Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.2, Apr. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.2.327

RSA 문제와 동등한 안전성을 갖는 온라인/오프라인 서명 기법*

최 경 용,† 박 종 환‡
상명대학교

On-Line/Off-Line Signature Schemes with Tight Security Reduction to the RSA Problem*

Kyung-yong Choi,† Jong Hwan Park‡
Sangmyung University

요 약

온라인/오프라인(On-line/off-line) 서명은 오프라인 단계에서 서명생성에 필요한 무거운 연산을 수행하고, 온라인 단계에서 간단한 연산만으로 최종 서명을 완성하는 기법이다. 이는 다수의 사용자에게 즉각적인 서명 응답을 해야 하는 응용환경에 적합하다. 본 논문에서는 RSA 문제에 기반한 새로운 온라인/오프라인 서명기법을 두 가지로 제안한다. 첫 번째 기법은 온라인 서명 시 고정된 밑수에서의 지수승이면 되고, 두 번째 기법은 해쉬연산과 같은 매우 간단한 계산만으로 온라인 서명을 완성할 수 있다. 두 서명의 안전성은 모두 RSA 문제로 환원되는데, 랜덤 오라클 모델에서 안전성 손실 없이 증명이 된다.

ABSTRACT

On-line/off-line signature is a technique for performing heavy computations required for signature generation in the off-line stage and completing the final signature by a simple operation in the online stage. This is suitable for application environments that require immediate signing responses to multiple users. In this paper, we propose two new on-line/off-line signature schemes based on RSA problem. The first technique can generate a signature with a fixed base exponentiation when signing online, and the second technique can complete an online signature with a very simple calculation such as a hash operation. The security of both signatures is based on the RSA problem, which is proven to be tightly secure without security loss in the random oracle model.

Keywords: On-line/off-line signature, RSA problem, tight security reduction

I. 서 론

온라인/오프라인(On-line/off-line) 서명[1]은 서명알고리즘을 이원화하여 오프라인 단계에서 비용이 많이 드는 연산을 사전에 수행하고, 온라인 단계에서는 매우 간단한 연산만으로 해당 메시지에 대한 최종 서명을 생성하는 기법이다. 이 서명기법은 하나의 서버가 대규모 사용자들에게 전자서명을 생성해서 전달해야 하는 응용환경에서 유용하다. 예를 들어, 최근 TLS(Transport Layer Security) v1.3 표준[2]에서는 디피-헬만(Diffie-Hellman) 키 교환을 중심으로 사용자와 서버가 보내는 디피-헬만 값을

Received(01. 09. 2018), Modified(03. 09. 2018), Accepted(03. 10. 2018)
* 본 연구는 2017년도 상명대학교 교내연구비를 지원받아 수행하였음.
† 주저자, vdqvdw@naver.com
‡ 교신저자, jhpark@smu.ac.kr(Corresponding author)

서명하여 응답하는 것이 핸드쉐이크(Handshake) 프로토콜[1]의 핵심적인 과정이다. TLS v1.2[3]에서도 임시(Ephemeral) 디피-헬만 방식의 경우 디피-헬만 값을 서명하는 과정이 여전히 요구된다. 이 경우 서버는 다수의 사용자와 TLS 보안채널을 형성하는 과정에서 온라인상에서 매우 빠른 서명을 생성하는 것이 필요하다. 온라인/오프라인 서명을 사용할 경우, 서버는 여유 있는 시간에 다수의 오프라인 서명(이를 토큰(token)이라 함)을 사전생성해서 저장하고 온라인 서명에서는 사용자가 요청하는 메시지에 대해 저장된 토큰을 활용, 매우 빠른 서명을 생성하는 것이 가능하다.

온라인/오프라인 서명을 설계하는 일반적인 방법은 몇 가지가 제시되었다. 첫 번째 방식은 일회용 서명을 이용하는 방법[1]으로, 사전에 메시지와 무관하게 일회용 공개키와 서명키를 생성한 후 일회용 공개키를 메시지로 취급하고 (본래의) 서명을 생성한다. 이후 메시지가 입력되면 일회용 서명키를 이용하여 서명을 한 후 두 개의 서명을 합해서 하나의 서명으로 출력하는 방식이다. 이 방식은 최종서명에 일회용 공개키 및 두 종류의 서명이 포함되므로 서명길이가 길어지는 단점이 있다. 두 번째 방식은 카멜레온 해쉬함수(Chameleon hash function)[4]를 이용하는 방법[5]으로, 카멜레온 해쉬값을 생성한 후 메시지와 무관하게 그 해쉬값에 (본래의) 서명을 생성한 후 메시지가 입력되면 카멜레온 해쉬함수의 비밀값(즉, 트랩도어(trapdoor)를 말함)을 이용하여 메시지에 대응하는 해쉬 충돌쌍을 찾고 최종 서명을 완성하는 방식이다. 이 방식은 공개키에 카멜레온 해쉬함수의 공개키까지 포함되므로 전체 공개키가 길어지는 단점이 있으며, 서명에 공개키까지 전송하는 경우 결국 일회용 서명방식과 마찬가지로 전체 서명이 길어지는 단점이 있다.

본 논문에서는 위 설계원리를 따르지 않고 RSA 기반에서 직접 온라인/오프라인 서명이 가능한 두 개의 기법을 제안한다. 현재까지 제안된 RSA 기반 서명기법들, 즉, RSA-FDH[6]와 RSA-PSS[7]은 (위의 비효율적인 변형방법을 따르지 않는 경우) 온라인/오프라인 서명으로 변형하는 것이 쉽지는 않다. 그 이유는 두 개의 서명 모두 메시지가 입력되고 인코딩이 된 후에 서명키로 지수승을 하기 때문이다. 즉, 메시지와 무관하게 사전계산을 하는 것이 현 상태로는 불가능하다. 본 논문에서는 이러한 문제를 극복해서 RSA 기반 서명에서도 온라인/오프라인 서명의 직접적인 설계가 가능하다는 것을 보인다. 온라인 서명에서의 연산량을 설명하면, 첫 번째 기법은 온라인 단계에서 고정된 밑수에서의 지수승만으로 서명생성이 가능하고, 두 번째 기법은 지수승도 필요 없이 단순한 연산만으로 서명생성이 가능하다. (표준 가정이 아닌) 더 강한 Strong RSA 문제에 기반한 기법[8]이 제시되었으나, [8]에서는 서명 생성 시마다 새로운 소수를 생성해야 하는 단점이 있다.

