Improvement on a Trapdoor Hash Function

Fuw-Yi Yang
Department of Computer Science and Information Engineering
Chaoyang University of Technology, Taichung County 41349, Taiwan
(Email: yangfy@cyut.edu.tw)
(Received Nov. 27, 2007; revised and accepted Jan. 30, 2008)

Abstract

By appending some bits to the original signature, a trapdoor hash function converts any signature scheme into secure signature scheme with very efficient online computation. Many of them have been proposed. However, all of them require performing a modular reduction during the online phase. The paper presents a trapdoor hash function to get rid of the modular reduction in online computation.

Keywords: Digital signature, trapdoor hash function

1 Introduction

The digital counterpart to a handwritten signature is the digital signature, which is an important primitive element in public key cryptosystems. Generally, a signer signed on a message (document) and then sent receiver (verifier) the digital signatures. The recipient verifies signatures by means of a predefined formula for verification of signatures. The signer may be a host computer, a mobile computer, or a smart card. Usually the latter two entities (called mobile signers) are powered by battery, which implies that they have limited processing capability.

Some trapdoor hash functions [3, 11] have been proposed to economize on the online computation when generating signature. The calculation after receiving message to be signed is called online computation. We will discuss this in detail in a later paragraph. The followings use a popular signature scheme to illustrate the computational requirement of this scheme and indicate that a mobile signer may be in a computational predicament when she/he constructing a signature.

In the RSA public key cryptosystem [10], signer's cryptographic parameters are selected as follows. Firstly, a signer chooses a random number e. After choosing e, the signer also chooses two large primes and computes the product of them, say N. Let $\phi(N)$ denote the Euler totient function, i.e. the cardinality of $Z_N^* = \{a|\ a \in Z_N$ and $\gcd(a,\ N) = 1\}$. Make sure that e and $\phi(N)$ are relatively prime, namely $\gcd(e,\ \phi(N)) = 1$. Then the modular inverse of e in the finite group $Z_{\phi(N)}^*$ is computed, $d = 1/e \bmod \phi(N)$. The signer publishes public key (N, e) and keeps secret key (d) privately.

Let H be a collision-resistant hash function, defined by $H : \{0,1\}^* \rightarrow Z_N$. A digital signature on a message $m \in \{0,1\}^*$ is obtained by computing $s = (H(m))^d \bmod N$. The pair (m, s) is called signed message. It is verified by testing that $s^e = H(m) \bmod N$.

Before discussing computational cost of the RSA signature described above, we quantify the cost of computation. For a typical public key cryptosystem, the bit length of N is 1024. Therefore, we use MM to denote a modular multiplication of two 1024-bit numbers modulo a 1024-bit modulus. Also let $|a|$ stand for the bit length of string a.

Ignoring the computational cost of hash function $H()$, generation and verification of signature demand $1.5|d|$ and $1.5|e|$ MMs (on average) respectively to complete these processes. A typical value for $|d|$ may be $|d| = |N| = 1024$. Thus constructing a digital signature requires an amount of 1536 MMs. This would consume 63.9 milliseconds. The timing was obtained using NTL library in a 3.2 GHz Pentium 4 running XP with 512 M Bytes RAM. Thank for NTL [5] which is a library for doing Number Theory. For 3.57 MHz Motorola 6805 CPU (the case of smart card in [9]), the timing will be at least 450 seconds. The heavy computation is not acceptable in some conditions, e.g. mobile signer, real-time applications. The notion of online/offline signature has been introduced in [1] to deal with the problem. The generation of signature is thus split up into two phases: offline process and online process. The offline computation is performed before the message is given. Upon receiving the message to be signed, the signature is constructed by online computation, using information that was computed during the offline phase.

Using the technique of trapdoor hash function proposed in [3], the online cost of generating a RSA signature is downed from 1536 MMs to only one MM. The work in [11] further reduces the online cost from one MM to only one modular reduction of a 1184 bit number modulo a 1024 bit modulus (it is estimated to be about 0.13 MMs). Section 2 will review the processing and discuss the performance.

