Intelligent Information Management, 2009, 1, 174-179
doi:10.4236/iim.2009.13026 Published Online December 2009 (http://www.scirp.org/journal/iim)

Secure Signature Protocol

Shundong LI[1], Daoshun WANG[2], Yiqi DAI[2]
[1]*School of Computer Science, Shanxi Normal University, Xi'an, China*
[2]*Department of Computer Science and Technology, Tsinghua University, Beijing, China*
Email: *shundong@mail.tsinghua.edu.cn*

Abstract: This paper studies how to take advantage of other's computing ability to sign a message with one's private key without disclosing the private key. A protocol to this problem is presented, and it is proven, by well known simulation paradigm, that this protocol is private.

Keywords: cryptography, secure computation, signature, service, protocol

1. Introduction

Feigenbaum proposes the following problem [1]: Alice has a function $f()$ and a its instance x. She needs to compute $f(x)$, but does not have the corresponding computing resources, or because she is too lazy to do the computing. The sources here are in general sense which includes the computing time, algorithmic knowledge or corresponding hardware. If Bob has the corresponding resources, can she take advantage of Bob's resources without trust him? That is, can she use Bob's resources to compute $f(x)$ without letting Bob know x and $f(x)$?

In some cases, this problem can easily be solved. For example, if Alice and Bob are geographically near. In this case, Alice can rent Bob's computing resources to compute $f(x)$, and we call that Bob supplies Alice with computing service. In other cases, it may be very difficulty to solve. For example, Alice and Bob are far from in distance. In this case, complicated cryptographic protocol will be necessary to solve it.

Studying secure computing service is of great theoretical importance to computing science and cryptography. R. Cramer once said: "If we can securely compute any function, computing science will have a new powerful tool [2]." The combination of secure multiparty computation [3] and secure computing service can realize the objective to securely compute any function. Furthermore, studying secure computing service is of great practical importance to information security. In contrast to the rapid development of secure multiparty computation [4–7], secure computing service is still stagnant. Only the secure computations of discrete logarithm and the inequality of vectors have been solved. No other problems have been solved. This paper studies the secure signature problem, proposes a protocol for it. It is proven, by well known simulation paradigm, that the protocol has privacy-preserving property.

2. Preliminaries

2.1. Notations

Let x be a variable, $f()$ be a function, and $f(x)$ the value of $f()$ at x. $dom(f)$ is used to denote the domain of $f()$, and $range(f)$ the codomain. If two objects A, B are computationally indistinguishable, we denote $A \overset{c}{\equiv} B$. Two computationally indistinguishable objects can be considered completely equivalent in all computations. Computational indistinguishability is an important basis for many kind of security in cryptography. For more details of computational indistinguishability, see [8].

2.2. Related Work

Feigenbaum's secure computation protocol for discrete logarithm is as follows: Let p be a big prime number, Z_p^* be the set $\{1,2,\cdots,p-1\}$ and the multiplicative operation on it, g the generator of Z_p^*. Instance $x \in Z_p^*$, Alice wants to take advantage of Bob's computing resources to compute $e = f(x) \in Z_p^*$ such that $g^e \bmod p = x$ without letting Bob knowing x.

- Alice randomly chooses a $c \in Z_p^*$, computes $x' = x \cdot g^c \bmod p$.
- Alice sends x' to Bob, asks Bob to figure out an $e' = f(x')$ such that $g^{e'} \bmod p = x'$.
- Bob sends $f(x') = e'$ to Alice.
- Alice computes $e = f(x) = (e' - c) \bmod (p-1)$.

In the protocol above, apart from other simple operations, Alice still needs to compute complicated modular exponential operations. Though modular exponential operation is rather complicated, its complexity is much less than that of computing discrete logarithm. So this scheme is meaningful. We assume that Bob is semi-honest, that is, he will execute the protocol properly and he will sends correct result to Alice, but he may also keep the record of intermediate computation to try to figure out *x* or *f*(*x*) provided he is interested in them.

