Active Trial-and-error Attack on SASC Protocols

Heeyoul Kim, Younho Lee, Seong-Min Hong, and Hyunsoo Yoon
(Corresponding author: Seong-Min Hong)

Department of EECS, Korea Advanced Institute of Science and Technology
373-1 Guseng-dong, Yuseong-gu, Daejeon 305-701, Rep. of Korea
(Email: {hykim, yhlee, smhong, hyoon}@camars.kaist.ac.kr)

(Received July 1, 2005; revised and accepted Aug. 2, 2005)

Abstract

SASC (Server-Aided Secret Computation) protocols enable a client (a smart card) to borrow computing power from a server (e.g., an untrustworthy auxiliary device like an ATM) without revealing its secret information. In this paper, we propose a new active attack on server-aided secret computation protocols. We describe our attack by using Beguin and Quisquater's protocol. (We modify the protocol in order to immunize it against Nguyen and Stern's lattice reduction attack.) The proposed attack reduces the search space $\mathcal{P}$ to $\frac{1}{p} + p\mathcal{P}$, where $0 < p < 1$. It is $2\sqrt{\mathcal{P}}$ for optimal p. Practically, it effectively threatens SASC protocols because an attacker can choose an appropriate value p according to the situation. Therefore, the security parameters in the existing SASC protocols must be reconsidered.

Keywords: Active attack, SASC protocol, smart card

1 Introduction

Management of secret information is one of the most important problems that needs to be solved in cryptosystems. It is especially necessary to use a device that can carry secret information in public-key cryptosystems such as RSA [14] because of their large key sizes. Smart cards (plastic cards to which IC chips are attached) are very useful for this purpose due to their portability and security. Additionally they have computability, and are widely used as electronic wallets in electronic commerce, ID cards, and so on.

However, RSA signature generation requires such a heavy computation that devices with poor computing power such as smart cards cannot perform it efficiently. To answer this weakness, there have been many studies on how to enable a smart card to borrow computing power from a server. (As it is generally used in cooperation with auxiliary devices including ATMs, it is natural to put more computing power into few large servers than into portable smart cards.)

SASC (Server-Aided Secret Computation) protocols enable a smart card (a client) to perform secret computations faster with the aid of a server (an untrusted auxiliary device like an ATM). Matsumoto, Kato, and Imai proposed the first SASC protocol for the RSA signature generation [11], and it significantly accelerated the computation. Afterwards, a lot of effective attacks that can threaten SASC protocols have been designed and the corresponding countermeasures also have been proposed [1, 2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 16, 17]. The previous studies related to this topic are reviewed in [3, 8] in detail.

Attacks on SASC protocols are divided into two groups: passive attacks and active ones. A passive attack does not disturb the protocol and uses the only information that can be obtained by observing it. Representative passive attacks are Pfitzmann and Waidner's birthday-like attack [13] and Nguyen and Stern's orthogonal lattice reduction attack [12]. On the other hand, in an active attack, an attacker participates in the protocol as a malicious server and obtains additional information by returning wrong results to the client. There are some representative active attacks: Anderson's one-round attack which uses prime numbers [1], Shimbo and Kawamura's factorization attack [15] and Lim and Lee's generalized version of it [8], and Pfitzmann and Waidner's multi-round attack which uses the Jacobi symbol [13].

The one-round active attacks that can break the system in one step are prevented by checking at the client whether the resulting signature is correct [1, 8, 15]. However, the final signature checking cannot be the countermeasure of the multi-round attack that reveals some information by observing whether the client gives the correct signature [13].

There are three protocols that were designed to be secure against the Pfitzmann and Waidner's multiround active attack. Beguin and Quisquater proposed a server-aided RSA computation protocol that is secure against all known passive and active attacks including the multi-round attack [3]. Lim and Lee immunized Matsumoto, Imai, Laih, and Yen's two-phase protocols to all known active attacks [8, 10]. Hong, Shin, Lee, and Yoon proposed a new approach (blinding-based technique) which adds and multiplies a series of random numbers with the secret information to hide it [5].

Their common approach to the countermeasure of the multi-round active attack is to newly decompose (or to blind) the client's secret with random values on each execution of the protocol. In this paper, we show that a risk is still present though the client re-decomposes its secret every time. We propose a new attack that threatens all the existing SASC protocols including Beguin and Quisquater's scheme, Lim and Lee's scheme, and Hong et al.'s blinding scheme. The proposed attack allows an attacker (a malicious server) to guess the vector that is used to decompose the secret information. (If the guess is correct, the client gives the signature.) Although our attack seems to be similar to Pfitzmann and Waidner's multi-round active attack, the proposed attack can be applied even if the SASC protocols newly select the random numbers each time.