온라인/오프라인 서명방법으로 제안된 두 개의 서명기법 모두 RSA 문제에 효율적인 환원관계(Reduction)로 증명됨을 보인다. 즉 증명과정에서 안전성 손실(Security loss)없이 RSA 문제의 어려움에 거의 동등하게 환원됨을 보일 수 있다. 이러한 증명효율성은 실제 기법설계를 위한 변수 설정에 영향을 미치는데, 보안상수(Security parameter)가 추구하는 안전성 레벨보다 (안전성 손실을 고려해서) 더 높게 설정되어야 한다. 자연스럽게 안전성 손실이 적도록 증명된 기법이 더 바람직하다. 기존 기법에서는 RSA-FDH의 경우 공격자의 해쉬 질의 수만큼 안전성 손실이 발생함에 비해, RSA-PSS의 경우 RSA 문제에 효율적으로 환원되는 것이 증명되었다. 본 논문에서 제시하는 증명방법은 RSA 문제에 효율적으로 환원하려는 기타의 기법들에도 도움이 될 것이다.

II. 배경지식

2.1 전자서명 정의

전자서명은 세 개의 알고리즘 ($KeyGen$, $Sign$, $Vrfy$)로 구성된다.

- **키생성(KeyGen)**(k): 보안상수 k을 입력받고 서명검증용 공개키(PK)와 서명생성용 개인키(SK)를 출력한다.
- **서명(Sign)**(SK, m): 서명키 SK와 메시지 m을 입력받고 서명 결과인 σ를 출력한다.
- **검증(Vrfy)**(PK, m, σ): 검증키 PK와 메시지 m, 그리고 서명 σ을 입력받고 비트 b를 리턴(return)한다. 만약 $b=1$이면 서명은 유효

1) 키 교환 시 전방향 안정성(Forward secrecy)를 위해 디피-헬만 키교환과 전자서명을 사용한다. TLS v1.2와 같은 공개키 암호 방식의 키 교환 모드는 고려되지 않는다.

(valid)하고, $b=0$이면 서명은 무효(invalid)라는 의미이다.

정확성(correctness). $KeyGen(k)$로 생성된 (PK, SK)에서, 모든 메시지 m에 대해 올바르게 생성된 서명 $Sign_{SK}(m)$은 항상 검증 알고리즘에 의해서 $Vrfy(PK, m, Sign_{SK}(m))=1$이다.

2.2 전자서명 안전성 정의

전자서명 위조(forgery)란 서명자의 공개키 (PK)에 대해서 이전에 서명자에 의해 생성된 적 없는 메시지 m에 대한 유효한 서명 σ을 만드는 것이다. 여기서 전자서명이 위조 공격에 안전하다는 것은, 임의의 공격자가 그가 선택한 메시지들에 대해서 서명결과를 (서명 오라클을 통해) 얻었더라고 새로운 메시지에 대한 서명을 위조할 수 없어야 함을 의미한다. 이를 선택 메시지 공격[9]에 대한 안전성으로 정의한다. 다음의 공격실험 $\mathrm{Exp}_{\Pi}^{uf}(A)$로 정의된다.

Π=(KeyGen, Sign, Vrfy)를 서명이라 하자. 공격자 A, 보안상수 k에 대해 다음 실험을 수행한다.

1. $KeyGen(k)$으로 (PK, SK)를 얻는다.
2. 공격자 A는 PK를 받고 오라클 $Sign_{SK}()$에 질의한다. 그리고 A는 (m^*, σ^*)를 리턴한다.
3. Q를 이전에 A가 오라클에게 질의한 메시지 집합이라고 하자. 만약 (1) $Vrfy(m^*, \sigma^*)=1$이고 (2)$m^* \notin Q$ 일 때에만 A는 '위조를 성공했다'라 하고 이 경우 실험의 결과로 1을 리턴한다.

공격자 A가 위의 실험에서 위조에 성공할 확률을 다음과 같이 정의한다.

$$Adv_{\Pi}^{uf}(A) = \Pr[\mathrm{Exp}_{\Pi}^{uf}(A)=1].$$

정의 1. 만약 모든 공격자가 최대 t_A시간 동안 최대 q_S개의 질의를 할 때 서명 Π를 최대 ϵ_A확률로 깨지 못한다면 서명 Π는 선택 메시지 공격에 대해서 (t_A, ϵ_A, q_S)-안전하다.

2.3 RSA 가정

보안상수 k을 입력받아 (N, e, d)를 출력하는 다항식 시간 알고리즘 $GenRSA$가 있다고 하자. 이 때 N은 두 개의 (거의 동일한 비트 사이즈를 갖는) 강한 소수(strong prime)들의 곱이라 한다. 즉, $N=pq$일 때, 서로 다른 소수 p_1, q_1에 대해서 $p = 2p_1+1$, $q = 2q_1+1$ 이다. 그리고 e와 d값은 $\gcd(e, \phi(N))=1$, $ed=1 \pmod{\phi(N)}$ 이 성립한다. 소인수분해가 알려지지 않은 N, 그리고 e를 알고 있을 때 $x^e=y \pmod{N}$ 를 만족하는 $y^{1/e} \pmod{N}$ 을 계산하는 것이 RSA 문제이다. 보다 정확하게 RSA 문제[6][7]는 다음과 같은 실험으로 정의된다.

알고리즘 B에 대해서 다음의 실험을 수행한다.

1. $GenRSA$을 수행해 (N, e, d)를 얻는다.
2. 변수 $y \in Z_N^*$를 랜덤하게 선택한다.
3. B는 (N, e, y)를 받고 $x \in Z_N^*$를 리턴한다.
4. $x^e=y \pmod{N}$ 일 때 실험결과로 1을 내고 아니면 0을 리턴한다.

알고리즘 B가 RSA 문제를 푸는 확률을 다음과 같이 정의한다.

$$Adv^{RSA}(B)=\Pr[B(N, e, y)= y^{1/e}]$$

정의 2. RSA 문제가 (t_B, ϵ_B)-안전하다는 것은 다음과 같이 정의된다: 만약 모든 (확률론적) 다항식 시간 알고리즘 B가 최대 t_B시간 동안 RSA 문제를 푼다면, $Adv^{RSA}(B) < \epsilon_B$라 한다.

III. On-line/off-line RSA 서명 기법 1

3.1 서명 기법 설계

약간 수정된 $GenRSA$ 알고리즘을 먼저 기술한다. 보안상수 입력 k에 대해서 (N, e, d, g) 를 생성한다. 여기서 N은 강한 소수(strong prime) p, q의 곱이다. 즉, 서로 다른 소수 p_1, q_1에 대해서 $p = 2p_1+1$, $q = 2q_1+1$ 이다. 그리고 e는 $\ell+1$비트 보다 큰 어떤 소수이다. 즉, $e > 2^{\ell+1}$이다. (여기서 ℓ의 크기는 안전성 증명에 의해 결정된다.) 특이한 점은 위수가 p_1q_1인 생성원 g를 추가적으로 선택한다. g는 Z_N^* 의 부분군을 구성하는 생

성원이고, Z_N^*에서 이차잉여(Quadratic Residue) 그룹 QR_N의 생성원이다.

