RSA Based Digital Signature for Secure Authentication

Shaikh Muhammad Allayear* and Sung Soon Park**
Dept. of Computer Science & Engineering, Anyang University
e-mail:*allayear@anyang.ac.kr,**sspark@aycc.anyang.ac.kr

Abstract

Now these days, many technical concepts and tools have been developed in the cryptographic field. Most digital signature schemes used in practice, such as RSA or DSA, have an important role in information privacy and secure authentication for perfect user. A clearly advantage of such schemes over with security proven relative to such common cryptographic assumptions, is their efficiency: as a result of their relative weak requirements regarding computation, bandwidth and storage, these scheme have so far beaten proven secure schemes in practice. Our aim is to contribute to bridge the gap that exists between the theory and practice of digital signature schemes. In this paper we present a digital signature that ensures information privacy. More precisely, under an appropriate assumption about RSA, the scheme is proven to be existentially forgeable under adaptively chosen message attacks. This mechanism can be applied to smart cards or E-Wallet for maintaining secure authentication for user's information privacy.

1. Introduction

When we consider an electronic transaction system, it requires a message authentication mechanism such as a digital signature scheme. Also, many types of Internet based applications need digital signatures for user authentication and for secure data integration. There are different approaches and all rely on the difficulty of factoring integers, computing discrete logarithms. Now this time, smart card and E-Wallet are the attractive developing area, so for security of smart card and E-Wallet need a powerful digital signature, which is essential requirement to guarantee user authentication and message integration. Even if the Smart card and E-wallet have to guarantee by the powerful digital signature based key with small sized storage but it is not easy work to develop small and limit storage sized signature for small storage.

In this paper, **section 2** describes the background of this paper to achieve our goal that is to build a secure RSA-Based signature for smart card or E-Wallet and Philosophy of our scheme. It will be basis of our schemes. In **section 3**, we derive our theoretical description of our scheme and algorithm steps of its preprocessing, initializations, signing and verification process. In **section 4**, we describe the performance of RSA digital signature, message attack security proof and optimization with one-way hash function for collision resistant.

2. Backgrounds and Goal

2.1. Background

In a sequence of results [1], [2], [3] and finally [4], it was established that the existence of one-way function is necessary and sufficient for the existence of a secure signature. This result, although theoretically very important, does not give rise to a practical signature scheme. The construction based on a general one-way hash function uses a costly "bit-by-bit" signing technique in conjunction with a tree authentication procedure [1]. As a result, the size of the signature is $o(k^2.\log i)$, where k stands for a security parameter and i indicate the number of signatures made.

Starting with the seminal paper [5], which proposed the RSA-functions as the first implementation of public key cryptography as envied by Diffie and Hellman[6], many practical digital signature schemes have been proposed, for instance, [7], [7], [9], [10], [11], [12], [13],[14] and [15].

However, none of these practical schemes have been shown to be secure in the sense [14] as none of these mentioned cryptographic assumptions holds. This implies that, independently of their validity, these requirement and common cryptographic assumptions may still turn out to be insufficient for the security of these signature schemes.

2.2. Our Philosophy

Our contribution is the design of a secure signature for information privacy where the size of the signatures is $(d+1)k$ bits, and where l^d signatures can be made. The integer numbers l and d can be chosen independently from the security parameter k. The security is derived from an appropriate RSA-assumption and can be helpful to build secure smart card or E-Wallet as a secure digital signature for information privacy and secure authentication.

3. RSA based Signature Scheme

In this section we have described our RSA based digital signature scheme, which can be convenient for using on Smart card or E-Wallet developing area to guarantee the security for the users.

3.1. Theoretical Description of our Scheme

In a preprocessing-phase, a security parameter k is determined as well as integers l and d. Next, a list $L=\{q, p_o, \ldots p_{l-1}$. consisting of $l+1$ distinct primes is generated by invoking an algorithm $H(1^k,1^l)$ here H is the hash function and the k is the bit string as and l is the integer. Ways of choosing H are discussed in the section 4.3.