Our attack reduces the search space $\mathcal{P}$(e.g., 2^{64}) to $\frac{1}{p} + p\mathcal{P}$, where p is the probability that the client gives the correct signature. Theoretically, the search space can be reduced to $2\sqrt{\mathcal{P}}$(e.g, 2^{33}). We also present several variants of the proposed attack to enable the attacker to select p as he wants with finer granularity. The proposed attack threatens all of the SASC protocols in the real world, because an attacker can select an appropriate p according to the available computing power and the number of chances that he can participate in the protocol as a malicious server. Practical threatening of the proposed attack will be discussed in Section 4.2 in detail.

This paper is organized as follows. In Section 2, we describe server-aided RSA computation protocols. We explain our attack in Section 3. In Section 4, the search space reduction by the proposed attack is computed, and it's practical meaning is discussed. In Section 5, we propose several variants of the proposed attack. Lastly, we conclude in Section 6.

2 Server-Aided RSA Computation

In RSA, a signer computes two large primes p,q and their product n and then chooses a random integer ν which is reciprocal to $\phi(n)(= (p-1)(q-1))$ and finds s which satisfies $s\nu \equiv 1 \bmod \phi(n)$. In this setting, the signature S for a message M is $M^s \bmod n$, and it can be verified by examining whether $S^\nu \bmod n$ is M.

The objective of SASC protocols is to enable the client to efficiently compute $M^s \bmod n$ with the aid of the server. The first SASC protocol used decomposition of secret s into several pieces (x_i and a_i, where $s = \sum_{i=0}^{m-1} x_i a_i \bmod \phi(n)$), and reveals some of them(x_i) and conceals the others(a_i) [11]. The more advanced ones which have been designed since use a similar basic decomposition with more refined techniques. The proposed attack can be applied to all protocols that use the basic decomposition.

Beguin and Quisquater's Protocol

Although the proposed attack can be applied to all SASC protocols, we use the Beguin and Quisquater's protocol in the detailed description of the proposed attack because of the space limit. Now we describe the modified BQ scheme. In the following description, $w_p = q(q^{-1} \bmod p)$ and $w_q = p(p^{-1} \bmod q)$.

Client Server

choose randomly $\vec{x}$ and $\vec{a}$:

$\vec{a} = a_0, a_1, \ldots, a_{m-1}$ *s.t.* $a_i \in 0, \ldots, h$,

$\vec{x} = x_0, \ldots, x_{m-1}$ *s.t.* $0 \le x_i \le \phi(n)$.

$\xrightarrow{M, n, \text{ and } \vec{x}}$

compute $\vec{z}$:

$\vec{z} = z_0, z_1, \ldots, z_{m-1}$

s.t. $z_i = M^{x_i} \bmod n$.

$\xleftarrow{\vec{z}}$

compute z_p and z_q :

$z_p = \Pi_{i=0}^{m-1} z_i^{a_i} \bmod p$ and $z_q = \Pi_{i=0}^{m-1} z_i^{a_i} \bmod q$.

compute $s_2 = s - s_1 \bmod \phi(n)$, where $s_1 = \sum_{i=0}^{m-1} a_i x_i$.

compute σ_p and σ_q :

$\sigma_p = s_2 \bmod (p-1) + \varrho_p(p-1)$,

$\sigma_q = s_2 \bmod (q-1) + \varrho_q(q-1)$,

where $\varrho_p \in_R \{0, \ldots, q-2\}$, and $\varrho_q \in_R \{0, \ldots, p-2\}$.

$\xrightarrow{\sigma_p, \sigma_q}$

compute y_p and y_q :

$y_p = M^{\sigma_p} \bmod n$

$y_q = M^{\sigma_q} \bmod n$

$\xleftarrow{y_p, y_q}$

compute $S = w_p S_p + w_q S_q \bmod n$, where

$S_p = y_p z_p \bmod p$ and $S_q = y_q z_q \bmod q$.

If $S^\nu \bmod n = M$, then transmit S.

