공개키 구조 환경에서 해쉬 알고리즘의 안전성 정도에 상관없는 안전한 디지털 서명 방식

송성근*, 윤희용*, 이호재, 이형수
*성균관대학교 정보통신공학부
한국 정보 보호 진흥원 정보 기반 보호 기술팀
전자 부품 연구원 정보시스템 연구센터
songsk@mail.skku.ac.kr, youn@ece.skku.ac.kr, leehj@kisa.or.kr, hslee@keti.re.kr

Secure Digital Signature Scheme regardless of the level of Security of the Hash Algorithm in Public Key Infrastructure Environment

Sung Keun Song*, Hee Yong Youn*, Ho-Jae Lee, Hyung Soo Lee
*School of Information and Communication, SungKyunKwan University
Critical Information Infrastructure Protection Division, Korea Information Security Agency
IT System Research Center, Korea Electronics Technology Institute

요 약

많은 디지털 서명 알고리즘들이 제안되었지만, 내재된 위험 요소들은 여전히 해결되지 않고 있다. 그 중 해쉬 함수의 충돌(Collision) 문제가 있는데, 중요한 것은 이 문제로 인해 메시지가 일단 위조 되면 어느 누구도 그 메시지의 위법성을 증명할 수 없다는 것이다. 따라서, 본 논문에서는 이 문제를 해결하는 방법에 대해 연구하고 새로운 디지털 서명 방식을 제안한다. 제안된 디지털 서명 방식은 해쉬(Hash) 알고리즘의 안전성 정도에 상관없고 오직 공개키 암호의 안전성에 연관되기 때문에 계산 속도가 빠른 해쉬 알고리즘을 사용할 수 있다. 또한, 이 방식은 기존 디지털 서명 알고리즘과 함께 사용될 수 있어서 E-commerce를 더욱 활성화 시킬 것으로 예상된다.

1. 서 론

디지털 서명은 잘 알려진 되로 수학적 문제들을 기반으로 하는 공개키 암호의 원리를 기초로 하고 있다. 대표적 디지털 서명 알고리즘으로 RSA 디지털 서명 알고리즘, ElGamal 디지털 서명 알고리즘, DSS(Digital Signature Standard) 등이 있다[1-3]. 이러한 알고리즘들이 있지만 디지털 서명에 내제된 위험 요소는 여전히 해결되지 못하고 있다. 내제된 위험 요소는 두 가지가 있는데, 그 첫 번째가 디지털 서명 알고리즘의 기초가 되는 공개키 암호의 안전성 문제이고, 두 번째가 서명할 때 사용되는 해쉬 알고리즘의 충돌 문제이다. 첫 번째는 디지털 서명 알고리즘에 있어서 원천적이기 때문에 본 논문에서는 이 문제를 다루지 않는다. 많은 기존 디지털 서명 알고리즘들은 두 번째 문제를 해결하기 위해 충돌이 되는 메시지를 찾기 어렵게 해쉬 알고리즘을 발전시켰을 뿐 본질적인 문제는 해결하지 못하고 있다. 해쉬 알고리즘의 많은 발전으로 공격자가 해쉬 알고리즘의 충돌 문제를 이용하여 디지털 서명을 위조하는 것은 어려운 문제가 됐지만 위조의 가능성은 여전히 존재한다. 왜냐하면, 해쉬 알고리즘에 있어서 충돌 문제는 본질적 특성이기 때문이다. 또한 중요한 문제점은 충돌 문제로 인해 메시지가 일단 위조 되면 어느 누구도 그 메시지의 위법성을 증명할 수 없다는 것이다. 따라서, 본 논문에서는 이 문제를 해결하는 방법에 대해서 연구하고, 디지털 서명의 안전성이 해쉬 알고리즘의 안전성 정도에는 상관없고 오직 공개키 암호의 안전성에 연관되는 새로운 디지털 서명 방식을 제안한다. 이 새로운 디지털 서명 방식은 해쉬 알고리즘의 안전성 정도에 상관없기 때문에 보안성은 떨어지지만 계산 속도가 빠른 해쉬 알고리즘을 사용할 수 있을 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 해쉬 알고리즘의 충돌 문제를 이용한 위조의 예들을 알아본다. 3장에서는 새로운 디지털 서명 방식을 제안하고, 4장에서는 새로운 디지털 서명 방식의 안전성에 대해서 논의한다. 마지막으로 5장에서 본 논문의 결론을 맺는다.

