SASC(Server-Aided Secret Computation) 프로토콜에 대한 확률적 공격

홍 성민, 진 성기, 윤 현수
305-701 대전광역시 유성구 구성동 373-1
한국과학기술원 전산학과

Probabilistic Attack on Server-Aided Secret Computation Protocol

Seong-Min Hong, Sungkee Jean, Hyunsoo Yoon
Department of Computer Science, KAIST
Taejon, 305-701, KOREA
E-mail: smhong@camars.kaist.ac.kr

요약 : Matsumoto, Kato, 그리고 Imai가 [1]에서 제안한 SASC(Server-Aided Secret Computation) 프로토콜은 스마트 카드와 같이 계산능력이 현저히 부족한 장치로 하여금 서버의 도움을 받아 효율적으로 서명생성을 할 수 있도록 하는 유용한 프로토콜이다. 그러나, 그 과정에서 유출되는 정보를 통해 스마트 카드의 비밀정보를 알아낼 수 있는 많은 공격방법들이 제안되었다. Crypto'95에서 Beguin과 Quisquater는 기존의 모든 공격방법들에 대해 안전하면서도 효율적으로 RSA 서명을 수행할 수 있는 SASC 프로토콜을 제안하였다. 본 논문에서는 Beguin과 Quisquater의 프로토콜을 공격할 수 있는 능동적 공격(active attack) 방법을 제안한다. 본 논문에서 제안하는 공격방법은 $0 < p < 1$ 인 p에 대해 탐색영역의 크기를 $\mathcal{P}$에서 $\frac{1}{p} + p\mathcal{P}$로 줄이며, 이론적으로 제곱근 정도의 탐색영역 이득을 얻을 수 있고, 실제로도 충분한 위협이 될 수 있다. 본 논문에서 제안하는 공격방법은 기존의 모든 SASC 프로토콜에 적용 가능하다.

제 1 절 서론

암호 시스템을 구현하기 위해 해결해야 할 가장 중요한 문제들 중 하나는 비밀정보를 관리하는 방법이다. 특히, 공개키 암호 시스템을 사용할 경우 비밀정보의 크기가 너무 커서 사람이 기억하기 어렵기때문에, 사용자의 비밀정보를 기억할 수 있는 별도의 장치가 필수적이다. 이러한 목적으로 유용한 장치가 스마트 카드(smart card)이다.

스마트 카드는 프로세서가 부착되어 있는 플라스틱 카드이므로 지니고 다니기에 용이하며 하드웨어적으로 안전하기때문에 사용자의 비밀정보를 기억하는 장

치로 많이 이용되고 있다. 또한, 계산 능력도 지니고 있어 전자서명 수행이나 사용자 인증과 같은 암호학적 연산을 수행하는 데에 많이 이용된다. 특히, 전자상거래에서 지불 수단으로서의 스마트 카드는 필수적이다. 그러나 스마트 카드에 부착될 수 있는 프로세서는 하드웨어 기술의 한계로 인해 범용 프로세서에 비해 성능이 많이 떨어진다. RSA와 같은 공개키 방식의 전자서명은 많은 연산을 필요로 하므로 이러한 스마트 카드에서 효율적으로 수행하기는 힘들다 [2].

이러한 문제점을 해결하기 위해, 1988년 Matsumoto, Kato, 그리고 Imai에 의해 SASC(Server-Aided Secret Computation) 프로토콜이 제안되었다 [1]. 이는 RSA 서명을 스마트 카드에서 효율적으로 수행하기 위해 서버(카드 리더, 혹은 현금지급기 등이 서버가 될 수 있다)의 도움을 얻을 수 있도록 하는 프로토콜이다. SASC 프로토콜을 이용하면 계산속도가 수배 혹은 수십배까지 향상될 수 있어 현실적으로 효용성이 크다.