In the above protocol, if the attacker knows s_1 then he can factor public modulus n, because $\gcd(n, S - y_p M^{s_1} \bmod n) = p$. This means that the secret information s is revealed to the attacker. If the attacker succeeds in guessing $\vec{a}$, then he can compute $s_1 = \sum_{i=0}^{m-1} a_i x_i$. Therefore, the search space $\mathcal{P}$ is $\frac{1}{2}(h+1)^m$. h and m are security parameters, and they should be selected so as to make the search space larger than 2^{64} and also to minimize the amount of computation.

In [3], the authors state that their scheme is secure against Pfitzmann and Waidner's multi-round active attack, because the client newly chooses decomposition vectors $\vec{x}$ and $\vec{a}$ on each execution in their scheme. While the multi-round active attack reveals a bit of information per a trial (a round), the revealed information is useless to the newly selected decomposition vectors. However, we show that a risk is still present though the client re-decomposes its secret every time. (Recently, their protocol was broken by Nguyen and Stern's lattice reduction attack. However, the attack is a passive attack and out of scope of this letter. Moreover, the attack is applicable to only Beguin and Quisquater's scheme whereas our active attack threatens all the previous SASC protocols.)

3 Probabilistic Active Attack

In an active attack on SASC protocols, an attacker participates in the protocol as a server and returns wrong results to the client. In Beguin and Quisquater's protocol, a malicious server can return $\vec{z}'$,y_p', and/or y_q' instead of $\vec{z}$,y_p and/or y_q, respectively. By doing that, if he can obtain useful information about the client's secret s, it is a successful active attack.

Beguin and Quisquater's scheme newly chooses two vectors $\vec{x}$ and $\vec{a}$ on each execution of the protocol to resist against the multi-round active attack. (Lim and Lee's immunization technique also involves re-decomposing the client's secret every time.) However, it cannot remove the possibility of active attacks.

If a malicious server guesses some parts of $\vec{a}$ with a probability p and the client gives the answer whether or not the guessing is correct, $\frac{1}{p}$ trials are sufficient for the attacker to guess it correctly on the average and the number of possible candidates for $\vec{a}$ is reduced directly proportional to p.

The basic idea of the proposed attack is to enable the malicious server to guess the ratios among a_is, i.e., the elements of $\vec{a}$. If he succeeds in guessing, the client gives the correct signature and this results in the reduction of the search space for a_is.

3.1 The Two-Term Attack

As described in Section 2, the knowledge of a_is reveals the secret s in Beguin and Quisquater's protocol. In this section, we explain the two-term attack to show the principle of guessing the ratios among a_is. The objective of the two-term attack is to reduce the search space by guessing $r_{1,0}$, which is the ratio of a_1 to a_0. From now on, we denote $\frac{a_i}{a_j}$ as $r_{i,j}$.

Firstly, the malicious server computes $l = \text{lcm}(1, 2, \ldots, h)$, and then returns z_0' and z_1' instead of z_0 and z_1:

$$\begin{aligned} z_0' &= M^{(1+l\frac{a_1'}{a_0'})x_0} \bmod n, \text{ and} \\ z_1' &= M^{x_1 - lx_0} \bmod n. \end{aligned} \tag{1}$$

In Equation (1), a_0' and a_1' indicates the values that the malicious server supposes as a_0 and a_1, respectively. Because a_0' divides l, the exponent of z_0', that is $(1 + l\frac{a_1'}{a_0'})x_0$, is always an integer. The malicious server returns correct z_is for $2 \le i \le m-1$, and conforms to the succeeding steps as stated in the protocol.

If $\frac{a_1'}{a_0'}$ is equal to $r_{1,0} = \frac{a_1}{a_0}$, the following equation is satisfied:

$$\begin{aligned} (z_0')^{a_0} \times (z_1')^{a_1} &\equiv M^{a_0x_0 + la_1x_0} \times M^{a_1x_1 - la_1x_0} \bmod n \\ &\equiv M^{a_0x_0} \times M^{a_1x_1} \bmod n \\ &\equiv (z_0)^{a_0} \times (z_1)^{a_1}. \end{aligned}$$

Therefore, the correct signature is generated and the final signature checking is passed.

$$(z_0')^{a_0} \times (z_1')^{a_1} \times \prod_{i=2}^{m-1} z_i^{a_i} \equiv M^{s_1} \bmod n. \tag{2}$$

That is, if the card gives the correct signature, the server can infer that $\frac{a_1'}{a_0'}$ is equal to $r_{1,0}$[1].