전자서명 알고리즘 $\Pi=(Gen,\ Sign, Vrfy)$는 다음과 같이 정의된다.

Gen(k): 보안상수 k을 받아서 $GenRSA$을 실행해 $(N,\ e,\ d, g)$ 를 얻는다. 또한 해쉬 함수 H, h를 각각 다음과 같이 정의한다.

$H : \{0,1\}^* \rightarrow Z_N^*$, $h : \{0,1\}^* \rightarrow \{0,1\}^{\ell}$.

$PK=(N, e,\ g,\ H,\ h)$ 와 $SK=(d)$이다.

Sign(m, d):

① $R \in \{0,1\}^{\ell_1}$을 랜덤하게 선택한다.

② $H(R)^d \pmod N$ 을 계산한다.

③ $g^{h(R\|m)d} \pmod N$ 을 계산한다.

④ $\sigma_1 = (H(R)g^{h(R\|m)})^d \pmod N$를 계산한다.

⑤ $\sigma=(\sigma_1, R)\in Z_N\times\{0,1\}^{l_1}$를 출력한다.

Vrfy(PK, m, σ) : 여기서 $\sigma=[\sigma_1,\ R]$이다. $\sigma_1^e = H(R)\times g^{h(R\|m)} \pmod N$ 이 성립하는지 확인한다. 만약 같다면 1을, 아니면 0을 출력한다.

3.2 온라인/오프라인 서명생성 과정

서명생성 알고리즘은 온라인/오프라인 과정에서 다음과 같은 계산으로 분리될 수 있다.

오프라인 계산과정: 메시지 m이 입력되기 전에 아래의 계산을 수행한다.

① $R \in \{0,1\}^{\ell_1}$을 랜덤하게 선택한다.

② $H(R)^d \pmod N$ 을 계산한다.

③ token=$(R,\ H(R)^d)$값을 안전하게 저장한다.

온라인 서명생성 과정: 메시지 m이 입력되면, 서명키 SK와 저장된 token을 이용하여 아래의 계산을 수행한다.

① $g^{h(R\|m)d} \pmod N$ 을 계산한다.

② $\sigma_1 = H(R)^d \times g^{h(R\|m)d} \pmod N$ 을 구한다.

③ $\sigma=[\sigma_1, R]\in Z_N\times\{0,1\}^{l_1}$를 출력한다.

공개키는 기존 RSA-FDH나 RSA-PSS에서 (N, e)임에 비해 제안기법에서는 원소 $g\in Z_N$가 추가된다. 그러나 작은 사이즈의 원소 g를 (혹은 필요하면 제곱을 해서) 선택해도 위수가 p_1q_1이 되는 경우가 대부분이므로, 실제로는 작은 사이즈의 g값을 기존 공개키 (N, e)에 추가하면 된다. 따라서 제안기법에서 공개키가 크게 증가하는 문제는 완화된다.

오프라인에서는 지수승 한번이 필요하나. 실제 온라인 서명생성 과정에서는 주된 계산이 고정된 밑수 g에서의 지수승, 즉, $g^{h(R\|m)d} \pmod N$, 이 필요하게 된다. 여기서 g의 위수 p_1q_1을 비밀키의 일부로 저장하게 되면, 지수값 $h(R\|m)d$ 을 모듈러스 p_1q_1으로 줄여서 계산하는 것이 가능하다. 일반적으로 고정 밑수 지수승이 일반 지수승보다 약 4배에서 5배정도 빠르다는 것이 알려져 있으므로[10] 제안된 기법의 서명 속도는 온라인 서명생성과정에서 RSA-PSS 또는 RSA-FDH보다 약 4배 빠르게 된다. 또한 서명키에 p, q를 포함하고, CRT를 이용하여 지수승을 하는 것도 가능하다. 이 경우에도 고정된 밑수 g를 기준으로 mod p와 mod q에서 지수승 연산이 가능하므로, 일반적인 RSA 서명을 CRT로 이용하는 것보다 더 빠른 지수승 연산이 가능하다.

3.3 안전성 증명

정리 1. 만일 RSA 문제가 $(t_B,\ \epsilon_B)$-안전하고, 해쉬함수 H, h가 랜덤 오라클로 동작한다면, 위에서 서술된 서명은 선택 메시지 공격에서 $(t_A,\ \epsilon_A,\ q_S)$-안전하다. 여기서

$$t_B \geq t_A + O((q_H+q_S)t_e),$$
$$\epsilon_A \leq \epsilon_B + \frac{(q_H+q_h+q_S)q_S}{2^{\ell_1}} + \frac{1}{2^{\ell}}.$$

해쉬 함수 H, h에게 질의하는 수를 각각 q_H, q_h라 하고 서명 오라클에게 질의하는 수를 q_S 라고 하자. 또한 t_e는 지수 승 계산 비용이다.

증명. $\Pi=(Gen, Sign, Vrfy)$를 위에서 기술된 서명 알고리즘이라 하고, A는 서명 알고리즘을 공

격하는 다항식 시간 공격자라 하자. A를 이용하여 RSA 문제를 푸는 알고리즘 B를 구성한다. B는 RSA 문제의 입력값으로 $(N,\ e,\ y)$를 받는다. 여기서 B의 목적은 $x^e = y \pmod N$ 를 만족하는 $y^{1/e} \pmod N$ 을 계산하는 것이다. 여기서 N은 알려지지 않은 두 강한 소수 p, q에 대해서, $N = pq$이다. 또한 서로 다른 소수 p_1, q_1에 대해서 $p = 2p_1 + 1$, $q = 2q_1 + 1$ 이다.

먼저 B는 Z_N에서 임의의 원소 g_1을 선택한 후, $g_2 = y g_1^e \pmod N$을 구한다. 만일 g_2이 Z_N^*에 속하지 않는다면, 즉 g_2이 N을 구성하는 소수의 배수라면, B는 N을 소인수 분해할 수 있으므로 주어진 문제를 쉽게 해결할 수 있다. 일반성을 잃지 않고, g_2이 Z_N^*에 속한다고 하자. 다음으로 B는 $g = g_2^2 = y^2 g_1^{2e} \pmod N$ 을 계산하여 QR_N에 속하는 원소 g를 구한다. 만일 g의 위수가 $p_1 q_1$이 아니라면, B는 N을 쉽게 소인수 분해할 수 있으므로 주어진 RSA 문제를 해결할 수 있다. 그 이유는 다음과 같다: 강한 소수들의 곱인 N에 대하여 Z_N^*에 속한 각 원소들의 위수는 1, 2, p_1, q_1, $2p_1$, $2q_1$, $p_1 q_1$, $2p_1 q_1$이 가능하다. 여기서 g는 QR_N에 속하므로 $2p_1 q_1$은 위수가 될 수 없다. 만일 g의 위수가 p_1이라면, CRT를 이용하여 $g^{p_1} = 1 \pmod p \ \wedge\ g^1 = 1 \pmod q$ 임을 알 수 있다. 이 경우 $q \mid g-1$이므로, $\gcd(g-1,\ N)$을 구하면 소인수 q를 쉽게 구할 수 있다. 반대로 위수가 q_1이라면 $\gcd(g-1,\ N) = p$를 계산할 수 있다. 비슷하게 g의 위수가 $2p_1$, $2q_1$, 2의 경우도 B의 소인수분해를 가능하게 한다. 따라서 일반성을 잃지 않고, B가 계산한 g의 위수가 (실제 기법과 동일하게) $p_1 q_1$이라고 하자.