SASC의 큰 효용성으로 인해 그동안 많은 SASC 프로토콜들이 개발되어 왔으나, 그와 함께 이에 대한 다양한 공격방법들도 제안되어 왔다 [3, 4, 5, 6, 7, 8, 9, 10]. SASC 프로토콜에 대한 공격방법은 크게 수동적 공격(passive attack)과 능동적 공격(active attack)으로 나뉜다. 수동적 공격은 프로토콜의 진행에는 전혀 관계하지 않고, 단지 프로토콜을 수행함으로써 스마트 카드 외부에 주어지는 정보들만으로 비밀정보를 찾아내는 것을 말한다. 대표적인 수동적 공격에는 birthday attack의 원리를 활용한 Pfitzmann과 Waidner의 방법 [6]이 있는데, 이는 brute-force search에 필요한 탐색영역의 제곱근 만으로 공격에 성공할 수 있는 방법이다. 반면, 능동적 공격은 프로토콜에 직접 개입하여 프로토콜에 규정된 값과는 다른 값을 클라이언트에 전달함으로써 추가적인 정보를 얻는 것이다. 대표적인 능동적 공격으로는 소수들을 이용한 Anderson의 단일회전 공격법 [5], Shimbo와 Kawamura의 인수분해 공격법 [11], Lim과 Lee의 확장된 인수분해 공격법 [12], 그리고 Jacobi symbol을 이용한 Pfitzmann과 Waidner의 다회전 공격법 [6] 등이 있다. 수동적 공격은 공격자가 공격하고 있다는 사실이 발각되지 않는 반면 공격 효율성이 떨어지고, 능동적 공격은 공격효율성을 높일 수 있기는 하지만 발각위험이 크다.

Beguin과 Quisquater는 Crypto'95에서 기존에 알려진 모든 공격을 피할 수 있는 효율적인 SASC 프로토콜을 제안하였다 [13]. 이 프로토콜이 Pfitzmann과 Waidner의 다회전 공격을 피하는 방식은 비밀정보를 분할하는 방법을 매 번 다르게 하는 것이다. 구체적인 방법은 2 절에서 간략히 설명한다. 본 논문에서는 Beguin과 Quisquater 프로토콜의 안전성을 위협할 수 있는 새로운 능동적 공격방법을 제안한다. 본 논문에서 제안하는 공격방법은 비밀정보를 분할하는 데 사용되는 벡터를 악의의 서버로 하여금 유추할 수 있도록 조절하는 방식이다. 본 논문에서 제안하는 공격방법을 이용하면 $0 < p < 1$인 p에 대해 탐색영역을 $\mathcal{P}$에서 $\frac{1}{p} + p\mathcal{P}$로 줄일 수 있다. 이론적으로 $2\sqrt{\mathcal{P}}$까지 탐색영역을 줄일 수 있으며, 실제적으로도 사용되는 환경에 따라 p를 조절함으로써 충분한 위협이 될 수 있다. 또한,

본 논문에서 제안하는 공격방법은 비밀정보를 분할하는 방법이 같은 모든 SASC 프로토콜에 적용될 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 RSA 서명생성 방법과 Beguin과 Quisquater 프로토콜을 설명하며, 3절에서는 본 논문에서 제안하는 공격방법에 대해 설명한다. 4절에서는 제안 공격방법의 성능을 평가하고, 실제적인 위협에 대해 기술한다. 5절에서는 2단계 공격방법을 제안하고, 마지막으로 6절에서는 제안 공격방법을 피하는 방법에 대해 설명하고 결론을 맺는다.

제 2 절 관련연구

RSA computation 우선 RSA 서명생성 방법을 간략히 설명한다. 서명자는 매우 큰 소수 p, q를 구하고 $n = pq$를 계산한다. 그리고나서, $\phi(n)(= (p-1)(q-1))$과 서로 소인 임의의 정수 ν를 고른 후에 $s\nu \equiv 1 \bmod \phi(n)$이 성립하는 s를 찾는다. 이러한 초기 설정이 끝나면, 서명자는 n, ν를 공개하고, s는 자신만이 아는 개인키의 역할을 하게 된다. 임의의 메시지 M에 대한 서명은 $M^s \bmod n$이 된다. SASC 프로토콜의 목적은 서버의 도움을 받아서 임의의 M에 대해 $M^s \bmod n$을 계산하는 것이다.

SASC 처음 제안된 SASC의 기본 골격은 비밀정보 s를 여러개의 조각 (x_i and a_i, where $s = \sum_{i=0}^{m-1} x_i a_i \bmod \phi(n)$)들로 나누어 일부($x_i$)를 공개하고 나머지($a_i$)는 숨기는 방식을 사용한다 [1]. 그 이후에 발전된 방법들도 같은 기본 골격을 이용한다. 본 논문에서 제안하는 공격방법은 이러한 기본 골격을 사용하는 SASC 프로토콜에는 모두 적용할 수 있다.