As the successful malicious server knows $r_{1,0}$, he does not need search $a_1(=a_0 \times r_{1,0})$ separately. Also, because $a_0 \times r_{1,0}$ should be less than or equal to h and a_0 is an integer, the range of a_0 is limited according to $r_{1,0}$. For example, if $r_{1,0} = 2$, then a_0 should be an integer less than $\frac{h}{2}$, which is half the original range. (If $r_{1,0}$ is less than one, the attacker should swap a_0 and a_1. For example, if $r_{1,0} = \frac{1}{2}$, the attacker should search a_1 instead of a_0. Then, the range of possible values of a_1 is half the original one.)

The probability of the success of the attack is that of $\frac{a_1'}{a_0'} = r_{1,0}$. If we let the probability be p_0, the search space for a_is, $\mathcal{P}$, is reduced to $p_0 \times \mathcal{P}$. p_0 will be computed in Section 4.

3.2 Multi-Term Attack

The two-term attack can be generalized to be applied to t terms. The malicious server returns z_i's instead of z_is, where $0 \le i \le t-1$. t is even and z_i's are as follows:

$$\begin{aligned} z_{2k}' &= M^{(1+l\frac{a_{2k+1}'}{a_{2k}'})x_{2k}} \bmod n, \text{ and} \\ z_{2k+1}' &= M^{x_{2k+1} - lx_{2k}} \bmod n, \text{ where } 0 \le k \le \tfrac{t}{2} - 1. \end{aligned} \tag{3}$$

For $i > t$, the correct z_is are computed and returned to the client. And, the succeeding procedures are executed as designated in the original protocol.

In Equation (3), if every $\frac{a_{2k+1}'}{a_{2k}'}$ is equal to $r_{2k+1,2k}$, the correct signature is generated and the final signature check is passed.

[1] In Equation (1), a_0 cannot be zero. If the malicious server guesses a_0 as zero, he should swap the equations of z_0' and z_1'. And, if a_0 and a_1 are all zeros, the correct signature is always generated for any z_0' and z_1'. Therefore, the success of the attack means that $a_0' + a_1'$ may be zero.

If we let the probability of $\frac{a'_{2k+1}}{a'_{2k}} = r_{2k+1,2k}$ be p_k, the probability of a successful active attack is $\prod_{k=0}^{\frac{t}{2}-1} p_k$. Therefore, $\prod_{k=0}^{\frac{t}{2}-1} \frac{1}{p_k}$ trials are sufficient to succeed with a probability larger than $1-\frac{1}{e}$.

As the successful malicious server knows $r_{2k+1,2k}$, where $0 \le k \le \frac{t}{2}-1$, he does not need to search $a_1, a_3, \ldots, a_{t-1}$ separately, because $a_{2k+1} = a_{2k} \times r_{2k+1,2k}$. Also, the range of a_{2k} is limited according to $r_{2k+1,2k}$. As a result, the search space $\mathcal{P}$ is reduced to $\prod_{k=0}^{\frac{t}{2}-1} p_k \times \mathcal{P}$.

4 Risk Analysis

In this section, we compute the lower bound of the search space and discuss the practical threat of our attack. Our attack means the multi-term attack.

4.1 Minimum Search Space

If we let p be the probability that the attacker can successfully guess the secret, the attacker who performs a passive attack $\frac{1}{p}$ times can find secret s with a probability of one. On the other hand, the attacker who performs an active attack $\frac{1}{p}$ times can find the secret with a probability of $1-(1-p)^{\frac{1}{p}} (= p_a)$. The range of p_a can be represented as follows:

$$p_a = 1 - (1-p)^{\frac{1}{p}} > 1 - \frac{1}{e} \approx 0.63.$$

We adopt a new definition of search space to compute the one reduced by the proposed active attack[2].

Definition 1. In a server-aided secret computation protocol, **a search space** is the minimum number of trials that is required for the attacker to find the client's secret with a probability larger than $1 - \frac{1}{e}$.