Definition 1 [3] Let *f*() be a computable function, π a two party protocol for computing *f*(). On input $x \in dom(f)$, the message that Bob received during the execution of π, denoted by is ($r, m_1, \cdots, m_t$), where r is Bob's independent coin toss for determining what algorithm and strategy be adopted to compute *f*(*x*), m_i the *i*-th message Bob received. For function *f*(), if there exists a polynomial time algorithm *S* such that

$$\{f(m,d), S(s)\}_{m,d\in dom(f)} \overset{c}{\equiv} \{(output_A^{\pi}(m,d), view_B^{\pi}(m,d)\}_{x\in dom(f)} \tag{1}$$

we say that π privately computes *f*(), where $output_B^{\pi}(x)$ is Alice's final output, and in most case $output_B^{\pi}(x)$ = *f*(*x*). That is, if there exists a simulator *S* which on input *f*(*x*), can simulate the protocol execution on input *x*, and obtains a sequence that is computationally indistinguishable from ($r, m_1, \cdots, m_t$), then the protocol is private for computing *f*() at *x*.

3. Secure Signature Protocol

In this problem, Alice has a message *m* to be signed. Because digital signature needs to compute $m^e \bmod n$ where *m, e* may be relative large and *n* is a very large number (may be greater than 2^{1000}). Alice does not have corresponding computing resources, so she has to take advantage of Bob's resources to finish her signature. We assume that Alice signs message by RSA algorithm with private/public keys pair *d/e*, then her signature on message *m* is $m^d \bmod n$. *d* can be written as

$$d = d_0 + d_1 \cdot 2 + \cdots + d_k \cdot 2^k \tag{2}$$

For example, 127 can be written as

$$127 = 1\cdot 2^0 + 1\cdot 2^1 + 1\cdot 2^2 + 1\cdot 2^3 + 1\cdot 2^4 + 1\cdot 2^5 + 1\cdot 2^6 \tag{3}$$

and in this case, $d_0 = d_1 = \cdots = d_7 = 1$, or be written as

$$127 = (-1)\cdot 2^0 + 0\cdot 2^1 + 0\cdot 2^2 + 0\cdot 2^3 + 0\cdot 2^4 + 0\cdot 2^5 + 0\cdot 2^6 + 0\cdot 2^7 \tag{4}$$

and in this case, $d_0 = -1,\ d_7 = 1,\ d_1 = \cdots = d_6 = 0$. To facilitate Alice's signature, *d* = 127 should be expressed with (4). In contrast, if *d* = 129, it should be expressed with (3), and in this case, $d_0 = d_7 = 1,\ d_1 = \cdots = d_6 = 0$. No matter which form is adopted, it holds that

$$m^d \bmod n = m^{d_0 + d_1\cdot 2 + \cdots + d_k \cdot 2^k} \bmod n$$

In what follows, we denote *d* by

$$d = [\, d_0, d_1, \cdots, d_k \,].$$

Thus following protocol follows:

3.1. Protocol

Protocol 1 Secure signature protocol
Inputs: message *m* and Alice's private key *d* = [$d_0, d_1, \cdots, d_k$].

Output: the signature on *m*, that is, $m^d \bmod n$

1) Alice sends *m* to Bob, and asks Bob to compute *U* = ($u_1, u_2, \cdots, u_k, u_{k+1}, \cdots, u_p$) where

$u_i = m^{2^i} \bmod n \ (i = 1, 2, \cdots, p)$

and *V* = ($v_1, v_2, \cdots, v_k, v_{k+1}, \cdots, v_p$), where

$v_i = (m^{2^i})^{-1} \bmod n = u_i^{-1} \bmod n \ (i = 1, 2, \cdots, p)$

2) Bob computes *U, V*, and sends them to Alice.
3) Having received *U, V*, Alice computes

$$s(m) = \prod_{i=0}^{k} (u_i)^{d_i} \bmod n$$

Then *s*(*m*) is Alice's signature on message *m* with private key *d*. If *m* is the result obtained by encrypting some message with Alice's public key, then this protocol decrypts the message using Bob's computing resources without letting Bob know Alice's private key and the plaintext.