능동적 공격 능동적 공격은 프로토콜에 직접 개입하여 악의의 서버가 프로토콜에 규정된 값과는 다른 값을 클라이언트에 전달함으로써 추가적인 정보를 얻는 것이다. SASC에 대한 능동적 공격은 단일회전 공격법과 다회전 공격법으로 나뉜다. 단일회전 공격법은 악의의 서버가 단 한 번의 공격으로 클라이언트의 비밀키를 알아낼 수 있는 공격으로서, Anderson의 소수를 이용한 공격법, Shimbo와 Kawamura의 인수분해를 이용한 방법, 그리고 Lim 과 Lee의 확장된 인수분해 공격방법들이 단일회전 공격법에 속한다. 하지만, 단일회전 공격법은 최종 서명확인 절차를 통해서 쉽게 막을 수 있다. 즉, 클라이언트가 서명결과를 서버에 전해주기 전에 공개키를 이용해서 서명결과가 올바른지를 확인하고, 올바른 서명결과에 대해서만 그 값을 전달하면 단일회전 공격법은 사용할 수 없다.

그러나, 최종서명 확인 절차로도 Pfitzmann과 Waidner의 Jacobi symbol을 사용한 다회전 공격법은 막을 수 없다. 다회전 공격법은 클라이언트가 서명을 생성해서 전해주는가 아닌가를 관찰함으로써 클라이언트의 비밀정보에 대한 부분 정보를 알 수 있도록 하는 방법이다. 이를 막기 위해 Beguin과 Quisquater는 매 번

비밀정보를 나누는 벡터를 새로 선택하는 방법을 사용하였다. Lim과 Lee도 같은 방법으로 다회전 공격법에 대해 안전하도록 할 수 있음을 보였다.

Beguin과 Quisquater의 프로토콜 다음은 Beguin과 Quisquater의 SASC 프로토콜이다 [13]. 다음 프로토콜에서 $l(a) = \lfloor \log_2 a \rfloor + 1$이라고 할 때 $t = \max\{l(p), l(q)\} - 1$이다. 그리고, $w_p = q(q^{-1} \bmod p)$이고 $w_q = p(p^{-1} \bmod q)$이다.

Client		Server
chooses randomly $\vec{x}$ and $\vec{a}$: $\vec{a} = a_0, a_1, \ldots, a_{m-1}$ $s.t.$ $a_i \in 0, \ldots, h$, $\vec{x} = x_0, \ldots, x_{m-1}$ $s.t.$ $l(x_i) \le t - \log_2(m \cdot h) - 2$.		
	M, n, and $\vec{x}$ $\longrightarrow$	computes $\vec{z}$: $\vec{z} = z_0, z_1, \ldots, z_{m-1}$ $s.t.$ $z_i = M^{x_i} \bmod n$.
	$\vec{z}$ $\longleftarrow$	
computes z_p and z_q : $z_p = \Pi_{i=0}^{m-1} z_i^{a_i} \bmod p$ and $z_q = \Pi_{i=0}^{m-1} z_i^{a_i} \bmod q$. computes $s_2 = s - s_1$, where $s_1 = \sum_{i=0}^{m-1} a_i x_i$. computes 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 $\varrho_q \in_R \{0, \ldots, p-2\}$.		
	σ_p, σ_q $\longrightarrow$	computes y_p and y_q : $y_p = M^{\sigma_p} \bmod n$ $y_q = M^{\sigma_q} \bmod n$
	y_p, y_q $\longleftarrow$	
computes $S = w_p S_p + w_q S_q \bmod n$, 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 transmits S.		

위의 프로토콜에서 s_1을 알게 되면 $\gcd(n, S - y_p M^{s_1} \bmod n) = p$이므로 결과적으로 스마트 카드의 비밀정보 s를 알 수 있게 된다. $s_1 = \sum_{i=0}^{m-1} a_i x_i$ 이므로 $\vec{a}$를 유추하여 맞추게 되면 s_1을 알 수 있다. $0 \le a_i \le h$ 이므로 아무런 정보가 없는 공격자가 수동적 공격을 수행한다면, $\frac{1}{2}(h+1)^m$가지 경우에 대해 탐색해야 한다. h와 m은 보안인수로서 예를 들면 $< h = 10, m = 19 >$ 정도의 수를 사용한다. 이러한 보안인수들은 공격자의 탐색영역을 일정범위 (e.g., 2^{64}) 이상이 되도록 유지하면서 클라이언트의 계산량을 최소화 할 수 있는 수들로 선택된다.

[13]에서 저자들은 그들의 프로토콜이 기존의 모든 수동적 공격과 능동적 공격에 대해 안전하다고 밝히고 있다. 수동적 공격에 대해서는 클라이언트가 $M^{s_1} \bmod$

n을 서버에게 알려주지 않기때문에 Pfitzmann과 Waidner의 birthday식 공격에 대해서 안전하다. 능동적 공격에 대해서는 최종 서명확인 절차가 프로토콜에 포함되어 단일회전 능동적 공격법들은 적용될 수 없다. 또한, 매 번 비밀정보를 나누는 벡터 $\vec{x}$와 $\vec{a}$를 새로 선택함으로써 Pfitzmann과 Waidner의 Jacobi symbol을 이용한 다회전 공격법으로부터 안전하다고 주장한다.