Setup 단계:

B는 A에게 $PK = (N,\ e,\ g)$를 준다. 여기서 $g = y^2 g_1^{2e} \pmod N$ 이다.

Query 단계:

A는 해쉬 오라클 H와 h, 서명 오라클 $Sign$에 각각 질의 할 수 있다.

- 해쉬 오라클 H

① H-테이블을 $(R_i,\ w_i,\ r_i,\ H(R_i))$의 빈 테이블로 초기화한다.

② A가 H에게 질의의 입력으로 R_i를 준다.

③ B는 $w_i \in \{0,1\}^{\ell}$, $r_i \in Z_N^*$를 랜덤하게 뽑아 $H(R_i) = y^{-2w_i} r_i^e \pmod N$를 계산해 A에게 준다.

④ H-테이블 $(R_i,\ w_i,\ r_i,\ H(R_i))$에 질의에 이용된 변수들을 저장한다.

- 해쉬 오라클 h

① h-테이블을 $(R_j \| m_j,\ h(R_j \| m_j))$의 빈 테이블로 초기화한다.

② A가 h에게 질의의 입력으로 $R_j \| m_j$를 준다.

③ B는 $w_j \in \{0,1\}^{\ell}$를 랜덤하게 생성해서 $h(R_j \| m_j) = w_j$로 정의하고 A에게 준다.

④ h-테이블 $(R_j \| m_j,\ w_j)$에 질의에 이용된 변수들을 저장한다.

- 서명 오라클 $Sign$

① A가 $Sign$에게 질의의 입력으로 m을 준다.

② B는 $R \in \{0,1\}^{\ell_1}$을 랜덤하게 선택한다.

만약 R이 H-테이블에 존재하면, B는 시뮬레이션을 정지한다. (이 사건을 abt_1이라 하자.) 그 이유는 동일한 R에 대해서 항상 H-테이블에서 같은 w를 참조하도록 B가 동작하므로, 서로 다른 메시지 m_1, m_2에 대해 h 해쉬값이 $h(R \| m_1) = w = h(R \| m_2)$로 같게 된다. 이는 실제 공격 환경과 시뮬레이션을 구분하는 경우이므로, B는 정상적인 수행을 할 수 없다. 이 사건 abt_1이 일어날 확률 $\Pr[abt_1]$은 전체 $\{0,1\}^{\ell_1}$중에 같은 R을 뽑을 확률 이므로 최대 $(q_H + q_S) q_S \ / 2^{\ell_1}$이다. 이는 총 H-테이블 상의 R값이 H질의와 $Sign$질의만큼 선택이 되고, 매 서명마다 R값을 선택할 때 발생할 수 있는 충돌가능성의 확률이다.

③ 그렇지 않다면, R이 H-테이블에 존재하지 않으면, 다음 단계를 진행한다.

B는 $w \in \{0,1\}^{\ell}$, $r \in Z_N^*$를 뽑아 $H(R) = y^{-2w} r^e \pmod N$ 을 계산한다.

만약 $R\|m$이 h-테이블에 존재하면, B는 정지한다. (이 사건을 abt_2이라 하자.) 그 이유는 B가 $h(R\|m) = w$으로 지정하려고 하나, 이미 입력값 $R\|m$이 다른 값으로 지정된 경우이므로 B가 시뮬레이션을 진행할 수 없기 때문이다. 이 사건이 일어날 확률 $\Pr[abt_2]$은 최대 $(q_h + q_S)q_S \ / 2^{\ell_1}$이다. 이 값도 위에서와 비슷하게 총 h-테이블 상의 R값이 h질의와 $Sign$질의만큼 선택이 되고, 매 서명마다 R값을 선택할 때 발생할 수 있는 R값의 충돌가능성의 확률이다.

④ 그렇지 않다면, $h(R\|m) = w$로 정의하고, B는 $\sigma_1 = r g_1^{2w} \pmod N$을 계산한다. 여기서

$$r g_1^{2w} = (y^{-2w} r^e \times (y^2 g_1^{2e})^w)^d = (H(R) g^w)^d \pmod N.$$

A에게 m에 대한 서명으로 $[\sigma_1, R]$을 준다.

Output 단계:

충분히 질의를 진행한 후 A는 위조 서명으로 메시지 m^*에 대한 서명 $\sigma^* = (\sigma_1^*, R^*)$를 B에게 리턴한다. B는 A에게 받은 R^*를 이용해 H-테이블을 참조한다. 여기서 R^*에 해당하는 w^*, r^*를 가져온다. 그리고 h-테이블을 참조하여 $h(R^*\|m^*) = \tilde{w}$인 $\tilde{w}$값을 가져온다. 안전성 모델 정의에 의해 m^*는 이전에 서명 질의된 메시지가 아니고, $H(R^*) g^{h(R^*\|m^*)} = (\sigma_1^*)^e \pmod N$ 이 성립한다.

- 만약 $w^* = \tilde{w}$이면 B는 정지한다. (이 사건을 abt_3이라 하자.) 이 경우 $\gcd(e, 2(\tilde{w} - w^*)) \neq 1$이 되어, B는 정상적인 RSA 문제에 대한 답 y^d를 계산 할 수 없다. 이 사건이 일어날 확률 $\Pr[abt_3]$은 전체 $\{0,1\}^{\ell}$중에 하나의 값을 선택한 후 똑같은 값을 고를 확률이므로 최대 $\Pr[abt_3] = 1/\ 2^{\ell}$ 이다.

