near differential attack is effective on the classical hash functions MD5, SHA-0, SHA-1 etc which have the Merkle-Damgård iterative structure [6][7].

It is well known that MD5, SHA-0, and SHA-1 separately contain a quantity of rounds of inner iteration on each block, and each round consists of such linear or simple arithmetics as addition, shift, and logic operators.

The input of the lightweight hash function is a short message which is treated only as a block, and the number of rounds of inner iteration is at most n. The iteration consists of modular multiplication of the $\hat{b}_i$ power of C_i with $i \in [1, n]$ which is nonlinear and intricate. Thus, differential analysis loses a basis. Furthermore, the iteration leads to the fierce snowslide effect and the noninvertibility (see Section 4.4.1), and makes it impossible to derive a set of sufficient conditions which ensure that the collision differential characteristics hold for two messages which are expected to produce a collision through the lightweight hash [29][30].

Therefore, the lightweight hash is substantially distinct from the classical hashes MD5, SHA-0, SHA-1 etc, and the multi-block near differential attack suitable for the classical hashes will be utterly ineffective on the lightweight hash function.

4.4.6 Lightweight Hash Is Strongly Collision-free

Firstly, it is known from Section 4.4.2 that the lightweight hash function $l\hat{h}$ is weakly collision-free.

Secondly, for any arbitrary short message $b_1 \ldots b_n$, if want to find another short message $b'_1 \ldots b'_n$ such that $l\hat{h}(b_1 \ldots b_n) = l\hat{h}(b'_1 \ldots b'_n)$, an adversary must take $\hat{b}'_1 \ldots \hat{b}'_n$ from

$$\prod_{i=1}^{n} C_i^{\hat{b}_i} \equiv \prod_{i=1}^{n} C_i^{\hat{b}'_i} \ (\% \ M),$$

and further compute the bit string $b'_1 \ldots b'_n$. It is known from Section 4.4.2 that such a collision problem is computationally harder than IFP at present.

Thirdly, the lightweight hash is resistant to the birthday attack, the meet-in-the-middle attack, and the multi-block differential attack, and its security is up to $O(2^m)$.

Lastly, people do not find any subexponential time algorithm for solving ASPP, and the best method of solving ASSP is the force attack so far, which indicates that the subexponential time solution for any ASPP, including $\prod_{i=1}^{n} C_i^{\bar{y}_i} \equiv 1 \ (\% \ M)$, is not found now.

The above analysis manifests that the lightweight hash function is strongly collision-free.

Theorem 1: If a collision of the JUNA lightweight hash function can be found, the ASPP $\prod_{i=1}^{n} C_i^{\bar{y}_i} \equiv 1 \ (\% \ M)$ can be solved efficiently.

Proof.

Assume that $b_1 \ldots b_n \neq b'_1 \ldots b'_n$ are two arbitrary bit strings, $\hat{b}_1 \ldots \hat{b}_n$ and $\hat{b}'_1 \ldots \hat{b}'_n$ are two corresponding bit long-shadow strings, and $\prod_{i=1}^{n} C_i^{\hat{b}_i} \equiv \prod_{i=1}^{n} C_i^{\hat{b}'_i} \ (\% \ M)$ is a collision.

From $\prod_{i=1}^{n} C_i^{\hat{b}_i} \equiv \prod_{i=1}^{n} C_i^{\hat{b}'_i} \ (\% \ M)$, we have

$$\prod_{i=1}^{n} C_i^{\hat{b}_i - \hat{b}'_i} \equiv 1 \ (\% \ M).$$

Let $\bar{y}_i \equiv \hat{b}_i - \hat{b}'_i \ (\% \ \bar{M})$, and then

$$\prod_{i=1}^{n} C_i^{\bar{y}_i} \equiv 1 \ (\% \ M),$$

which means that the ASPP $\prod_{i=1}^{n} C_i^{\bar{y}_i} \equiv 1 \ (\% \ M)$ can be solved efficiently.

Therefore, the JUNA lightweight hash function is strongly collision-free. □

5 Time Complexity of the Lightweight Hash

Assume that time complexity is measured in the number of bit operations. Again we know that the time complexity of a modular multiplication is $O(2\lg^2 M)$.

The initialization algorithm in Section 3.1 is one-shot, and not real-time; thus it is unnecessary to care about its time complexity.

In what follows, we consider the time complexity of the compression algorithm in Section 3.2.

Due to $n \le \sum_{i=1}^{n} \hat{b}_i \le 2n$ for a nonzero bit string $b_1 \ldots b_n$, the compression algorithm takes at most $(2n - 1)$ modular multiplications, which means that the time complexity of the compression algorithm is $O(2(2n - 1)\lg^2 M) = O(n\, m^2)$.