제 3 절 공격방법

3.1 기본 아이디어

능동적 공격의 정의는 악의의 서버가, 클라이언트에게 넘겨줄 부분 계산결과를 프로토콜에 규정되지 않은 방법으로 계산함으로써, 비밀정보 s에 대한 정보를 얻어내는 것을 말한다. Beguin과 Quisquater의 프로토콜의 경우에는 서버가 $\vec{z}, y_p$, 그리고/또는 y_q대신 $\vec{z}', y_p'$, 그리고/또는 y_q'를 클라이언트에 돌려줌으로써 s에 대한 유효한 정보를 얻어낼 수 있으면 능동적 공격이 성공한 것이다.

프로토콜의 마지막 단계에 계산된 서명이 올바른 서명인지를 확인하여 올바른 서명일 경우에만 그 값을 서버에게 전달하는 절차를 최종 서명확인(final signature checking)이라 한다. 이러한 최종 서명확인 절차를 SASC에 포함 함으로써, Anderson의 일회전 능동적 공격방법과 Shimbo과 Kawamura의 인수분해 공격은 피할 수 있다 [5, 11]. 하지만, Pfitzmann과 Waidner의 다회전 능동적 공격방법 [6]에 의하면 최종 서명확인 만으로는 안전하지 못함이 확인 되었다. 따라서, 이러한 공격방법들을 모두 피하기 위해 Beguin과 Quisquater는 매 번 비밀정보를 분할하는 벡터 $\vec{x}$와 $\vec{a}$를 다르게 선택하는 방법을 이용하였다 [13].

그러나, 매 번 벡터 $\vec{x}$와 $\vec{a}$를 다르게 선택한다고 해서 능동적 공격을 피할 수 있는 것은 아니다. 수동적 공격에 의한 탐색영역을 계산할 때, 공격대상이 되는 비밀값의 범위를 탐색영역으로 설정한다. 즉, 탐색영역을 모두 검색하면 확률 1로, 무조건 공격에 성공할 수 있다. 매번 벡터 $\vec{x}$와 $\vec{a}$가 바뀌는 시스템에서 $\vec{a}$를 매 번 유추할 수 있다면 확률 1은 아니더라도 매우 높은 확률로 성공할 수 있다.

주사위 놀이의 예를 들어 보면, '갑'은 주사위를 던지고 '을'은 어떤 값이 나왔는지를 맞춘다고 가정한다. 수동적 공격은 '갑'이 주사위를 한 번 던지고 '을'은 그 값이 어떤 값인지를 매 번 유추하는 경우와 같다. 주사위의 눈금은 6가지이므로 '을'이 6번을 유추하면 반드시 그 중 한 번은 맞는다. 매 번 벡터를 바꾸는 것에 대한 능동적 공격은 '갑'이 매 번 주사위를 던지고 그 때마다 '을'은 한 번씩의 유추할 기회만 주어지는 경우와 같다. 이러한 경우, 6번을 시도한다 하더라도 반드시 맞출 수 있는 것은 아니다. 하지만, $0.67(= 1 - (5/6)^6)$의 확률로 맞출 수 있다.

일반적인 확률 p에 대해 계산해 보면, 공격자가 유추하여 맞출 확률을 p라고 할 때, 수동적 공격의 경우 $\frac{1}{p}$번을 시도하면 확률 1로 맞출 수 있다. 능동적 공격의 경우 $\frac{1}{p}$번을 시도하면 $1 - (1-p)^{\frac{1}{p}} (= p_a)$의 확률로 성공할 수 있다. p_a의 범위는

다음과 같다.

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

본 논문에서 제안하는 공격방법의 기본 아이디어는 공격자가 a_i들 간의 관계를 유추하여 성공시에는 올바로 계산된 서명이 생성되도록 함으로써 a_i들을 찾기 위한 탐색영역을 줄이는 것이다. 그렇게 함으로써, 매 번 분할벡터 $\vec{x}$와 $\vec{a}$가 새로 선택되더라도 원하는 확률로 공격에 성공할 수 있도록 설계할 수 있다.