1.1 Contributions

Further improvement in the calculation of online phase is possible. The paper will propose a scheme to replace the modular reduction with a conventional multiplication (a 160-bit number multiplied by a 1024-bit number). Usually, the computational cost required to do a modular reduction is more expensive than a multiplication. Thus the online computation is cut down. An implementation shows that our scheme saves about 30% as compared with the scheme in [11] (Section 3 describes the details). Since the processing does not require the operation of division, it is easier than other schemes to implement in the environment of smart cards.

1.2 Organization

Section 2 reviews the scheme of trapdoor hash function in [11]. In order to describe its usage, an example shows how to integrate a conventional scheme of digital signature with the trapdoor hash function. The requirements of a secure trapdoor hash function are also introduced. Section 3 describes the proposed schemes, proves the correctness as well as security. To compare the performances of the proposed scheme and the scheme in [11], the sizes of public hash key and secret trapdoor key, the bit length of appended string, offline computation, and online computation were listed in a table. Finally, Section 4 concludes the paper.

2 Scheme Review and Definitions

This section reviews the trapdoor hash function, called TH, presented in [11] and its associated properties. Assume that a composite number n is a product of two safe primes, namely $n = PQ, P = 2p + 1, Q = 2q + 1$, and both p and q are large primes with the same bit size. An element g with order $\lambda(n) = \text{lcm}(P - 1, Q - 1) = 2pq$ is selected. Then the secret trapdoor key is $TK = \lambda(n)$, and public hash key $HK = (g, n)$. Also defines a collision resistant hash function $H_1 : \{0,1\}^* \rightarrow \{0,1\}^l$, where l is the security parameter, e.g. $l = 160$.

The *hash operation* is defined as:

$$TH_{HK}(m_1, r_1) = g^{m_1||r_1} \bmod n, \quad (1)$$

where $m_1 \in_R \{0,1\}^l, r_1 \in_R Z_{\lambda(n)}$.

Note that symbol "$a \in_R A$" means that an element a is randomly selected from the set A; $m||r$ represents concatenation of strings m and r. Since the values of m_1 and r_1 are drawn randomly from the corresponding domains, the quantity $R = TH_{HK}(m_1, r_1)$ can be computed during idle time (offline). Also, signer applies signing key to generate signature, $s = H(R)^d \bmod N$. The triple (m_1, r_1, s) is stored for later usage.

Assume that a signer has determined to sign a target message m_2. Then the signer chooses a stored triple (m_1, r_1, s) and performs the *trapdoor operation* which is defined as:

$$\begin{aligned} & TH_{TK}(m_1, r_1, m_2) \\ = \; & r_2 \\ = \; & 2^k(m_1 - m_2) + r_1 \bmod \lambda(n). \end{aligned} \quad (2)$$

In Equation (2), $k = |\lambda(n)|$ is the bit length of the trapdoor key $TK = \lambda(n)$.

2.1 An Example

Perhaps an example will help us to describe the usage of the trapdoor hash function. This example assume that a trapdoor hash function and a RSA signature scheme are combined together to improve the efficiency of online computation when generating signature.

The *hash operation* and signing signature are performed in the offline phase. Namely, the signer chooses two random numbers, m_1 and r_1, and computes the hash value, $R = TH_{HK}(m_1, r_1)$. Then signer signs on the resultant hash value. Therefore all of the heavy computations are calculated in the offline phase.

Upon receiving the target message, M, signer initiates the online phase and executes the *trapdoor operation*. Signer uses the secret trapdoor key to find m_2 and r_2 which shall lead to $TH_{HK}(m_1, r_1) = TH_{HK}(m_2, r_2)$. Therefore signer needs not to sign again. The details are as follows.

Offline computation: Signer chooses at random a pair $(m_1, r_1) \in_R \{0,1\}^l \times Z_{\lambda(n)}$ and performs hash operation on this pair, i.e., $R = TH_{HK}(m_1, r_1) = g^{m_1||r_1} \bmod n$. Then using signer's secret key d, a RSA signature s is generated, $s = H(R)^d \bmod N$. The signer stores the triple (m_1, r_1, s) in storage.