Furthermore, we assume that we are given a probabilistic polynomial time generator G that, on input 1^k, outputs a triple (n,r,s), where r and s are primes and $n=r.s$ is a k bits integer. It is assumed that G is defined such that it is infeasible to factor n. Finally, we can get that q and p_i are co-prime to $\phi(n)$ [in RSA algorithm, when e is the encryption key and $\phi(n)$ co-prime, i.e. $gcd(e, \phi(n))=1$, in which $\phi(n)=(p-1)(q-1)$ is called Euler's totient fuction]. Given input n and L, define e as the smallest integer such that $q^e>n$ and e_i as the smallest integer such that $p_i^{e_i} > n$ for $i=0\ldots l-1$. In the following, w denotes q^e and v_i denotes $p_i^{e_i}$, for $i=0,\ldots l-1$.

The signer has a public key and RSA-modulus n, $h \in {}^*_n$ and $x_0 \in {}^*_n$. Here, n is generated by $G(1^k)$ and h and x_0 are chosen at random from ${}^*_n$ by the signer. In a possible variation of the scheme, x_0 and h are chosen mutually at random and are the same for all signers. In any case, h and x_0 must be chosen at random to avoid weak keys. As always, the knowledge of the factorization (r,s) of n enables the signer to compute $x^{\frac{1}{u}} \bmod n$ for any $X \in {}^*_n$ and any integer u such that $gcd(u,(r-1)(s-1))=1$. The public key consists of the triple (n, h, x_0). The factorization of n is private input to the signer.

The algorithm DFS(i) used in the formal description of our scheme, gradually develops a full l-ary tree of depth d by selecting the nodes at random from ${}^*_n$. The tree is constructed in depth-first fashion. Although not explicitly given as input to DFS(i), it is assumed that it has access to l, d, x_0 and n. The value x_0 serves as the root to the tree. Each time DFS(i) is invoked $(I=1\text{-}l^d)$, it creates a path to new leaf x_d and outputs this path, $x_1,\ldots,x_d$. This sequence is ordered such that x_{j-1} is the parent of x_j $(j=1\ldots d)$.

Furthermore, for each node x_j in this sequence, DFS(i) also outputs an indicator i_j $(j=1\ldots d)$ in such a way that i_j is assigned to x_j if and only if x_j is the i_j-th child of x_{j-1}. The amount of storage needed for this procedure (apart from l, d, x_0 and n) does not exceed the amount of storage needed for d-1 pairs consisting of a node and an indicator.

By invoking DFS(i), the signer gradually constructs, in a depth first fashion, an l-ary authentication tree with depth d: each time a new signature is required he constructs a path to a new leaf. All nodes x are members of ${}^*_n$, given by their smallest non-negative representative modulo n. The message space is equal to the set $\{0,1\}^k$, which we will also identify with the set of non-negative integers smaller than 2^k.

In Figure 1, the signers are making his/her i-th signature, on the message $m \in {}^*_n$. So, in particular x_d is the i-th leaf he reaches. The part of the tree on the right side of the path $x_0,\ldots, x_{d-1}, x_d$ is not yet constructed. Since x_1 happens to be the i_1-st child of x_0, the signer authenticates x_1 with respect to the prime p_{i_1} by computing

$$y_1 \leftarrow (x_0.h^{x_1})^{\frac{1}{v_{i_1}}} \bmod n .$$

Similar rules apply to the authentication of the remaining nodes in this path. Notice that the prime q is only used when the actual signature is computed, while the other primes in the list L are used exclusively in the process of constructing the authentication tree. The signature on m consists of the y_j and indicates i_j $(j=1\ldots d)$ and z.

To concerning the storage needed for the signer, notice that the part of the tree left from the path $(x_0,\ldots, x_{d-1},x_d)$ can be deleted. Actually x_d itself can be removed. In order to carry on with the depth-first construction of the tree, it is sufficient to store $x_0,\ldots,x_{d-1},x_d$ and the indicator to their parents. This storage amounts to all most $(d-1)(k+logl)$ bits (the root x_0 is the part of the public key).