3.2 2항 공격방법

2절에서 설명했듯이 Beguin과 Quisquarter의 프로토콜에서 a_i들을 알게 되면 비밀정보 s가 유출된다. 본 절에서는 a_i들의 관계를 유추하는 공격방법의 원리를 설명하기 위해 z_0와 z_1의 두 항만을 이용해서 공격하는 2항 공격방법을 기술한다. 2항 공격방법의 목표는 a_1과 a_0의 비율인 $r_{1,0}$를 유추하여 탐색영역을 줄이는 것이다. $r_{i,j}$는 $\frac{a_i}{a_j}$를 나타낸다.

우선, 악의의 서버는 $l = lcm(1,2,\ldots,h)$를 계산한다. 그리고, z_0와 z_1 대신 다음과 같은 z_0'과 z_1'을 보낸다.

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

식 (1)에서 a_0'과 a_1'은 악의의 서버가 각각 a_0와 a_1이라고 추측하는 값을 의미한다. $a_0'|l$이므로 z_0'의 지수인 $(1+l\frac{a_1'}{a_0'})x_0$는 항상 정수가 된다.

$2 \leq i \leq m-1$인 i들에 대해서는 z_i를 제대로 계산해서 보내고, 그 이후의 모든 과정을 프로토콜에 규정된 대로 수행한다. 따라서, 악의의 서버가 유추한 $\frac{a_1'}{a_0'}$이 $r_{1,0}$와 같으면, $z_0'^{a_0} \times z_1'^{a_1} \times \prod_{i=2}^{m-1} z_i^{a_i} \equiv M^{s_1} \pmod n$ 이므로, 올바른 서명을 생성하게 되어 최종 서명확인 과정을 무사히 통과할 수 있다. 즉, 올바른 서명이 생성된 것은 악의의 서버가 유추한 $\frac{a_1'}{a_0'}$이 $r_{1,0}$와 같음을 의미한다[1].

공격에 성공한 악의의 서버는 $r_{1,0}$를 알고 있으므로, a_i들을 탐색할 때 $a_1 = a_0 \times r_{1,0}$이므로 a_1은 탐색할 필요가 없다. 또한, $0 \leq a_0 \times r_{1,0} \leq h$인 정수이어야 하므로 a_0가 가질 수 있는 값의 범위도 $r_{1,0}$의 값에 따라 제한된다.

$\frac{a_1'}{a_0'} = r_{1,0}$일 확률을 p_0로 놓으면, 악의의 서버가 능동적 공격에 성공할 확률은 p_0이고 성공 했을 때의 탐색영역은 $\mathcal{P}$에서 $p_0 \times \mathcal{P}$로 줄어든다. p_0의 계산은 다음 절에서 다룬다.

[1] 식 (1)에서 보면 a_0를 0으로 유추할 수는 없다. 그런 경우는 z_0'과 z_1'의 식을 바꾸어 전개하면 a_0'을 0으로 유추하여 z_0'과 z_1'을 계산할 수 있다. 또한, a_0와 a_1이 모두 0이면 어떠한 z_0'과 z_1'에 대해서도 올바른 서명이 생성된다. 따라서, 공격이 성공하면 a_0'과 a_1'이 모두 0일 수 있다.

3.3 다항 공격방법

2항 공격방법을 일반화 시켜서 t개의 항에 대해 적용시킬 수 있다. $0 \leq i \leq t-1$에 대해 z_i들 대신 z'_i들을 다음과 같이 계산하여 보낸다. t는 짝수이다.

$$\begin{aligned} &z'_{2k} = M^{(1+l\frac{a'_{2k+1}}{a'_{2k}})x_{2k}} \bmod n, \text{ and} \\ &z'_{2k+1} = M^{x_{2k+1}-lx_{2k}} \bmod n, \\ &where \;\; 0 \leq k \leq \tfrac{t}{2}-1. \end{aligned} \tag{2}$$

$t \leq i \leq m-1$인 i들에 대해서는 z_i들을 제대로 보내고, 그 이후의 모든 과정을 프로토콜에 규정된 대로 수행한다. 식 (2)에서 모든 $\frac{a'_{2k+1}}{a'_{2k}}$들이 각각 $r_{2k+1,2k}$와 같으면, 최종 서명확인 과정을 통과하는 올바른 서명이 생성된다. $\frac{a'_{2k+1}}{a'_{2k}}$과 $r_{2k+1,2k}$가 같을 확률을 p_k로 놓으면 능동적 공격이 성공할 확률은 $\prod_{k=0}^{\frac{t}{2}-1} p_k$이다. 즉, $\prod_{k=0}^{\frac{t}{2}-1} \frac{1}{p_k}$번 공격을 시도하면 $1-\frac{1}{e}$ 이상의 확률로 성공할 수 있다.