- 그렇지 않고 만약 $w^* \neq \tilde{w}$이라면

$$\begin{aligned}\sigma_1^* &= (H(R^*) g^{\tilde{w}})^d \pmod N \\ &= (y^{-2w^*} (r^*)^e \times (y^2 g_1^{2e})^{\tilde{w}})^d \pmod N \\ &= y^{2d(\tilde{w} - w^*)} \times r^* g_1^{2\tilde{w}} \pmod N .\end{aligned}$$

여기서 r^*와 g_1을 알고 있으므로 $\sigma_1^* / r^* g_1^{2\tilde{w}} = y^{2d(\tilde{w} - w^*)} \pmod N$ 을 계산할 수 있다. 여기서 y^d를 구하는 것은 [11]의 아이디어를 이용한다. $y = (y^d)^e \pmod N$ 이고, 지수값 $2(\tilde{w} - w^*)$와 e에 대하여, $\gcd(e, 2(\tilde{w} - w^*)) = 1$임을 알 수 있다. 서로 소인 이유는 $-2^{l+1} < 2(\tilde{w} - w^*) < 2^{l+1}$이고, $e > 2^{\ell+1}$가 되도록 소수 e값을 선택했기 때문이다. 서로 소 이므로 어떤 두 정수 a, b가 존재하여 $ea + b2(\tilde{w} - w^*) = 1$이다. 따라서 B는 a, b를 구한 후, 아래의 계산을 통해 y^d를 구한다.

$$\begin{aligned}&(y^{2d(\tilde{w} - w^*)})^b \times ((y^d)^e)^a \\ &= (y^{ea + 2b(\tilde{w} - w^*)})^d = y^d \pmod N .\end{aligned}$$

B는 계산한 결과를 RSA 문제의 답으로 출력한다.

분석: B의 시간 복잡도는 H질의와 서명 질의에 답하는 과정에서 필요한 지수승 연산과 A의 동작시간이다. 따라서 시간복잡도는 쉽게 보일 수 있다.

B가 RSA 문제를 풀 확률은 abt사건 없이 정상적으로 수행된 A가 서명 Π에 공격을 성공할 확률이다. 따라서

$$\begin{aligned}Adv_{\Pi}^{uf}(A) &= \Pr[\mathrm{Exp}_{\Pi}^{uf}(A) = 1] \\ &= \Pr[\mathrm{Exp}_{\Pi}^{uf}(A) = 1 \wedge abt_1] + \\ &\quad \Pr[\mathrm{Exp}_{\Pi}^{uf}(A) = 1 \wedge \overline{abt_1}].\end{aligned}$$

여기서 $\Pr[\mathrm{Exp}_{\Pi}^{uf}(A) = 1 \wedge abt_1] \leq \Pr[abt_1]$ 이므로,

$$\begin{aligned}Adv_{\Pi}^{uf}(A) \leq \Pr[abt_1] + \\ \Pr[\mathrm{Exp}_{\Pi}^{uf}(A) = 1 \wedge \overline{abt_1}]\end{aligned}$$

가 된다. 비슷한 방법으로 두 사건 abt_2와 abt_3에 대해서도 전개하면, 아래와 같은 부등식을 얻는다.

$$\begin{aligned}&Adv_{\Pi}^{uf}(A) \\ &\leq \Pr[\mathrm{Exp}_{\Pi}^{uf}(A) = 1 \wedge \overline{abt_1} \wedge \overline{abt_2} \wedge \overline{abt_3}] \\ &\quad + \Pr[abt_1] + \Pr[abt_2] + \Pr[abt_3].\end{aligned} \tag{1}$$

$\Pr[\mathrm{Exp}_{\Pi}^{uf}(A)=1 \wedge \overline{abt_1} \wedge \overline{abt_2} \wedge \overline{abt_3}]$은 abt 사건들이 발생하지 않았을 경우 공격자 A가 서명위조에 성공할 확률이다. 이 경우 B는 RSA 문제를 풀 수 있으므로

$$\Pr[\mathrm{Exp}_{\Pi}^{uf}(A)=1 \wedge \overline{abt_1} \wedge \overline{abt_2} \wedge \overline{abt_3}]=\epsilon_B$$

이다. 또한 $Adv_{\Pi}^{uf}(A)=\epsilon_A$이고, 위에서 계산했던 abt의 확률들을 위 부등식 (1)에 적용하면 다음과 같다.

$$\epsilon_A \le \epsilon_B + \frac{(q_H+q_h+2q_S)q_S}{2^{l_1}} + \frac{1}{2^l}.$$

□

IV. On-line/off-line RSA 서명 기법 2

4.1 서명 기법 설계

앞 절의 $GenRSA$ 알고리즘과 동일하게 보안상수 입력 k에 대해서 $(N,\ e,\ d, g)$ 를 생성한다. 여기서 N은 강한 소수(strong prime) p, q의 곱이고, $p = 2p_1+1$, $q = 2q_1+1$ 이다. 그리고 e는 $\ell+2$비트 보다 큰 어떤 소수이다. 즉, $e > 2^{\ell+2}$이다. g는 Z_N^* 의 부분군을 구성하는 위수가 p_1q_1인 생성원이다.

전자서명 알고리즘 $\Pi=(Gen,\ Sign, Vrfy)$는 다음과 같이 정의된다.

Gen(k): 보안상수 k을 받아서 $GenRSA$을 실행해 $(N,\ e,\ d, g)$ 를 얻는다. 또한 해쉬 함수 H, h를 각각 다음과 같이 정의한다.

$H : \{0,1\}^* \rightarrow Z_N^*$, $h : \{0,1\}^* \rightarrow \{0,1\}^{\ell}$.

$PK=(N, e,\ g,\ H,\ h)$ 와 $SK=(d)$이다.

Sign(m, d):

① $r \in \{0,1\}^{\ell_2}$을 랜덤하게 선택한다.

② $A = g^{er} \pmod N$을 계산한다.

③ $\sigma_1 = H(A)^d \pmod N$ 을 계산한다.

④ $s = r + h(\sigma_1 \| m)d \in \{0,1\}^{l_2}$을 계산한다.

⑤ $\sigma = [\sigma_1,\ s] \in Z_N \times \{0,1\}^{l_2}$를 출력한다.

Vrfy(PK, m, σ) : 여기서 $\sigma = [\sigma_1,\ s]$이다.

① $A = g^{es - h(\sigma_1 \| m)} \pmod N$ 을 계산한다.

② $(\sigma_1)^e = H(A) \pmod N$ 을 확인한다. 만약 같다면 1을, 아니면 0을 출력한다.

정확성: 메시지 m에 대한 서명 $\sigma = [\sigma_1,\ s]$이 주어지면, $g^{ed} = g \pmod N$ 사실을 이용하여 다음과 같이 계산된다.

$$\begin{aligned} A &= g^{es - h(\sigma_1 \| m)} \\ &= g^{e(r + h(\sigma_1 \| m)d) - h(\sigma_1 \| m)} = g^{er} \pmod N. \end{aligned}$$

이하의 과정은 쉽게 보일 수 있으므로 생략한다.

4.2 온라인/오프라인 서명생성 과정

서명생성 알고리즘은 온라인/오프라인 과정에서 다음과 같은 계산으로 분리될 수 있다.

오프라인 계산과정: 메시지 m이 입력되기 전에 아래의 계산을 수행한다.

① $r \in \{0,1\}^{\ell_2}$을 랜덤하게 선택한다.

② $A = g^{er} \pmod N$을 계산한다.

③ $\sigma_1 = H(A)^d \pmod N$ 을 계산한다.