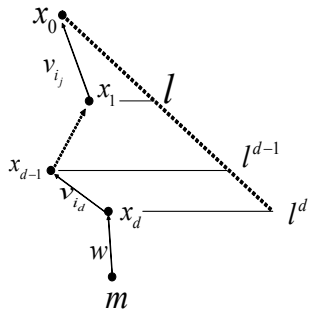

Figure 1. The i-th Signature by RSA.

A receiver of this signature gets only the message m, authentication values y_j, the indicators $i_j(j=1\ldots d)$ and z. So, what will be the nodes? These are re-computing as follows. On input of the public key, the list L, m and z he recomputed x_d as $x_d \leftarrow z^w.h^{-m} \bmod n$. Recursively the receiver re-computes x_{j-1} from x_j, y_j and i_j in a similar fashion $(j=d\ldots 1)$. The last nodes x_0 he thus computes should be equal to the actual x_0, which is part of the public key. If so, the signature is accepted.

3.2. Algorithm Steps of our Scheme

Now we are going to describe our scheme in more detail. We have divided it into four algorithmic steps as follows: preprocessing, initialization, signing and verification.

Preprocessing.

Step1. A security parameter k, integer l and d are determined.

Step2. Constant $L=\{q,p_0,\ldots,p_{l-1}\}$ consisting of $l=1$ distinct primes is generated by invoking $H(1^k,1^l)$.

Step3. Define e as the smallest integer such that $w \equiv q^e > n$, and e_i as the smallest integer such that $v_i \equiv p_i^{e_i} > n$, for $i=0\ldots l-1$. For possible choices of H, see section 4.3.

Initialization process.

Step1. The signer runs $G(1^k)$.

Step2. Obtains a triple (n,r,s) such that q and the p_i are co-prime to $\phi(n)$.

Step3. Chooses h and x_0 at random in ${}^*_n$.

Step4. Get *pk* (Public Key) pair *(n,h,x0),* while his secret key *sk* consists of the pair (*r,s).*

Signing process.

Step1. Let a *k* bit message *m* be given. Then the *i*-th signature, where $1 \le i \le l^d$, is computed as follows.

Step2. First, the signature puts $(x_1,i_1,....,x_d,i_d) \leftarrow DFS(i)$.

Step3. Next, he computes (for $j=1...d$)

$$y_j \leftarrow (x_{j-1}.h^{x_i})^{\frac{1}{v_{i_j}}} \bmod n .$$

Step4. Computes $z \leftarrow (x_d.h^m)^{\frac{1}{w}} \bmod n$. The signature σ on m consists of the value $z,y_1,i_1,...,y_d,i_d$.

Verification process.

Step1. Inputs the receiver signature $\sigma=(Z,Y_1,i_1...Y_d,i_d)$,

Step2. Input of $pk=(n,h,x_0)$,m and σ,

Step3. Computes $X_d \leftarrow Z^w.h^{-m} \bmod n$.

Step4. Finally, computes

$X_{j-1} \leftarrow Y_j^{u_{i_j}}.h^{-X_j} \bmod n (j=d...1)$. The signature is accepted if $X_0 \equiv x_0 \bmod n$.

4. Performances and Optimization

4.1. Proof Adaptive Chosen Message Attack Security

In this section, we proved the security of RSA-based signature schemes against the forgeable under adaptive chosen message attack.

Assumption.

Let k be a security parameter and *l* be of polynomial size in *k.* Let *L* be generated by $H(1^k,1^l)$ and *n* be an RSA modulus as generated by $G(1^k)$ and let *x* be a random member of $*_n$. Then there is no probabilistic polynomial time algorithm that has non-negligible probability of computing $x^{\frac{1}{\alpha}} \bmod n$ with $\alpha \in L$, on input *L, n* and *x.*

Theorem.

Under the above assumption, the signature theoretical scheme presented in Section 3.1 is not existentially forgeable under adaptively chosen message attacks.

Proof.

Before proving our security based on adaptively chosen message attacks, we will focus on some notations at below related to our proof:

l and *d*: used as integer number.

$*_n$: used as a part of signers public key.