Online computation: When receiving the target message M, the signer performs the trapdoor operation on (m_1, r_1) and $m_2 = H_1(M)$; namely $TH_{TK}(m_1, r_1, m_2) = r_2 = 2^k(m_1 - m_2) + r_1 \bmod \lambda(n)$. Then the signed message on M is the tuple (M, r_2, s). Note that in addition to the message M and signature s, an appended string r_2 is added to the signed message.

Verification: The signed message is verified by checking that $s^e = H(TH_{HK}(m_2, r_2)) \bmod N$, where $m_2 = H_1(M)$. It is easy to see that $TH_{HK}(m_2, r_2) = TH_{HK}(m_1, r_1)$. Suppose that the original signature is secure against generic chosen message attack, the resultant signature is enhanced to be secure against adaptive chosen message attack as shown by Theorem 1 in [11].

Performance: The major contribution to online computation (performing $TH_{TK}()$) is the reduction of a 1184 bits number modulo a 1024 bit modulus. It is estimated to be about 0.13 MMs. The offline computation of trapdoor hash function (performing $TH_{HK}()$) requires 1776 MMs, $1776 = 1.5 \cdot 1184$.

Note that the timing of computations is highly dependent on the underlying software and hardware. The running times obtained are as follows (using NTL library in a 3.2 GHz Pentium 4 running XP with 512 M Bytes RAM):

- Cost of performing $TH_{HK}() : 74.3 \times 10^{-3}$ seconds,
- Cost of performing $TH_{TK}() : 9.5 \times 10^{-6}$ seconds.

2.2 Secure Trapdoor Hash Function

A secure and practical trapdoor hash function must possess some properties. As introduced in [3, 11], Definition 1 lists these properties.

Definition 1. *A secure trapdoor hash function has three properties:*

1) Efficiency: Given hash key HK and $(m, r) \in \{0,1\}^l \times Z_{\lambda(n)}$, the hash value $TH_{HK}(m, r)$ is computable in polynomial time.

2) Collision resistant: Given hash key HK, there exists no probabilistic polynomial time algorithm outputs two pairs (m_1, r_1) and (m_2, r_2) producing the same hash value with non-negligible probability, where $m_1 \neq m_2, (m_i, r_i) \in \{0,1\}^l \times Z_{\lambda(n)}, i = 1, 2$.

3) Trapdoor collisions: Given trapdoor key TK and a triple $(m_1, m_2, r_1) \in \{0,1\}^l \times \{0,1\}^l \times Z_{\lambda(n)}$, there exists a probabilistic polynomial time algorithm outputs a value $r_2 \in Z_{\lambda(n)}$ and satisfies $TH_{HK}(m_1, r_1) = TH_{HK}(m_2, r_2)$. Further, if r_1 is uniformly distributed over its domain then r_1 and r_2 have statistically indistinguishable distribution in the same domain.

3 The Proposed Trapdoor Hash Function

If we interchange the concatenation order of r and m in Equation (1) (the Definition of hash operation), the new Definition of hash operation is as follows:

$$TH_{HK}(m, r) = g^{r||m} \bmod n.$$

Then the new trapdoor operation would be

$$\begin{aligned} & TH_{TK}(m_1, r_1, m_2) \\ = \; & r_2 \\ = \; & 2^{-l}((m_1 - m_2) + 2^l r_1) \\ = \; & 2^{-l}(m_1 - m_2) + r_1 = x(m_1 - m_2) + r_1, \end{aligned}$$

where $x = 2^{-l} \bmod \lambda(n)$.

Note that the modular reduction in the original trapdoor operation (Equation (2)) disappears. The first papers in which online signature computations save the cost of modular reduction are Girault at Eurocrypt'91 [2] and Poupard-Stern at Eurocrypt'98 [8]. The quantity r_2 is computed using integer arithmetic as those in [2, 6, 7, 8, 9]. The bit length of r_2 may vary in a wide range because that r_2 is a result of integer arithmetic. Therefore the new definition is forced to switch the position of r and m. We describe the details of the proposed trapdoor hash function as below.