④ token=$(r,\ \sigma_1)$값을 안전하게 저장한다.

온라인 서명생성 과정: 메시지 m이 입력되면, 서명키 SK와 저장된 token을 이용하여 아래의 계산을 수행한다.

① $s = r + h(\sigma_1 \| m)d \in \{0,1\}^{l_2}$을 계산한다.

② $\sigma = [\sigma_1,\ s]$를 서명값으로 출력한다.

오프라인에서는 한 번의 고정된 밑수에서의 지수승과 한 번의 지수승이 필요하지만, 온라인 서명과정에서는 해쉬연산, 곱셈 및 덧셈 연산만 수행하면 된다. 여기서 g의 위수 p_1q_1을 비밀키의 일부로 저장하게 되면, 지수값 er을 모듈러스 p_1q_1으로 줄여서 계산하는 것이 가능하다. 또한 앞 절의 기법과 마찬가지로 서명키에 p, q를 포함하고, CRT를 이용하여 지수승을 하는 것도 가능하다.

4.3 안전성 증명

정리 1. 만일 RSA 문제가 $(t_B,\ \epsilon_B)$-안전하고, 해쉬함수 H, h가 랜덤 오라클로 동작한다면, 위에서 서술된 서명은 선택 메시지 공격에서 $(t_A,\ \epsilon_A,\ q_S)$-안전하다. 여기서

$$t_B \geq t_A + O((q_H + q_S)t_e),$$
$$\epsilon_A \leq \epsilon_B + \frac{(q_H + q_h + q_S)q_S}{p_1 q_1} + \frac{1}{2^\ell}.$$

해쉬 함수 H, h에게 질의하는 수를 각각 q_H, q_h라 하고 서명 오라클에게 질의하는 수를 q_S 라고 하자. 또한 t_e는 지수 승 계산 비용이다. $p_1 q_1$은 QR_N의 그룹 위수이다.

증명. $\Pi = (Gen, Sign, Vrfy)$를 위에서 기술된 서명 알고리즘이라 하고, A는 서명 알고리즘을 공격하는 다항식 시간 공격자라 하자. A를 이용하여 RSA 문제를 푸는 알고리즘 B를 구성한다. B는 RSA 문제의 입력값으로 $(N,\ e,\ y)$를 받는다. 여기서 B의 목적은 $x^e = y \pmod N$ 를 만족하는 $y^{1/e} \pmod N$ 을 계산하는 것이다. 여기서 N은 알려지지 않은 두 강한 소수 p, q에 대해서, $N = pq$이다. 또한 서로 다른 소수 p_1, q_1에 대해서 $p = 2p_1 + 1$, $q = 2q_1 + 1$ 이다.

정리 1의 증명과 비슷하게 B는 Z_N에서 임의의 원소 g_1을 선택한 후, $g = g_1^2 \pmod N$을 계산하여 QR_N에 속하는 원소 g를 구한다. 일반성을 잃지 않고, B가 계산한 g의 위수가 (실제 기법과 동일하게) $p_1 q_1$이라고 하자.

Setup 단계:
B는 A에게 $PK = (N,\ e,\ g)$를 준다. 여기서 $g = g_1^2 \pmod N$ 이다.

Query 단계:
A는 해쉬 오라클 H와 h, 서명 오라클 $Sign$에 각각 질의할 수 있다.

- 해쉬 오라클 H
① H-테이블을 $(A_i,\ t_i,\ H(A_i))$로 초기화한다.
② 공격자 A가 H에게 $A_i \in Z_N$를 준다.
③ B는 $t_i \in Z_N^*$를 랜덤하게 뽑아 $H(A_i) = y t_i^e \pmod N$를 계산해 A에게 준다.
④ H-테이블 $(A_i,\ t_i,\ H(A_i))$에 질의에 이용된 변수들을 저장한다.

- 해쉬 오라클 h
① h-테이블을 $(R_j \| m_j,\ h(R_j \| m_j))$의 빈 테이블로 초기화한다.
② A가 h에게 질의의 입력으로 $R_j \| m_j$를 준다.
③ B는 $w_j \in \{0,1\}^\ell$를 랜덤하게 생성해서 $h(R_j \| m_j) = w_j$로 정의하고 w_j를 A에게 준다.
④ h-테이블 $(R_j \| m_j,\ w_j)$에 질의에 이용된 변수들을 저장한다.

- 서명 오라클 $Sign$
① A가 $Sign$에게 질의의 입력으로 m을 준다.
② B는 $r \in \{0,1\}^{\ell_2}$와 $w \in \{0,1\}^l$을 랜덤하게 선택한다. 다음 $\widetilde{A} = g^{er - w} \pmod N$을 계산한다. 이 경우 B가 모르는 d에 대해서 $g^{ed} = g \pmod N$이므로, $\widetilde{r} = r - wd$이고 $\widetilde{A} = g^{e\widetilde{r}}$임을 상기하자.

만약 $\widetilde{A}$값이 H-테이블 상에 존재하면, B는 시뮬레이션을 정지한다. (이 사건을 abt_1이라 하자.) 그 이유는 동일한 $\widetilde{A}$에 대해서 항상 H-테이블에서 같은 (y가 포함된) $H(\widetilde{A})$를 참조하도록 B가 동작하므로, 이 경우 B가 서명질의에 답할 수 없기 때문이다. 이 사건 abt_1이 일어날 확률 $\Pr[abt_1]$은 g의 위수 $p_1 q_1$중에 같은 위수를 뽑을 확률이므로, 최대 $(q_H + q_S)q_S \ / p_1 q_1$이다. 이는 총 H-테이블 상의 A_i값이 H질의와 $Sign$질의만큼 선택이 되고, 매 서명마다 A_i값을 선택할 때 발생할 수 있는 충돌 가능성의 확률이다.

③ 그렇지 않다면, $\widetilde{A}$이 H-테이블에 존재하지 않으면, B는 $t \in Z_N^*$를 랜덤하게 뽑고, $H(\widetilde{A}) = t^e \pmod N$를 계산한다. H-테이블에 $(\widetilde{A},\ t,\ H(\widetilde{A}))$ 및 관련된 (r, w)를 저장한다.

④ B는 $\sigma_1 = t$로 설정한다. 이 값이 정상적인 것은 아래 식 $\sigma_1 = H(\widetilde{A})^d = (t^e)^d = t$로 확인된다.

⑤ B는 $h(\sigma_1 \| m) = w$로 정의한다. 만일 $\sigma_1 \| m$이 h-테이블에 존재하면, B는 정지한다. (이 사건을 abt_2이라 하자.) 그 이유는 B가 $h(\sigma_1 \| m) = w$으로 지정하려고 하나, 이미 입력값 $\sigma_1 \| m$이 다른 값으로 지정된 경우이므로 B가 시뮬레이션을 진행할 수 없기 때문이다. 이 사건이 일어날 확률 $\Pr[abt_2]$은 최대 $(q_h + q_S)q_S \,/\, p_1 q_1$이다. 이 값도 위에서와 비슷하게 총 h-테이블 상의 σ_1값이 h 질의와 $Sign$질의만큼 선택이 되고, 매 서명마다 σ_1 값을 선택할 때 발생할 수 있는 σ_1값의 충돌가능성의 확률이다.