For instance, when $m = 80$ and $n = 80$, the time complexity of the lightweight hash function is $80 \times 6400 = 512000$ bit operations which is equivalent to that of SHA-1 with 20 rounds of outer iteration, and less than that of SHA-1 with the number of rounds of outer iteration > 20. We know that the time complexity of SHA-1 with one round of outer iteration is about $32 \times 10 \times 80 = 25600$ bit operations. Thereby, the computation of the lightweight hash function for a single block is relatively small.

6 Reformation of a Classical Hash Function

Because the lightweight hash function is resistant to the birthday attack and the meet-in-the-middle attack, a classical hash function of which the output is n bits, and the security is intended to be $O(2^{n/2})$ may be reformed into a compact hash function of which the output is $n/2$ bits, and the security is equivalent to $O(2^{n/2})$.

For example, let $b_1 \ldots b_{128}$ be an output of MD5 [31], $\hat{b}_1 \ldots \hat{b}_{128}$ be a related bit long-shadow string, and $\lceil \lg M \rceil = 64$. Then, regard $\underline{d} = \prod_{i=1}^{128} C_i^{\hat{b}_i} \,\%\, M$ as the 64-bit output of the reformed MD5 with the equivalent security, where $C_i = (A_i W^{\ell(i)})^{\delta} \,\%\, M$ which is produced by the algorithm in Section 3.1.

Again for example, let $b_1 \ldots b_{160}$ be an output of SHA-1, $\hat{b}_1 \ldots \hat{b}_{160}$ be a related bit long-shadow string, and $\lceil \lg M \rceil = 80$. Then, regard $\underline{d} = \prod_{i=1}^{160} C_i^{\hat{b}_i} \,\%\, M$ as the 80-bit output of the reformed SHA-1 with the equivalent security.

The above two examples indicate that we may exchange time for space when the related security remains unchanged.

7 Conclusion

In the paper, the authors propose a lightweight hash function which contains the initialization algorithm and the compression algorithm, and converts a short message or a message digest of n bits into a string of m bits, where $80 \le m \le n \le 4096$.

The authors prove that both MPP and ASPP are at least equivalent to DLP in complexity, and analyze the security of the lightweight hash function. The analysis shows that the lightweight hash is computationally one-way, weakly collision-free, and strongly collision-free. Moreover, at present, any subexponential time algorithm for attacking the lightweight hash is not found, and its security is expected to be $O(2^m)$ magnitude.

Especially, the analysis illustrates that the lightweight hash function is resistant to the birthday attack and the meet-in-the-middle attack. By utilizing this characteristic, one can reform a classical hash function with the output of n bits and the security of $O(2^{n/2})$ into a compact hash function with the output of $n / 2$ bits and the equivalent security.

Simultaneously, the authors dissect the time complexity of compression algorithm of the lightweight hash function which is $O(n\, m^2)$ bit operations.

The lightweight hash function opens a door to convenience for the utilization of a lightweight digital signing scheme of which the length of modulus is not greater than 160 bits.

Acknowledgment

The authors would like to thank the Academicians Jiren Cai, Zhongyi Zhou, Changxiang Shen, Zhengyao Wei, Andrew C. Yao, Binxing Fang, Xicheng Lu, and Guangnan Ni for their important guidance, suggestions, and help.

The authors also would like to thank the Professors Dingyi Pei, Dengguo Feng, Jie Wang, Ronald L. Rivest, Moti Yung, Adi Shamir, Dingzhu Du, Mulan Liu, Huanguo Zhang, Maozhi Xu, Yixian Yang, Xuejia Lai, Xiaoyun Wang, Yupu Hu, Jiwu Jing, Rongquan Feng, Ping Luo, Jianfeng Ma, Xiao Chen, Dongdai Lin, Zhenfu Cao, Chao Li, Lei Hu, Lusheng Chen, Wenbao Han, Xinchun Yin, Bogang Lin, Qibin Zhai, Dake He, Hong Zhu, Zhiying Wang, and Quanyuan Wu for their important advice, suggestions, and corrections.

References

[1] I. F. Blake, G. Seroussi, and N. P. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, Cambridge, UK, 1999, ch. 3-5.

[2] T. ElGamal, A Public-key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Transactions on Information Theory*, v31(4), 1985, pp. 469-472.

[3] D. C. Ranasinghe, Lightweight Cryptography for Low Cost RFID, *Networked RFID Systems and Lightweight Cryptography*, Springer-Verlag, 2007, pp. 311-346.

[4] H.-Y. Chien, SASI: A new ultralightweight rfid authentication protocol providing strong authentication and strong integrity, *IEEE Transactions on Dependable and Secure Computing*, v4(4), 2007, pp. 337-340.