a_i들을 탐색할 때 $a_1 = a_0 \times r_{1,0}$이므로 a_1은 탐색할 필요가 없다. 또한, $0 \leq a_0 \times r_{1,0} \leq h$인 정수이어야 하므로 a_0가 가질 수 있는 값의 범위도 $r_{1,0}$의 값에 따라 제한된다.

t항 공격방법에 성공한 악의의 서버는 $0 \leq k \leq t/2-1$인 k에 대해 $r_{2k+1,2k}$들을 알고 있다. 따라서, a_i들을 탐색할 때 $a_{2k+1} = a_{2k} \times r_{2k+1,2k}$이므로 $a_1, a_3, \ldots, a_{t-1}$들은 탐색할 필요가 없다. 또한 a_{2k}들이 가질 수 있는 값의 범위도 $r_{2k+1,2k}$의 값에 따라 제한된다. 결과적으로 탐색영역이 $\prod_{k=0}^{\frac{t}{2}-1} p_k \times \mathcal{P}$로 줄어든다.

제 4 절 위험 분석

본 절에서는 우선 본 논문에서 제안하는 공격방법을 이용할 경우의 이론적 최저 탐색영역을 계산한다. 그리고, 그것이 실제로 SASC 프로토콜을 사용할 때 주는 위협에 대해 논한다.

4.1 최저 탐색영역

수동적 공격과 능동적 공격의 탐색영역을 일원화 하기 위해, 탐색영역의 정의를 새로 한다[2].

[2]공격이 성공할 확률이 p일 때, $5 \times \frac{1}{p}$번을 시도하면 약 $0.99(\approx 1-\frac{1}{e^5})$ 이상의 매우 높은 확률로 성공할 수 있다. 즉, $c \times \frac{1}{p}$번을 시도하면 $1-\frac{1}{e^c}$ 이상의 확률로 성공할 수 있다. 그러나, 매 번의 공격은 Bernoulli trial로서 binomial distribution을 따르기때문에 $c \times \frac{1}{p}$번의 공격에 대한 기대값은 c이다. 즉 $c \times \frac{1}{p}$번 공격을 시도하면 평균 c개에 대해서 성공한다. 따라서, 탐색영역의 새로운 정의에서는 기대값이 1이 되는 횟수를 선택하기 위해 확률을 $1-\frac{1}{e}$로 하였다.

정의 1 *SASC에서의* **탐색영역**은 *비밀 정보를 알아낼 확률을* $1-\frac{1}{e}$*보다 크게 하기 위해 공격자가 탐색해야 하는 경우의 수이다.*

SASC에서 s_1을 알아내기 위한 수동적 공격의 탐색영역을 $\mathcal{P}$라고 하면, 능동적 공격을 제한없이 수행할 수 있다고 가정할 때의 총 탐색영역 $\mathcal{A}$는 다음과 같다:

$$\mathcal{A} = (\prod_{k=0}^{t/2-1} \frac{1}{p_k}) + (\prod_{k=0}^{t/2-1} p_i) \times \mathcal{P}. \tag{3}$$

a_i들은 모두 0부터 h 사이의 정수이므로, $\gcd(a_{2k}, a_{2k+1}) = 1$인 a_i들에 대해, $\frac{a_{2k+1}}{a_{2k}}$의 가짓수는 $\lfloor \frac{h}{\max\{a_{2k}, a_{2k+1}\}} \rfloor$이다. 따라서, p_k는 다음과 같이 표현될 수 있다.

$$\frac{1}{(h+1)^2} \leq p_k = \frac{\lfloor \frac{h}{\max\{a_{2k}, a_{2k+1}\}} \rfloor + 1}{(h+1)^2} \leq \frac{1}{h+1} \tag{4}$$

공격자의 입장에서 a_i들은 모두 1부터 h 사이의 값을 동일한 확률로 가정할 수 있으므로, $\gcd(a_{2k}, a_{2k+1}) = 1$인 a_i들에 대해 $\frac{a_{2k+1}}{a_{2k}}$이 가질 수 있는 가짓수 만큼의 비율로 유추하게 된다. 따라서, p_k들의 평균 $\tilde{p_k}$은 다음과 같이 계산할 수 있다.

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

정의 1에 의한 탐색영역의 크기는 다음과 같다.

$$\mathcal{A} = (\tilde{p_k})^{-t/2} + (\tilde{p_k})^{t/2} \times \mathcal{P}. \tag{5}$$

$\mathcal{A}$가 최소가 되려면 $\mathcal{A}$를 이루고 있는 두 항인 $(\tilde{p_k})^{-t/2}$과 $(\tilde{p_k})^{t/2} \times \mathcal{P}$가 같아야 한다. 그러한 최적의 t를 구하면, $t_{opt} = -\frac{1}{2} \times \log_{\tilde{p_k}} \mathcal{P}$가 된다. $0 < \frac{t_{opt}}{m} < 1$이므로 t_{opt}는 항상 존재한다. t_{opt} 개의 항을 이용해 능동적 공격을 수행할 경우의 탐색영역은 다음과 같다.