The setting of parameters are similarly to the setting of the scheme TH reviewed in Section 2, i.e. $n = PQ, P = 2p + 1, Q = 2q + 1$, and both p and q are large primes with $|p| = |q|$. An element $g \in Z_n^*$ with order $\lambda(n) = 2pq$ is selected. Then publish the hash key $HK = (g, n)$. On the other hand, the secret trapdoor key TK is computed as $TK = x = 2^{-l} \bmod \lambda(n)$, where l is the security parameter, e.g. $l = 160$.

The *hash operation* is:

$$TH_{HK}(m_1, r_1) = g^{r_1||m_1} \bmod n. \tag{3}$$

In Equation (3), $m_1 \in_R \{0,1\}^l, r_1 \in_R \{0,1\}^{k+l}, k = |\lambda(n)|$.

The *trapdoor operation* is:

$$TH_{TK}(m_1, r_1, m_2) = r_2 = x(m_1 - m_2) + r_1, \tag{4}$$

where $(m_1, m_2, r_1) \in_R \{0,1\}^l \times \{0,1\}^l \times \{0,1\}^{k+l}$.

3.1 Performance

The running times obtained are as follows (computing environment is the same as those in Section 2.1):

- Cost of performing $TH_{HK}() : 97.9 \times 10^{-3}$ seconds,
- Cost of performing $TH_{TK}() : 6.5 \times 10^{-6}$ seconds.

The online computational cost has been reduced from 9.5 μs to 6.5 μs, save 31.5%. We know that doing a modular reduction requires more computational power than that of multiplication. Therefore, the result is consistent with expectations, since the modular reduction has been replaced with a conventional multiplication.

However, the appended string, i.e. r, has been lengthened from 1024 bits to 1184 (or 1185) bits. Also, the timing of computing hash operation is increased from 74.3 *ms* to 90.5 *ms*, spend 21.8%. The lengthened appended string causes this increment. Table 1 summarizes the comparison between the proposed scheme and scheme TH in [11].

Table 1: Comparison among proposed scheme and TH

	Proposed scheme	TH in [11]
Trapdoor operation	6.5 μs	9.5 μs
Hash operation	90.5 *ms*	74.3 *ms*
Trapdoor keys (bits)	2048	2048
Hash key (bits)	1024	1024
Bits appended	1184 or 1185	1024

* Trapdoor operation is executed in the online phase; Hash operation is performed during the offline phase.

3.2 Security of the Proposed Scheme

Section 2.2 requires that a useful trapdoor hash function must satisfy the properties of efficiency, trapdoor collisions and collision resistant. In the followings, we will illustrate that the proposed trapdoor hash function, satisfies all of the three properties.

Efficiency: Equation (3) describes how to perform the hash operation. It can be seen that the quantity $TH_{HK}(m,r) = g^{r||m} \bmod n$ is computed at the cost of 2016 MMs, $2016 = 1.5(|r| + |m|)$.

Trapdoor collisions: The trapdoor operation is defined in Equation (4). Using the trapdoor key x, the quantity $TH_{TK}(m_1, r_1, m_2) = r_2 = x(m_1 - m_2) + r_1$ is calculated at the cost of a conventional multiplication. It is clear that if r_1 is uniformly distributed over $\{0,1\}^{k+l}$, then r_2 is also uniformly distributed over $\{0,1\}^{k+l}$.

Before proving the property of collision resistant, a lemma is borrowed from [4].