⑥ 그렇지 않다면, $h(\sigma_1 \| m) = w$로 정의하고, h-테이블에 $(\sigma_1 \| m, w)$를 저장한다. 다음 B는 $s = r$로 설정한다. 여기서 $\tilde{r} = r - wd$로 설정되었으므로, $s = \tilde{r} + h(\sigma_1 \| m)d = r - wd + wd = r$이 되므로 정당성을 확인할 수 있다. 최종적으로 A에게 m에 대한 서명으로 $[\sigma_1, s]$을 준다.

Output 단계:

충분히 질의를 진행한 후 A는 위조 서명으로 메시지 m^*에 대한 서명 $\sigma^* = (\sigma_1^*, t^*)$를 B에게 리턴한다. B는 $A^* = g^{e s^* - h(\sigma_1^* \| m^*)} \pmod N$를 계산한다. B는 H-테이블을 참조하여 $(A^*, t^*, H(A^*))$을 구한다.

① 만일 A^*값이 공격자에 의해 질의된 입력값이면, $H(A^*) = yt^e$이 되고, 정상적인 검증을 통과하므로 $\sigma_1^* = H(A^*)^d \pmod N$이다. 따라서 $\sigma_1^* = y^d\, t$이므로 B는 σ_1^*/t로 y^d를 구할 수 있다.

② 만일 A^*값이 B에 의해 (서명응답 도중에) 생성된 입력값이면, $H(A^*) = t^e$이 된다. 이 경우 A^* 값을 구성하는 지수값을 B는 (간접적으로) 알고 있으므로, 이 지수값을 $e\tilde{r} = e(r^* - w^* d)$라 하고, (r^*, w^*)을 안다고 하자. 정상적인 서명 검증식에 의해 $A^* = g^{e s^* - h(\sigma_1^* \| m^*)} \pmod N$이므로 아래의 식이 성립한다.

$$e(r^* - w^* d) = e s^* - h(\sigma_1^* \| m^*) \pmod{p_1 q_1} \tag{2}$$

여기서 $\phi(N) = 4p_1 q_1$이므로 $ed = 1 \pmod{\phi(N)}$식을 만족하는 e의 역원 d는 $ed = 1 \pmod{p_1 q_1}$에서도 역원이다. 따라서 위 식 (2)에 d를 곱해주면

$$\begin{aligned} &r^* - w^* d = s^* - h(\sigma_1^* \| m^*)d \pmod{p_1 q_1}, \\ &(h(\sigma_1^* \| m^*) - w^*)d = s^* - r^* \pmod{p_1 q_1}, \\ &s^* - r^* = (h(\sigma_1^* \| m^*) - w^*)d + k p_1 q_1 \end{aligned} \tag{3}$$

이 된다. 여기서 k는 임의의 정수이다. 위 식(3)의 양변에 4를 곱해서 y의 지수로 올리면

$$y^{4(s^* - r^*)} = y^{4(h(\sigma_1^* \| m^*) - w^*)d}\, y^{k 4 p_1 q_1} \pmod N$$

이고, $4p_1 q_1 = \phi(N)$이므로 $y^{4 p_1 q_1 k} = 1 \pmod N$이 되어 $y^{4(s^* - r^*)} = y^{4(h(\sigma_1^* \| m^*) - w^*)d} \pmod N$의 값을 구할 수 있다.

③ 만일 $h(\sigma_1^* \| m^*) = w^*$이면 B는 정지한다. (이 사건을 abt_3이라 하자.) 이 경우 B는 정상적인 RSA 문제에 대한 답 y^d를 계산 할 수 없다. 이 사건이 일어날 확률 $\Pr[abt_3]$은 전체 $\{0,1\}^\ell$중에 하나의 값을 선택한 후 똑같은 값을 고를 확률이므로 최대 $\Pr[abt_3] = 1/\,2^\ell$ 이다.

④ 그렇지 않고 만약 $h(\sigma_1^* \| m^*) \neq w^*$라면, 정리 1처럼, B는 $\gcd(e,\ 4(h(\sigma_1^* \| m^*) - w^*)) = 1$이므로 $ea + b4(h(\sigma_1^* \| m^*) - w^*) = 1$를 만족하는 두 정수 a, b를 구하고, a, b를 이용하여 아래의 계산으로 y^d를 구한다.

$$\begin{aligned} &((y^d)^e)^a\, (y^{4(h(\sigma_1^* \| m^*) - w^*)d})^b \\ &= (y^d)^{ea + 4(h(\sigma_1^* \| m^*) - w^*)b} = y^d \pmod N. \end{aligned}$$

B는 계산한 결과를 RSA 문제의 답으로 출력한다.

Table 1. Comparison between previous RSA-based signatures and our proposed ones

	Signing cost		Verification cost	Signature size	Security loss	Security assumption
	Off-line	On-line				
RSA-FDH[6]	-	$1E$	$1E^e$	$1Z_N$	$O(q_h)$	RSA
RSA-PSS[7]	-	$1E$	$1E^e$	$1Z_N$	$O(1)$	RSA
Ours 1	$1E$	$1E_f$	$1E^e+1E$	$1Z_N+\{0,1\}^{l_1}$	$O(1)$	RSA
Ours 2	$1E+1E_f$	-	$1E^e+1E$	$1Z_N+\{0,1\}^{l_2}$	$O(1)$	RSA

E: general exponentiation; E^e: exponentiation with RSA exponent e; q_h: the number of hash queries; E_f: fixed-base exponentiation; l_1: bit size of randomness in our 1st construction; $l_2(\approx \log N+l_h)$: bit size of randomness in our 2nd construction for hash output size l_h.

분석: B의 시간 복잡도는 H질의와 서명 질의에 답하는 과정에서 필요한 지수승 연산과 A의 동작시간이다. 따라서 시간복잡도는 쉽게 보일 수 있다.