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

즉, 능동적 공격을 무제한 수행할 수 있다고 가정하면, 필요한 탐색영역은 $\mathcal{P}$에서 $2\sqrt{\mathcal{P}}$로 줄어든다. [13]에서 저자들은 2^{64}의 탐색영역을 지니도록 보안인수들을 설정하였는데, 본 논문에서 제안하는 공격방법을 이용하면 2^{33}번의 탐색만으로 비밀정보 s가 누출된다. 2^{33}번의 탐색은 현재의 PC로도 아주 쉽게 수행될 수 있다.

4.2 실제적 위험

앞 절에서 이론적 최저 탐색영역을 계산하기 위해 능동적 공격을 무한정 수행할 수 있다고 가정하였다. 그러나, 실제 시스템에서는 제한된 횟 수 이상의 서명생성에 실패하면 경보음을 낼 수도 있고, 그러한 횟수가 잦은 단말기 혹은 상점이나 사업자들은 의심을 받게 된다. 따라서, 실제로 능동적 공격을 무제한 가능하다고 가정하기는 힘들다.

하지만, 카드를 분실한 경우 혹은 카드 주인이 위협을 받는 경우와 같은 특수한 상황에서는 능동적 공격이 무제한으로 가능할 수 있다. 또한, [6]에서 Pfitzmann과 Waidner가 지적하였듯이 스마트 카드의 특성상 올바른 결과가 나왔는지 혹은 그렇지 못했는지를 정확히 알아내는 것이 쉬운일은 아니다.

따라서, 능동적 공격을 수행하는 것이 발각될 위험이 비록 크긴 하지만 충분히 있을 수 있다고 가정해야 한다. 악의의 서버는 자신이 동원할 수 있는 계산능력(computing power)과 자신이 공격을 시도할 수 있는 스마트 카드와의 transaction의 수를 감안해서 공격방법을 정할 수 있다.

예를들어, [13] 에서 저자들은 2^{64}의 탐색영역을 유지하면서 계산량을 최소로 할 수 있는 보안인수 설정들 중 하나로 $< h = 10, m = 19 >$를 선택하였다. 이러한 시스템에서 2^{50}-trial/day의 계산능력(2^{50}번의 시도를 하루에 수행할 수 있는 계산능력)을 지닌 악의의 서버를 가정한다. 이 서버는 수동적 공격만으로는 2^{14}일, 즉 10년 이상을 공격해야 한다. 그러나, 본 논문에서 제안하는 능동적 공격을 수행할 경우, $t = 4$로 놓고 공격을 수행하면, 평균 16,000 번의 transaction만으로 하루만에 공격에 성공할 수 있다. 또한, $t = 2$로 놓으면 평균 100번의 transaction 만으로 약 1달만에 공격에 성공할 수 있다. 이는 특정 공격대상이 한 악의의 서버에 평균 100번 정도 접속하면 비밀정보가 드러남을 의미한다. 또한, 100 명 중 1명 꼴로 비밀정보가 드러남을 의미하기도 한다.

본 논문에서 제안하는 능동적 공격이 성공했을 경우에는 클라이언트가 공격사실을 모르기때문에 오랜 시간동안(클라이언트의 비밀키는 실제로 자주 바꾸기가 쉽지 않기때문에 유효기간이 보통 1년 이상 된다) 계속 몰래 탐색(수동적 공격)할 수 있다. 또한, 이와함께 특정 서버에서 능동적 공격이 수행되었다는 사실이 드러날 경우 그 서버를 이용한 모든 클라이언트는 위험가능성을 내포하고 있어 혼란을 야기할 수 있다.

제 5 절 성공확률 미세조정

본 논문에서 제안한 공격방법에서 공격에 성공할 확률 p는 공격자가 원하는대로 조절할 수 있다. 하지만 p는 $(h+1)^2$의 배수가 되어야만 한다. 본 절에서는 p가 가질 수 있는 수의 범위를 넓힐 수 있는 다른 몇 가지 방법들을 설명한다.