[5] A. Shamir, SQUASH - A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags, *Proc. of FSE' 08*, 2008.

[6] R. Merkle, One way hash functions and DES, *Proc. of Advances in Cryptology: CRYPTO 89*, Springer-Verlag, 1989, pp. 428-446.
[7] I. Damgard, A design principle for hash functions, *Proc. of Advances in Cryptology: CRYPTO 89*, Springer-Verlag, 1989, pp. 416-427.
[8] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, London, UK, 1997, ch. 2, 3, 5.
[9] W. Stallings, *Cryptography and Network Security: Principles and Practice* (2nd ed.), Prentice-Hall, New Jersey, 1999, ch. 8, 9.
[10] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (2nd ed.), John Wiley & Sons, New York, 1996, ch. 18.
[11] S. Su and S. Lü, A Public Key Cryptosystem Based on Three New Provable Problems, *Theoretical Computer Science*, v426-427, Apr. 2012, pp. 91-117.
[12] S. Y. Yan, *Number Theory for Computing* (2nd ed.), Springer-Verlag, New York, 2002, ch. 1.
[13] T. W. Hungerford, *Algebra*, Springer-Verlag, New York, 1998, ch. 1-3.
[14] K. H. Rosen, *Elementary Number Theory and Its Applications* (5th ed.), Addison-Wesley, Boston, 2005, ch. 12.
[15] M.J. Wiener, *Cryptanalysis of Short RSA Secret Exponents*, IEEE Transactions on Information Theory, v36(3), 1990, pp. 553-558.
[16] D. Chaum, E. Van Heijst, and B. Pfitzmann, Cryptographically strong undeniable signatures, unconditionally secure for the signer, *Proc. of Advances in Cryptology: CRYPTO '91* (LNCS 576), Springer-Verlag, 1992, pp. 470-484.
[17] D. Z. Du and K. Ko, *Theory of Computational Complexity*, John Wiley & Sons, New York, 2000, ch. 3-4.
[18] B. Schröder, *Ordered Sets: An Introduction*, Birkhäuser, Boston, 2003, ch. 3-4.
[19] M. Davis, *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*, Dover Publications, Mineola, 2004, ch. 2-4.
[20] D. Naccache and J. Stern, A new public key cryptosystem, *Proc. of Advances in Cryptology: EUROCRYPT '97*, Springer-Verlag, 1997, pp. 27-36.
[21] S. Su, S. Lü, and X. Fan, Asymptotic Granularity Reduction and Its Application, *Theoretical Computer Science*, vol. 412, issue 39, Sep. 2011, pp. 5374-5386.
[22] O. Goldreich, *Foundations of Cryptography: Basic Tools*, Cambridge University Press, Cambridge, UK, 2001, ch. 1-2.
[23] E. F. Brickell, Solving Low Density Knapsacks, *Proc. of Advance in Cryptology: CRYPTO '83*, Plenum Press, New York, 1984, pp. 25-37.
[24] M. J. Coster, A. Joux, B. A. LaMacchia etc, Improved Low-Density Subset Sum Algorithms, *Computational Complexity*, vol. 2, issue 2, 1992, pp. 111-128.
[25] T. Xie and D. Feng, Construct MD5 Collisions Using Just A Single Block Of Message, *Cryptology ePrint Archive*, http://eprint.iacr.org/2010/643, Dec. 2010.
[26] M. Bellare and T. Kohno, Hash Function Balance and Its Impact on Birthday Attacks, *Proc. of Advances in Cryptology: EUROCRYPT '04*, Springer-Verlag, Berlin, 2004, pp. 401-418.
[27] M. Girault, R. Cohen, and M. Campana, A Generalized Birthday Attack, *Proc. of Advances in Cryptology:EUROCRYPT '88* (LNCS 330), Springer-Verlag, Berlin, 1988, pp. 129-156.
[28] W. Diffie and M. E. Hellman, Exhaustive Cryptanalysis of the NBS Data Encryption Standard, *Computer*, v10 (6), 1977, pp. 74-84.
[29] E. Biham and R. Chen, A. Joux etc, Collisions of SHA-0 and Reduced SHA-1, *Proc. of Advances in Cryptology: EUROCRYPT '05*, Springer-Verlag, Berlin, 2005, pp. 36-57.
[30] X. Wang, Y. L. Yin, and H. Yu, Finding collisions in the full SHA-1, *Proc. of Advances in Cryptology: CRYPTO '05*, Springer-Verlag, New York, 2005, pp. 17-36.
[31] R. L. Rivest, The MD5 Message Digest Algorithm, *RFC 1321*, Apr. 1992.