If we let $\mathcal{P}$ be the search space for a passive attack against Beguin and Quisquater's protocol, the one for our attack, $\mathcal{A}$, is as follows:

$$\mathcal{A} = (\prod_{k=0}^{\frac{t}{2}-1} \frac{1}{p_k}) + (\prod_{k=0}^{\frac{t}{2}-1} p_k) \times \mathcal{P}. \qquad (4)$$

We compute p_k. Let d_0 and d_1 be integers between zero and h, and let $d_{1,0}$ be the ratio of d_1 to d_0. Then, the number of (d_0, d_1) pairs which have the same $d_{1,0}$ is $\lfloor \frac{h \times \gcd(d_0,d_1)}{\max\{d_0,d_1\}} \rfloor$. As p_k is the probability of $\frac{a'_{2k+1}}{a'_{2k}} = r_{2k+1,2k}$, p_k is as follows:

$$\frac{1}{(h+1)^2} \le p_k = \frac{\lfloor \frac{h \times \gcd(a_{2k},a_{2k+1})}{\max\{a_{2k},a_{2k+1}\}} \rfloor + 1}{(h+1)^2} \le \frac{1}{h+1} \qquad (5)$$

Every a_i has a value between zero and h with same probability. Therefore, the attacker chooses (a'_{2k}, a'_{2k+1}) in the proportion of the number of (d_0, d_1)s which have the same $d_{1,0}$. Average value of p_ks, $\tilde{p_k}$, is as follows:

$$\tilde{p_k} = \sum_{i=0}^{h} (\frac{i+1}{(h+1)^2})^2 = \frac{(h+2)(2h+3)}{6(h+1)^3}.$$

The average search space for the proposed attack is as follows:

$$\mathcal{A} = (\tilde{p_k})^{-\frac{t}{2}} + (\tilde{p_k})^{\frac{t}{2}} \times \mathcal{P}. \qquad (6)$$

When the two terms in Equation (6), $(\tilde{p_k})^{-\frac{t}{2}}$ and $(\tilde{p_k})^{\frac{t}{2}} \times \mathcal{P}$, are the same, $\mathcal{A}$ has a minimum value. Such optimal t is $t_{opt} = -\frac{1}{2} \times \log_{\tilde{p_k}} \mathcal{P}$. As $0 < \frac{t_{opt}}{m} < 1$, t_{opt} always exists in the appropriate range. The search space for the multi-term attack using t_{opt} terms is as follows:

$$\begin{aligned} \mathcal{A} \quad &= (\tilde{p_k})^{-\frac{t_{opt}}{2}} + (\tilde{p_k})^{\frac{t_{opt}}{2}} \times \mathcal{P} \\ &= 2 \times (\tilde{p_k})^{-\frac{t_{opt}}{2}} \\ &= 2\sqrt{\mathcal{P}}. \end{aligned}$$

Consequently, if we assume that active attacks can be performed with no restriction, the search space $\mathcal{P}$ is reduced to $2\sqrt{\mathcal{P}}$. In [3], the authors selected security parameters h and m so that the search space would be 2^{64}. However, the proposed attack reduces it to 2^{33} for that parameters, which can be easily searched thoroughly.

4.2 Practical Threatening

In the previous section, to compute the theoretical lower bound of the search space, we assumed that active attacks could be performed with no restriction. However, in the real world, having more failures than expected in the final signature check may cause alarm, and the card readers or vendors will be suspected. Therefore, it is hard to assume that active attacks can be tried unlimitedly.

However, it is sometimes possible to perform active attacks with no restriction. For example, a lost card or one whose owner is under menace is vulnerable to active attacks with little restriction. In addition, as stated in [13], it is not easy to inform a card's owner if an active attack has been performed, because a smart card does not have a monitor or other auxiliary devices.

In any case, we should assume that active attacks are possible. The attacker can choose an appropriate t according to his computing power and the number of chances in which he can participate as a malicious server.

For example, in [3], the authors selected $< h = 10, m = 19 >$ as one of the security parameter pairs that are secure and efficient. It makes search space 2^{64} and minimizes the amount of computation. In a system with this

[2] We let p be the probability that the proposed attack succeeds. If the attacker tries $5 \times \frac{1}{p}$ times, he can succeed with a very high probability, larger than $0.99 (\approx 1 - \frac{1}{e^5})$. Generally, $c \times \frac{1}{p}$ trials ensure the success with a probability larger than $1 - \frac{1}{e^c}$. However, as each trial is a Bernoulli trial, it conforms to a binomial distribution, and the expected value of $c \times \frac{1}{p}$ trials is c. That is, if the attacker tries $c \times \frac{1}{p}$ times, he succeeds in c values on the average. Therefore, we choose the criterion value as $1 - \frac{1}{e}$ such that the expected value is one.

setting, we assume an attacker with the computing power of 2^{50}-trials/day. He would have to search for 2^{14} days, that is, for more than ten years. However, if he performs the proposed attack with $t = 4$, he can find s in a day after 16,000 transactions. If he performs the attack with $t = 2$, he can succeed in a month after a hundred transactions. This means that a smart card that is contacted by a malicious server a hundred times can be broken. It also means that one card per hundred which have ever had transactions with the server could be broken after a month.