2단계 공격법 3절에서 설명한 공격방법은 Beguin과 Quisquater의 프로토콜 뿐 아니라 Matsumoto 등이 제안한 방법의 기본 골격을 이용하는 모든 SASC 프로토콜에 적용할 수 있다. 이번 절에서는 Beguin과 Quisquater의 방법처럼 2단계로 진행되는 프로토콜에 적용할 수 있는 공격방법을 제안한다. 이번 절에서 제안하는 2단계 공격방법은 a_i들을 각각 독립적으로 유추할 수 있도록 함으로써, a_i들의 관계를 유추하는 다항공격방법에 비해, 공격에 성공할 확률 p를 보다 더 세밀하게 조절할 수 있다.

1. 우선 악의의 서버가 $0 \le i \le t-1$인 i에 대해 z_i 대신 $z_i' = M^{x_i-(h+1)^i} \bmod n$을 클라이언트에 보낸다. 그리고, $t \le i \le m-1$인 i에 대해서는 z_i를 그대로 보낸다.
2. 클라이언트가 보낸 σ_p, σ_q 에 대해 y_p, y_q 대신 $y_p' = M^{\sigma_p+\sum_{i=0}^{t-1} a_i'(h+1)^i} \bmod n$과 $y_q' = M^{\sigma_q+\sum_{i=0}^{t-1} a_i'(h+1)^i} \bmod n$을 보낸다. a_i'은 공격자가 a_i라고 추측하는 값을 의미한다.

공격자가 추측한 a_i'과 실제 a_i값이 일치하는 경우, $z_p'y_p' \bmod p + z_q'y_q' \bmod q \equiv S \pmod n$이므로 클라이언트는 제대로 된 서명을 생성하게 되어 최종 서명확인 과정을 통과하게 된다. 최종 서명확인을 통과할 확률 p는 $(h+1)^t$이고, 통과할 경우 탐색영역은 $p\mathcal{P}(=\frac{\mathcal{P}}{(h+1)^t})$로 줄어든다. 이론적 최저 탐색영역은 다항 공격방법과 마찬가지로 $2\sqrt{\mathcal{P}}$이다.

부분합 유추 공격법 공격자는 a_i들의 부분합을 유추할 수 있도록 클라이언트에 돌려주는 결과값을 조절할 수 있다. 예를 들어, 공격자가 $a_0 + a_1$이 h인지 아닌지를 알려고 한다면, 그는 2단계 공격법을 약간 변형함으로써 알 수 있다. 우선 첫번째 단계에서 $z_0' = M^{x_0-1} \bmod n$과 $z_1' = M^{x_1-1} \bmod n$을 z_0와 z_1 대신 보내준다. 두번째 단계에서는 y_p와 y_q 대신 $y_p' = M^{\sigma_p+h} \bmod n$과 $y_q' = M^{\sigma_q+h} \bmod n$을 클라이언트에 돌려준다. 만약 $a_0 + a_1$이 h라면 올바른 서명이 생성된다. 이러한 방법은 $a_0 + a_1 + \cdots + a_k$의 값을 유추할 수 있도록 쉽게 확장할 수 있다.

만약 공격자가 첫번째 단계만을 이용하려고 한다면 (그렇게 함으로써, 기존의 SASC 에도 적용할 수 있도록 할 수 있다.), a_i들의 합과 그들 중 한 값의 비율을 유추할 수 있다. 예를 들면, $\frac{a_0+a_1+\cdots+a_k}{a_0}$를 유추할 수 있다. 이 또한 쉽게 확장되어 일반적인 경우에 대해 적용할 수 있다.

제 6 절 결론

본 논문에서는 Beguin과 Quisquater의 프로토콜처럼 매 번 비밀벡터를 바꾸는 SASC 프로토콜의 안전성을 위협할 수 있는 능동적 공격방법을 제안하였다. 본 논문에서 제안하는 공격방법은 $0 < p < 1$ 인 p에 대해 탐색영역의 크기를 $\mathcal{P}$에서

$\frac{1}{p} + p\mathcal{P}$로 줄이며, 이론적으로 탐색영역을 $\mathcal{P}$에서 $2\sqrt{\mathcal{P}}$로 낮출 수 있음을 보였다. 실제로도 공격자가 동원할 수 있는 계산능력과 공격자가 서버로서 프로토콜에 참여할 수 있는 transasction의 수를 감안해서 적절한 방법으로 공격이 가능함을 보였다. 따라서, 기존의 SASC 프로토콜의 보안인수를 증가시켜야 한다.

참고 서적

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

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

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

[4] 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, p. 2250, 1992.

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

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

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

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

[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] S.Kawamura and A.Shimbo, "Fast server-aided secret computation protocols for modular exponentiation," *IEEE JSAC*, vol. 11, no. 5, pp. 778–784, 1993.

[11] 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.

[12] 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.

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