$m, m, z, z, x, \mathcal{X}_i, y, y_j$: By serially m is a message where signer did not sign in simulator, *m* is a signed message, $\mathcal{Z}$ is a collusion signed key, z is a signed key, x is the leaf of tree, $\mathcal{X}$ is a collisioned leaf, y and y are as authentication value, *i* and *j* are the largest integer value.

We are given integer *l* and *d*, a list $L=\{q,p_o,...,p_{l-1}\}$ consisting of $l+1$ distinct primes and an RSA modulus *n*. Let *w* and *vi* be defined as in section 3.1, for $i=0...l-1$. We assume that n is generated according to $G(l^k)$, but we are not given the factorization. Also, we assume that *q* and p_i are co-prime to $\phi(n)$ and that *L* is generated according to $H(1^k,1^l)$. The proof is contradiction. We show that existence of a successful attacker implies that we can compute $x^{\frac{1}{\alpha}} \bmod n$, given a random $\alpha \in L$, and a random $X \in *_n$, which contradicts by the assumption.

After signing by the simulator according to the section 3.1 (Our Theoretical description) for proving our schemes security, we run the attacker against the signers and show that we can compute $X^{\frac{1}{\alpha}} \bmod n$, for random $\alpha \equiv L$, and a random $X \in *_n$. Here, we have essentially the same success-probability as the attacker. Recall that *G* and *H* generated *n* and *L* respectively.

We proceed as follows. We choose a random $\alpha \in L$, a random $X \in *_n$ and a random ρ from $*_n$. Put $h \leftarrow X^{\prod_{\beta \in L/\{\alpha\}} \beta}.\rho^{\prod_{\beta \in L} \beta} \bmod n$. Next feed *L, n, h* and $h^{\frac{1}{\beta}} \bmod n$ for all β and *L* different from α to the simulated signer. Next run the attacker against the simulator. Assume that after l^d calls to the simulated signer, the attacker outputs a forgery. $m, z, x_0, \mathcal{X}_1, i_1, y_1, ...x_d, i_d, y_d$.

As an example, a signature on a message m, where has not been signed by the simulator in the course to the attack. Now, let *T* denote the full-tree of depth *d* and branching *l* that the simulated signer has output in the course of the attack. Define *j* to be the largest integer such that $x_0, \mathcal{X}_1, i_1, ...x_j, i_j$ is a path in *T*. If $j=d$, then $\mathcal{X}_d$ is a leaf. So, there exists a signature $m, z, x_i, i_1, y_1, ...x_d, i_d, y_d$.

Output by the simulated signer, such that $\mathcal{X}_d = x_d$. By the assumption on m, we have $m \neq m$. So, we have $(y_d.\mathcal{Y}_d)^q \equiv h^{m-m} \bmod n$ where $(y_d.\mathcal{Y}_d)^q$ is indicating the correct authenticate value and attacked authenticated value with prime *q* and then hashed by h^{m-m}.□

4.2. Performance Evaluation

Our main focus of this paper is secure digital signature, which will be carry small sized storage of public key for smart card or E-Wallet. In this section, we describe our simulation and show the performance result in bellow by Table 1.

By using our scheme, we can get the results from Table 1, which are indicating the performances of our RSA based digital signature scheme. For developing our simulation, we used 512-bit key sized RSA algorithm. According to above-mentioned description, our RSA digital signature simulator takes time 544.61 ms for key generation or preprocessing, 915 ms for signing and 160 ms for sign verification. After all these process, we got our desired result that is 3k bits public key, which is our main focus also to develop secure digital signature for smart card or E-Wallet.

Steps	Times	Result of Public Key Storage size
Key Generation*1(ms)	544.61 ms	
Sign*100(ms)	915 ms	3k bits
Verify*100(ms)	160 ms	

Table 1. Our RSA digital signature scheme's computing effort.

4.3. Optimization

In this section, we describe a number of provably secure methods for decreasing the required size of the exponents in the list *L*.

Using Hash Function.

Let *H* be a collision resistant hash function that maps arbitrary sized input strings of size $k_* << k$. All values to be authenticated in the signature scheme, for example, the nodes in the tree and the message, are to be hashed down to k_* bits first. Also, a hash of the root can replace the root of the authentication tree as part of the public key.