If the proposed attack succeeds, the client does not even know that there has been an attack. Therefore, the attacker can try passive attacks to find the left a_is for a long time. Furthermore, if a given server has performed a proposed attack, all smart cards that have ever been in contact with the server has the possibility that their secrets were or will be revealed. This can cause disorder or chaos in a credit-based society.

5 Attacks with a Finer Grain p

In the proposed attack, the probability that the attacker will succeed in guessing, p, can be selected as he wants. However, p can be only multiples of $(h+1)^2$. In this section, we propose several variants of the proposed attack to enable the attacker to select the probability p with finer granularity.

Two-Phase Attack

The proposed multi-term attack can be applied to all SASC protocols which use the basic decomposition. In this paragraph, we propose an attack that can be applied only to two-phase protocols such as Beguin and Quisquater's. The proposed attack in this section enables an attacker to be able to select p, the probability of successful attack, with finer granularity. It allows the attacker to guess a_is independently, instead of guessing the relation of a_is.

1) (In The First Phase) A malicious server returns $z'_i = M^{x_i-(h+1)^i} \bmod n$ instead of z_i to the client, where $0 \leq i \leq t-1$, and returns correct z_i for $i \geq t$.

2) (In The Second Phase) The server returns y'_p and y'_q instead of y_p and y_q: $y'_p = M^{\sigma_p + \sum_{i=0}^{t-1} a'_i (h+1)^i} \bmod n$ and $y'_q = M^{\sigma_q + \sum_{i=0}^{t-1} a'_i (h+1)^i} \bmod n$. Each a'_i is the value that the attacker supposes to be a_i.

If all a'_is are same as a_is, respectively, the correct signature is generated as $z'_p y'_p \bmod p + z'_q y'_q \bmod q \equiv S \bmod n$. The success probability p is $(h+1)^{-t}$, and the successful attack reduces the search space $\mathcal{P}$ to $p\mathcal{P}(= \frac{\mathcal{P}}{(h+1)^t})$. The average search space, $\mathcal{A}$, is $(h+1)^t + \frac{1}{(h+1)^t} \times \mathcal{P}$, and the lower bound of it is $2\sqrt{\mathcal{P}}$.

Guessing Partial Sums of a_is

The malicious server can also guess partial sums of a_is. For example, if the server wants to verify whether $a_0 + a_1$ is h, he can try a variant of the previous two-phase attack. In the first phase, he returns wrong values $z'_0 = M^{x_0-1} \bmod n$ and $z'_1 = M^{x_1-1} \bmod n$ instead of z_0 and z_1, respectively. In the second phase, he sends back $y'_p = M^{\sigma_p+h} \bmod n$ and $y'_q = M^{\sigma_q+h} \bmod n$ instead of y_p and y_q, respectively. If $a_0 + a_1$ is h, the correct signature is generated. This can be generalized easily.

If the server wants to attack only using the first phase, he can guess the ratio of the sum of several a_is to one of them. As it deviates from the protocol only in the first phase (which is the basic decomposition phase that is commonly used in the previous SASC protocols), it can be also used for attacking the previous one-phase SASC protocols like the proposed multi-term attack.

6 Conclusion

In this paper, we proposed several active attacks on SASC protocols. The proposed attacks can be applied to all of the existing SASC protocols. First, we modified the Beguin and Quisquater's server-aided RSA computation protocol in order to immunize it against Nguyen and Stern's attack. And we described the proposed attack using Beguin and Quisquater's scheme. The proposed attacks reduce the search space $\mathcal{P}$ to $2\sqrt{\mathcal{P}}$. Although it is the theoretical lower bound, it is sometimes possible in the real world. In addition, the proposed attacks threaten SASC protocols as the attacker can select an appropriate method according to his computing power and the number of chances to participate in the protocol as a malicious server. Therefore, the security parameters of the existing SASC protocols must be reconsidered (as much as twice) so that the protocols can be secure against our attack.

Acknowledgements

This work was supported by the KOSEF (Korea Science and Engineering Foundation) through the AITrc (Advanced Information Technology Research Center) and the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment)

References

[1] R. J. Anderson, "Attack on server assisted authentication protocols," *Electronics Letters*, vol. 28, no. 15, pp. 1473, 1992.