It seems that Alice does not obtain much advantage from this protocol, because she still has to compute multiplication of some big integers. In fact, this protocol gives Alice much advantage due to the following fact:

- It is very easy to compute $(u_i)^{d_i}$ because $d_i \in \{-1, 0, 1\}$, and

$$(u_i)^{d_i} = \begin{cases} u_i & if \ \ d_i = 1 \\ 1 & if \ \ d_i = 0 \\ v_i & if \ \ d_i = -1 \end{cases} \tag{5}$$

So, Alice does not need to compute $(u_i)^{d_i}$

- Many d_i's have specific structure, that is, many d_i's in *d* = [$d_0, d_1, \cdots, d_k$] is 0, which facilitates the computation. For example, 65535=2^{16} -1, if we want to compute m^{65535} mod *n*, it is sufficient to compute

$$m^{2^{16}-1} \bmod n = m^{2^{16}} \cdot m^{-1} \bmod n = u_{16} \cdot v_1 \bmod n$$

Similarly, if *d* = 4295098369 = 1 + 2^{17} + 2^{32}, then $d_0 = d_{17} = d_{32} = 1$ and all other d_i's are 0. So

$$s(m) = \prod_{i=0}^{k} (u_i)^{d_i} \bmod n = u_1 \cdot u_{17} \cdot u_{32} \bmod n$$

- Bob's computational complexity is not the concern of secure computing service protocol. In this protocol, Alice just needs to transform *d* into its binary representation *d* = [$d_0, d_1, \cdots, d_k$] and chooses the terms with $d_i \neq 0$ ($0 \le i \le k$) to compute by following algorithm

```
Sets $s \leftarrow 1$
For $i = 0$ to $k$
IF $d_i = 1$ then $s \leftarrow s \cdot u_i \bmod n$
IF $d_i = -1$ then $s \leftarrow s \cdot v_i \bmod n$
Outputs $s$
```

3.2. Privacy-Preserving Property

We first intuitionally analyze the security of this protocol followed by a rigorous proof. To know Alice's private *d*, Bob has to know every d_i of [$d_0, d_1, \cdots, d_k$]. In this protocol, Alice does not tell Bob any information about *d*, not even the largest index *k*, and what he knows is just *p*(*p* > *k*). He is not expected to obtain information about *d*.

Suppose that Bob can somehow know *k*, then he can guess every d_i with probability 1/3, because $d_i \in \{-1,0,1\}$. And he can successfully guess *d* with probability 3^{-k}. Reader may argue that d_i's are not uniformly distributed over {0,1,-1}, the successful probability is at most 2^{-k} even we neglect the possibility that d_i takes -1, because d_i's are at least uniformly distributed over {0,1}. This conclusion has nothing to do with Bob's computing power. No matter what powerful computing ability Bob has, there is no enough information to help him decide every d_i.

Of course, Bob may obtain more information to help him decide d_i after having seen Alice's signature on message *m*. This is beyond the scope of this paper, because even if he did not help Alice sign the message, he can still obtain corresponding information to do this. That is, if taking part in this protocol makes him can determine d_i, he can also do this without taking part in. In other words, this protocol does not leak any information about *d*. About the privacy preserving property, we have following theorem.

Theorem 1 The signature protocol above, denoted by π is private

Proof We prove this theorem by showing a simulator *S* such that (1) holds. The message sequence Bob received during the execution of π is $view_B^\pi(m,d) = \{m,r,U,V,s\}$, where *r* is Bob's independent coin toss for determining what algorithm to be used to compute *U, V*. The key here is that the simulator just has $m^d \bmod n$ as its input, and it does not even know *m*, how can it simulate the protocol execution. The simulation proceeds as follows:

1) On input *s*, simulator *S* can ask Alice to send her public key *e* to it, and computes

$$s^e \bmod n = m .$$

2) Without the requirement of Alice, *S* computes *U, V*. Same *m* will certainly result in same *U, V*.