Lemma 1. *[4] Assume that n is an RSA modulus, L is any multiple of $\phi(n)$, and $|L| = O(|n|^k)$. If n and L are available, then the factorization of n can be efficiently computed in time complexity $O(|n|^{4+k}M(|n|))$, where $M(|n|) = O(|n|log|n|loglog|n|)$.*

Collision resistant: This property will be proved by contradiction. Assume that public hash key $HK = (g, n)$ is given and there exists a polynomial time adversary $\mathcal{A}$ outputs two pairs (m_1, r_1) and (m_2, r_2) producing the same hash value with non-negligible probability, where $m_1 \neq m_2$ and $(m_i, r_i) \in \{0,1\}^l \times Z_{\lambda(n)}, i = 1, 2$. Thus the following equations hold true.

$$\begin{aligned} g^{r_1||m_1} &= g^{r_2||m_2} \bmod n \\ r_1 2^l + m_1 &= r_2 2^l + m_2 \bmod \lambda(n) \\ L &= (m_1 - m_2) + 2^l(r_1 - r_2) = 0 \bmod \lambda(n) \end{aligned}$$

Since $(m_1 - m_2) \neq 0$ we conclude that $L \neq 0$, which is a multiple of $\lambda(n)$.

Now, the quantities of n and $2L$ (a multiple of $\phi(n)$) are available. By Lemma 2, we can factor the large composite number n. The result contradicts the assumption that it is infeasible to factor a large composite number, which is a RSA modulus.

4 Conclusions

Trapdoor hash functions can aid any signature scheme to generate signatures online efficiently. An efficient TH has been proposed in [11], with our knowledge, it is the most efficient TH among trapdoor hash functions that have been proposed.

With the sacrifice of more appended bits and longer hash operation, the paper proposes a trapdoor hash function with very efficient in online computation. The online computation is showed to be more efficient than scheme TH. The proposed scheme seems attractive to the battery-powered computing devices because no more operation of modular reduction is required in online phase.

Acknowledgments

The author is grateful to the anonymous reviewers for valuable comments.

References

[1] S. Even, O. Goldreich, and S. Micali, "Online/Offline digital signature," *Advances in Cryptology (Crypto'89)*, LNCS 435, pp. 263-277, 1990.

[2] M. Girault, "Self-certified public keys," *Advances in Cryptology (Eurocrypt'91)*, LNCS 547, pp. 490-497, 1992.

[3] H. Krawczyk, and T. Rabin, "Chameleon signatures," *Symposium on Network and Distributed Systems Security*, pp. 143-154, 2000.

[4] G. Miller, "Riemann's Hypothesis and tests for primality," *Journal of Computer and System Sciences*, vol. 13, pp. 300-317, ACM, 1976.

[5] NTL, Available at http://shoup.net/ntl/.

[6] T. Okamoto, M. Tada, and A. Miyaji, "Efficient 'on the fly' signature schemes based on integer factoring," *Proceedings of the 2nd International Conference on Cryptology in India, Indocrypt*, LNCS 2247, pp. 275-286, 2001.

[7] D. Pointcheval, "The composite discrete logarithm and secure authentication," *Public-Key Cryptography*, LNCS 1751, pp. 113-128, 2000.

[8] G. Poupard and J. Stern, "Security analysis of a practical 'on the fly' authentication and signature generation," *Advances in Cryptology (Eurocrypt'98)*, LNCS 1403, pp. 422-436, 1998.

[9] G. Poupard and J. Stern, "On the fly signatures based on factoring," *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pp. 48-57, 1999.

[10] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signature and public key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978.

[11] A. Shamir and Y. Tauman, "Improved online/offline signature schemes," *Advances in Cryptology (Crypto'01)*, LNCS 2139, pp. 355-367, 2001.

[12] D. R. Stinson, *Cryptography, Theory and Practice*, CRC Press, 2nd Edition, 2002.

Fuw-Yi Yang received the B.Sc. degree and M.Sc. degree in the electronic engineering from National Taiwan University of Science and Technology, Taiwan, and the Ph.D. degree in the Department of Applied Mathematics, National Chung Hsing University, Taiwan. He is currently an associate professor with the Department of Computer

Science and Information Engineering, Chaoyang University of Technology. He is a member of the Chinese Cryptology and Information Security Association (CCISA). His research interests include computer cryptography, network security, and information security.