[2] P. Beguin and J. J. Quisquater, "Secure acceleration of DSS signatures using insecure server," in *Asiacrypt'94*, pp. 249–259, 1994.

[3] P. Beguin and J. J. Quisquater, "Fast server-aided RSA signatures secure against active attacks," in *Crypto'95*, pp. 57–69, 1995.

[4] J. Burns and C. J. Mitchell, "Parameter selection for server-aided RSA computation schemes," *IEEE Transactions on Computers*, vol. 43, no. 2, pp. 163–174, 1994.

[5] S. M. Hong, J. B. Shin, H. Lee-Kwnag, and H. Yoon, "A new approach to server-aided secret computation," in *International Conference on Information Secuirty and Cryptology (ICISC'98)*, pp. 33–45, 1998.

[6] S. Kawamura and A. Shimbo, "Fast server-aided secret computation protocols for modular exponentiation," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 778–784, 1993.

[7] S. Lee, S. M. Hong, H. Yoon, and Y. Cho, "Accelerating key establisment protocol in mobile communication," in *Information Security and Privacy*, LNCS 1587, pp. 51–63, Springer Verlag, 1999.

[8] C. H. Lim and P. J. Lee, "Security and performance of server-aided RSA computation protocols," in *Crypto'95*, pp. 70–83, 1995.

[9] C. H. Lim and P. J. Lee, "Server(prover/signer)-aided verification of identity proofs and signature," in *Eurocrypt'95*, pp. 64–78, 1995.

[10] T. Matsumoto, H. Imai, C. S. Laih, and S. M. Yen, "On verifiable implicit asking protocols for RSA computation," in *Auscrypt'92*, pp. 296–307, 1993.

[11] T. Matsumoto, K. Kato, and H. Imai, "Speeding up secret computations with insecure auxiliary devices," in *Crypto'88*, pp. 497–506, 1988.

[12] P. Nguyen and J. Stern, "The beguin-quisquater server-aided RSA protocol from crypto'95 is not secure," in *Advances in Cryptology - Asiacrypt'98*, LNCS 1514, pp. 372–379, Springer Verlag, 1998.

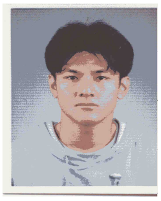
[13] B. Pfitzmann and M. Waidner, "Attacks on protocols for server-aided RSA computation," in *Eurocrypt'92*, pp. 153–162, 1992.

[14] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of ACM*, vol. 21, pp. 120–126, 1978.

[15] A. Shimbo and S. Kawamura, "Factorization attack on certain server-aided secret computation protocols for the RSA secret transformation," *Electronics Letters*, vol. 26, no. 17, pp. 1387–1388, 1990.

[16] S. M. Yen, "Cryptanalysis of secure addition chain for SASC applications," *Electronics Letters*, vol. 31, no. 3, pp. 175–176, 1995.

[17] S. M. Yen and C. S. Laih, "More about the active attak on the server-aided secret computation protocol," *Electronics Letters*, vol. 28, no. 24, pp. 2250, 1992.



Heeyoul Kim received the B.E. degree in computer science from Korea Advance Institute of Science and Technology (KAIST), South Korea, in 2000, the M.S. degree in computer science from KAIST, in 2002. He is currently working toward the Ph.D. degree at the Division of Computer Science, KAIST.

Younho Lee received the B.E. degree in computer science from Korea Advance Institute of Science and Technology (KAIST), South Korea, in 2000, the M.S. degree in computer science from KAIST, in 2002. He is currently working toward the Ph.D. degree at the Division of Computer Science, KAIST.

Seong-Min Hong received the B.E. degree in computer science from KAIST, South Korea, in 1994. He also received the M.S. degree and Ph.D degree in computer engineering from KAIST in 1996 and 2000, respectively. He is currently an research professor in the division of Computer Science at KAIST.

Hyunsoo Yoon received the B.E. degree in electronics engineering from SNU, South Korea, in 1979, the M.S. degree in computer science from KAIST, in 1981, and the Ph.D. degree in computer and information science from the Ohio State University, Columbus, Ohio, in 1988. From 1988 to 1989, with the AT & T Bell Labs. as a Member of Technical Staff. Since 1989 he has been a faculty member of Division of Computer Science at KAIST.