3) *S* has known *s*. If it can figure out $m^d \bmod n$ for *s, m*, it outputs $m^d \bmod n$, otherwise outputs *s*. In this protocol $f(m,d) = output_A^\pi(m,d) = output_B^\pi(m,d) = s$

4) Let $S(s) = \{m,r,U,V,s\}$, it is obvious that

$$\{f(m,d), S(s)\}_{m,d \in dom(f)} \overset{c}{\equiv} \{(output_A^\pi(m,d), view_B^\pi(m,d)\}_{x \in dom(f)}$$

The theorem follows.

The proof shows that if Bob can figure out Alice's *d*, after taking part in the execution of the protocol and obtaining *s*, then he can do this without taking part in the execution. That is, taking part in the protocol cannot help Bob obtaining Alice's private key. If RSA signature algorithm is secure, then this protocol is private.

3.3. Alice's Computing Power Saving

Taking multiplication as the basic unit to measure computational complexity, if Alice signs message *m* himself, her computational complexity is *O* (log *d*) which can not be further decreased. In this protocol, Alice takes advantage of Bob's computing resources greatly decreasing her computational complexity. For example, if *d* = 4295098369, Alice has to do multiplication 48 times on average (to compute $m^{2^1} \bmod n, m^{2^2} \bmod n, \cdots, m^{2^{32}} \bmod n$) and $\prod_{i=0}^{32} (u_i)^{d_i} \bmod n$ for those $d_i \neq 0$, while by this protocol, she just needs to do multiplication operation 2 times. On average, 1.5 log *d* multiplications are necessary for Alice to sign a message, but using this protocol, Alice just needs to compute 0.5log *d* multiplications. The bigger *d* is, the bigger computational complexity gap will be, and the more obvious advantage of secure signature protocol will be displayed.

3.4. An Example

Suppose that *n* = 6012707, Alice's private key is *d* = 65535 and the message that Alice wants to sign is *m* = 5234673. Because $d = 2^{17}$ -1, *d* = [-1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1].

1) Alice sends *m* and asks Bob to compute *U* = ($u_1, \cdots, u_{17}, \cdots, u_{24}$) and *V* = ($v_1, \cdots, v_{17}, \cdots, v_{24}$) where

$$u_i = m^{2^i} \bmod n \ (i = 1, 2, \cdots, 24).$$

2) Bob computes *U* = (5234673, 1615224, 4939341, 1743565, 262732, 2227464, 245501, 5378740, 770381, 2640726, 4457202, 4343034, 4300965, 2413687, 4765873, 3615999, 5681056, 1936650, 837333, 2827740, 862217, 1048902, 2304158, 1973155, 4642799, 3908573), uses Euclidean extended algorithm to compute *V*, and obtains *V* = (793604, 301394, 4378587, 2779681, 3344118, 1258019, 1182184, 1471018, 4884922, 1030980, 5442354, 3070495, 2943611, 5814105, 5409191, 5823024, 5614508, 1347304, 3850530,

3419982, 4474211,1691689, 3649001, 2506724, 1583928, 1862606).

3) Alice simply computes $S = v_0 \cdot u_{17} \bmod n$ = 5681056× 793604 mod 6012707 = 676014

4. Research Background

In scientific research, computing method has become the third important research method that parallels theoretical method and scientific experiment method. Computing method bridges the theoretical method and experiment method. The problems met in computing science show some new trends: the problem domain is wider and wider, the problems are more and more complicated, the problem sizes are bigger and bigger, the datum are more and more. To solve these complicated problems, supper computing power and data analysis ability are necessary. Supper computing power has become the symbol of the development level of science and technology of a country, and embodies the competition ability of national science and technology. To own supper computing power and data analysis ability, scientists developed high performance computing, parallel computing and grid computing etc. Many countries built their super computing center. But many super computing centers lack corresponding computing task after their construction which makes these super computing centers cannot fully play their roles. To fully play their roles and maximize their benefits, they have to open to outside and supply computing service to the public.