B가 RSA 문제를 풀 확률은 abt사건 없이 정상적으로 수행된 A가 서명 Π에 공격을 성공할 확률이다. 따라서 정리 1에서와 같이

$$\begin{aligned} &Adv_{\Pi}^{uf}(A) \\ &\le \Pr[\mathrm{Exp}_{\Pi}^{uf}(A)=1 \wedge \overline{abt_1} \wedge \overline{abt_2} \wedge \overline{abt_3}] \\ &\quad + \Pr[abt_1]+\Pr[abt_2]+\Pr[abt_3]. \end{aligned} \tag{4}$$

$\Pr[\mathrm{Exp}_{\Pi}^{uf}(A)=1 \wedge \overline{abt_1} \wedge \overline{abt_2} \wedge \overline{abt_3}]$은 abt 사건들이 발생하지 않았을 경우 공격자 A가 서명위조에 성공할 확률이다. 이 경우 B는 RSA 문제를 풀 수 있으므로

$$\Pr[\mathrm{Exp}_{\Pi}^{uf}(A)=1 \wedge \overline{abt_1} \wedge \overline{abt_2} \wedge \overline{abt_3}]=\epsilon_B$$

이다. 또한 $Adv_{\Pi}^{uf}(A)=\epsilon_A$이고, 위에서 계산했던 abt의 확률들을 위 부등식 (4)에 적용하면 다음과 같다.

$$\epsilon_A \le \epsilon_B+\frac{(q_H+q_h+2q_S)q_S}{p_1q_1}+\frac{1}{2^l}.$$

□

V. RSA 서명 기법들과의 비교

Table 1에서는 기존 RSA 기반 서명 기법들, RSA-FDH[6], RSA-PSS[7]과 본 논문에서 제안된 기법들 사이의 안전성 및 효율성 비교를 한다.

먼저 안전성 측면에서 RSA-FDH는 증명과정에서 RSA 문제에 효율적으로 환원관계가 성립하지 않고, $O(q_h)$정도의 안전성 손실(Security loss)이 발생한다. 여기서 q_h는 공격자가 해쉬함수에 질의하는 오라클의 개수이다. 이에 비해 RSA-PSS 및 제안기법 1과 2는 모두 RSA 문제와 동등한 환원관계로 증명이 된다. 따라서 대략 $q_h=2^{40}$라 하고 서명기법이 128비트의 안전성을 추구한다면, RSA-FDH는 나머지 기법들에 비해 약 40비트의 보안상수가 증가된, 즉, 168(=128+40)비트 보안상수에서 $GenRSA$ 알고리즘을 동작시켜야 한다.

효율성 측면에서 제안기법 1과 2는 RSA-FDH와 RSA-PSS에 비해 서명길이와 검증비용 측면에서 다소 비효율적이다. 그러나 RSA-FDH와 RSA-PSS는 온라인/오프라인 서명 구조를 지원하지 않아서, 매 서명시마다 온라인 서명만 할 수 있다. 이에 비해 제안기법 1은 오프라인에서 지수승 한 번을 수행하고, 온라인에서는 고정된 밑수에서의 지수승 한번만으로 서명이 가능하다. 일반적으로 고정된 밑수에서의 지수승은 일반적인 지수승보다 약 4배 빠르므로 제안기법 1은 온라인/오프라인 서명을 하는 경우 기존 RSA-FDH나 RSA-PSS에 비해 약 4배 빠른 온라인 서명이 가능하다. 제안기법 2는 아예 온라인 서명생성에 필요한 연산을 거의 모두 오프라인으로 사전에 수행하는 것이 가능하다. 이 경우 한 번의 지수승과 고정된 밑수 지수승을 한 번씩 오프라인에서 사전 계산해 놓으면, 온라인상의 특정 메시지에 대한 서명은 거의 공짜로 할 수 있다.

VI. 결 론

본 논문에서 제안하는 기법들은 다음의 두 가지 측면에서 의미를 가진다. 첫째, 안전성 증명 측면에서 RSA 문제에 안전성 손실 없이 환원됨을 보인 것이다. 이론적인 관점에서는 기존의 RSA-FDH 또는 RSA-PSS 기법이 아닌 새로운 접근방법으로 해결했다는 것이 의미가 있고, 실용적인 측면에서는 서명기법 구현 시 보안상수를 더 크게 하여 안전성 손실을 보정할 필요가 없다는 것이다. 즉, 서명기법이 128비트의 안전성을 추구한다면 128비트 정도로 RSA 문제가 안전한 키를 설정하게 된다. 둘째, 안전성 손실 없이도 온라인/오프라인 서명이 가능하다는 것이다. 이는 오프라인 단계에서 부담되는 연산을 미리 수행하고 다수의 서명응답 요청이 많아질 수 있는 온라인 서버 환경에 적합하다.

References

[1] S. Even, O. Goldreich, and S. Micali, "On-line/off-line digital signatures," CRYPTO'89, LNCS vol. 435, pp. 263-275, Aug. 1989

[2] H. Krawczyk and H. Wee, "The OPTLS Protocol and TLS 1.3," IEEE EuroS&P'16, pp. 81-96, Mar. 2016

[3] H. Krawczyk, K. G. Paterson, and H. Wee, "On the security of the TLS Protocol: a systematic analysis," CRYPTO'13, LNCS vol. 8042, pp. 429-448, Aug. 2013

[4] M. Bellare and T. Ristov, "A characterization of chameleon hash functions and new, efficient designs," Journal of Cryptology, vol. 27, pp. 799-823, Oct. 2014

[5] A. Shamir and Y. Tauman, "Improved on-line/off-line signature schemes," CRYPTO'01, LNCS vol. 2139, pp. 355-367. Aug. 2001

[6] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," ACM-CCS'93, pp. 62-73. Nov. 1993

[7] M. Bellare and P. Rogaway, "The exact security of digital signature-how to sign with RSA and Rabin," Eurocrypt'96, LNCS vol. 1070, pp. 399-416. May. 1996

[8] M. Joye, "An efficient on-line/off-line signature scheme without random oracles," CANS'08, LNCS vol. 5339, pp. 98-107. Dec. 2008

[9] S. Goldwasser, S. Micali, and R. Rivest, "A digital signature scheme secure against adaptive chosen message attacks," SIAM Journal of Computing, vol. 17(2), pp. 281-308, Apr. 1988

[10] X. Boyen, "A tapestry of identity based encryption: practical frameworks compared", International Journal of Applied Cryptography, vol. 1(1), pp. 3-21, Feb. 2008

[11] M. Bellare, C. Namprempre, and G. Neven, "Security proofs for identity-based identification and signature schemes," Journal of Cryptology, vol. 22(1), pp. 1-61, Jan. 2009

〈저자소개〉

최 경 용 (Choi Kyung Yong) 학생회원
2016년 2월: 상명대학교 컴퓨터과학과 졸업
2016년 9월~현재: 상명대학교 일반대학원 컴퓨터과학과 석사과정
〈관심분야〉 인증 및 키 교환, 운영체제 보안

박 종 환 (Jong Hwan Park) 정회원
1999년 2월: 고려대학교 수학과 졸업
2004년 2월: 고려대학교 정보보호대학원 석사
2008년 8월: 고려대학교 정보보호대학원 박사
2013년 9월~현재: 상명대학교 컴퓨터과학과 조교수
〈관심분야〉 함수 암호, 영지식 증명, 암호 프로토콜