To optimize with hash function we used MD5 algorithm. By optimizing with MD5, we can get the below results, as you can see in Table 2:

Hashing Algorithm	Bit Size	Byte Processed	Time taken
MD5	160-bit Message digest	8388608	0.190 sec

Table 2. Hash function's Computing effort

First compute the hash value then this hashed value will encrypted with RSA private key that is *sk* and then generate a secure digital signature. The effect is that the required size of the exponents is now k_* bits instead of *k* bits. The security statement now also requires that *H* is collision-resistant. This method can be used in conjunction with any of the other methods.

5. Conclusions

User authentication by digital signature is an important role in security area. Many kinds of algorithms have been developed, and RSA is one of them. In this paper, our main viewpoint is to develop secure RSA–based Signature for smart card and E-Wallet, which public key storage capacity should be follow the limitation for information privacy. Then, we designed tree based RSA digital signature by which we can limit the storage. It is convenient to take consecutive primes and we also can make the public key with 3 *k* bits, because in digital signature mechanism there is also one important role is size of keys. At last we proved our mechanism for proving the security with adaptively chosen message attacks the message attack.

Future Works. Design and implement smart card scheme according to RSA digital signature.

References

[1] R. C. Merkle: A Certified Digital Signature, Proceedings of Crypto '89, Springer Verlag LNCS series, pp. 234-246, 1989.

[2] M. Bellare and S. Micali: How to Sign Given any Trapdoor Funtion, Proceedings of Crypto '88, Springer Verlag LNCS series, pp. 200-215, 1988.

[3] M. Naor and M. Yung: Universal One-Way Hash Functions and Their Crytographic Applications, Proceedings of 21st STOC, pp. 33-43, 1998.

[4] J. Rompel: One-Way Functions are Necessary and Sufficient for Secure Signatures, Proceedings of 22nd STOC, pp. 387-394,1990.

[5] R. Rivest, A. Shamir and L. Adleman: A Method for Obtaining Digital Signatures and Public Key Cryptosystems, Communications of ACM, pp. 120-126, 21 (1978).

[6] W. Diffie and M. Hellman: New Directions in Crytography, IEEE Tansactions on Information Theory IT-22 (6): 644-654, 1976.

[7] T. ElGamal, A Pubulic-Key Cryptosystem and a Signature Scheme based on Discrete Logarithms, IEEE Transactions on Information Theory, IT31 (4): 469-472, 1985.

[8] A. Fiat and A. Shamir: How to Prove Yourself: Practical Solutions to Identification and Signature Problems, Proceedings of Crypto '86, pp. 186-194, 1986.

[9] C. Schnorr: Efficient Signature Generation by Smart Cards, Journal of Cryptology, 4 (3): 161-174, 1991.

[10] L. Guillou and J.J. Quisquater: A Practical Zero-Knowledge Protocol fitted to Security Microprocessor Minimizing both Transmission and Memory, Proceedings of Eurocrypt '88, Springer Verlag LNCS series, pp. 123-128, 1988.

[11] T. Okamoto: Provably secure and Practical Identification Schemes and Corresponding Signature Schemes, Proceedings of Crypto '92, Springer Verlag LNCS series, pp. 31-53, 1992.

[12] Information Technology – Security Techniques – Digital Signature Scheme Giving Message Recovery, ISO/IEC Standard 9796, first edition, International Standards Organization, Geneva, 2002-08-01.

[13] National Institute of Technology and Standards: Specifications for the Digital Signature Standard (DSS), Federal Information Processing Standards Publication, US. Department of Commerce, 1993.

[14] S. Goldwasser, S. Micali and R. Rivest: A Digital Signature Scheme Secure Against Chosen Message Attacks, SIAM Journal on Computing, 17(2): 281-308, 1988.

[15] C. Dwork and M. Naor: An Efficient Existentially Un-forgeable Signature Scheme and its Applications, Proceedings of Crypto '94, Springer Verlag LNCS series.1994.