This is good news to some organizations which are worrying about their irregular and great computing power requirements. Because their requirements of computing power are irregular, it does not pay to buy powerful computing equipment. If they can rent corresponding computing power from super computing centers when they need, then the computing power of super computing centers will be fully played and their irregular requirements will be met. They do not need to expensively purchase and maintain computing equipments. This is win-win to both the super computing centers and the public. In mobile computing environments, mobile equipments are often short of computing power, thus if some computing that mobile equipments have to make can be transported through internet to some computing service provider, then both the mobile service providers and computing service providers will be very interested in this new computing mode. This new computing mode can also promote the development of mobile e-commerce. To sum up, this computing service mode has a strong appeal to computing service providers and receivers. But the trouble this win-win computing mode meets is that the computing service providers cannot supply secure computing service. If Alice's problem is a secret computing task, signature operation for example, she will hesitate to use the computing power of computing service providers. If the computing process cannot keep the privacy of the computing, Alice would rather give up using other's computing power. Privacy-preserving property is a basic requirement of public computing service. It is also an urgent problem that must be solved for exploiting computing service market, privacy preserving and information security.

If fully studied, secure computing can solve all above problems. It can also exploit many new applications. For example, identification in cyberspace is realized by means of public key signature, but the computing power that public key signature needs is beyond that an ordinary internet user posses. It is unrealistic to demand an ordinary user to buy expensive computing equipment, or to take a long time learning corresponding algorithmic knowledge. It is a huge task to popularize signature and identification among ordinary users. If the computation can be done by secure computing service providers, while ordinary users just do some simple operations, then signature and identification will be very easy to popularize, and this will be of great theoretical and practical significance to information security and the development of computing science.

5. Conclusions

This paper studies secure signature and decryption problem, and presents a secure computing service protocol. It is also proven, by well known simulation paradigm, that this protocol is privacy preserving. This problem has important practical significance in computing science and cryptography. The combination of secure multiparty computation and secure computing service can make us near to the objective of secure computing any function, so it also has essential theoretical significance. Of course, this protocol does not give Alice too much advantage to sign a message, and we will further study more protocols that may give Alice more and more advantages.

6. Acknowledgment

This research is supported by the National Natural Science Foundation of China (Grant No. 60673065, 60873249) and National High Technology Development (863) Program (2008AA01Z419, 2009AA011906).

REFERENCES

[1] J. Feigenbaum, "Can you take advantage of someone without having to trust him," LNCS, Vol. 218, Springer-verlag, N.Y., pp. 477–488, 1986.

[2] R. Cramer and I. Damgaard, "Introduction to secure multi-party computations," In: Contemporary Cryptology, pp. 41–87, Advanced Courses in Mathematics CRM Barcelona, Birkhauser, at: http://homepages.cwi.nl/ cramer/, 2005.

[3] O. Goldreich, "Foundations of cryptography: Basic ap-

plications," Cambridge University Press, London, 2004.

[4] W. L. Du and M. J. Atallah, "Secure multi-party computaion problems and their applications: A review and open problems." In: Proceedings of New Security Paradigms Workshop, ACM Press, New York, pp. 13–22, 2001.

[5] S. Goldwasser, "Multi-party computations: Past and present [C]," In Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing (Santa Barbara, CA, August, 1997), ACM Press, New York, pp. 21–24, 1997.

[6] S. D. Li, D. S. Wang, Y. Q. Dai, and P. Luo, "Symmetric cryptographic solution to Yao's millionaires' problem and an evaluation of secure multiparty computations," Information Sciences, Vol. 178, pp. 244–255, 2008.

[7] Y. Lindell, "General composition and universal composability in secure multiparty computation," Journal of Cryptology, Vol. 22, No. 3, pp. 395–428, 2009.

[8] O. Goldreich, "Foundations of cryptography: Basic tools," Cambridge University Press, London, 2001.