2. 해쉬 알고리즘을 이용한 위조

많은 기존 해쉬 알고리즘들은 Merkle-Damgard 모델을 따르고 있다[4-5]. 이 모델의 특징은 다른 모델과 마찬가지로 메시지의 크기가 증가할수록 충돌 수가 증가한다는 것이다. 이런 특성 때문에 악의적인 서명자나 검증자 또는 제 3 자는 충돌 문제를 이용하여 서명자가 서명한 메시지에 새로운 문장, 단어, 공백을 삽입하므로서 메시지를 위조할 수 있다. 여기서 위조된 메시지의 서명은 유효할 뿐만 아니라 타임스탬프(TS: Time Stamp) 토큰(Token) 또한 유효하기 때문에 서명자는 자신이 위조 메시지에 서명하지 않았다는 것을 기술적으로 증명할 수 없다. 왜냐하면 TS 또한 서명자가 TSA(Time Stamp Authority)에게 메시지의 해쉬값을 보내면서 요구하기

때문이다[6]. 이러한 문제는 다 해쉬 알고리즘의 충돌 문제에서 기인된다. 이러한 문제 때문에 전자상거래에서 심각한 문제가 발생할 수 있다.

해쉬 알고리즘의 충돌 문제를 이용한 공격을 3 가지로 분류할 수 있다. 그 첫 번째가 공격자가 해쉬 알고리즘의 설계 구조의 문제점을 찾는 방법이다. 두 번째는 공격자가 공격 대상자의 유효한 디지털 서명과 TS token를 공개키 인증서의 유효기간 동안 데이터 베이스화하여 메시지를 위조할 때 데이터 베이스에서 위조된 메시지와 같은 해쉬값을 갖는 서명과 TS token을 찾아 위조 메시지에 첨부하므로서 위조하는 방법이다. 세 번째는 공격자가 공격 대상 메시지의 해쉬값과 같을 때까지 위조 문서를 수정하는 것을 반복하는 방법이다. 여기서 세 번째 공격 유형은 다시 두 가지 관점으로 분류할 수 있다. 그 첫 번째가 검증자의 관점에서의 위조 방법이다. 악의적인 검증자는 서명자에게 받은 메시지를 충돌 문제를 이용하여 위조하려 할 것이다. 검증자는 원본 메시지의 해쉬값과 같도록 위조 문서는 수정하면서 위조를 시행 할 것이다. 이후, 원본 메시지의 서명과 TS token를 위조 문서에 첨부할 것이다. 위에서 설명한 것처럼 위조 문서의 서명과 TS token은 유효하다. 왜냐하면, 원본 메시지의 해쉬값과 위조 메시지의 해쉬값이 같기 때문이다. 따라서, 만일 비즈니스상 메시지가 중요한 내용이라면, 이는 서명자에게 심각한 문제를 초래할 것이다.

두 번째는 서명자 관점에서의 위조 방법이다. E-commerce에서 계약서처럼 두 명의 사용자가 서명을 해야 하는 경우가 있을 것이다. 만일 서명자(사용자 A)가 악의적으로 메시지를 위조하려 한다면, 다음과 같이 할 것이다. 먼저 서명자는 위조 메시지가 원본 메시지의 해쉬값과 같도록 만든 다음 원본 메시지에 서명 후 검증자(사용자 B)에게 원본 메시지와 서명, TS token A을 보낸다. 검증자는 원본 메시지와 서명, TS token A의 이상 유무를 확인한 다음 이상이 없으면 서명 후 다시 서명자에게 보낸다. 서명자는 원본 메시지와 서명, TS token B을 받으면 자신의 서명과 검증자의 서명 그리고 TS token들을 위조 메시지에 첨부하므로서 위조를 완료 할 것이다. 여기서 또한 서명 및 TS token들이 모두 유효하다. 따라서 훗날에 서명자가 위조 메시지로 검증자와 최초로 계약을 맺었다고 주장하면, 검증자는 자신이 위조 메시지에 서명하지 않았다는 것을 기술적으로 증명할 수가 없다. 마찬가지로 비즈니스상 중요한 계약서라면, 경계적으로 큰 손실이 검증자에게 발생할 것이다.

3. 새로운 디지털 서명

새로운 디지털 서명 방식은 다른 두 개의 공개키 암호가 결합된 두 개의 공개키 암호를 사용한다. 이 암호는 두 개의 공개키, 하나의 비밀키, 그리고 두 개의 알고리즘으로 구성된다. 이는 컴퓨터상에서 키들이 비트로 표현되기 때문에 가능하다. 대표적인 예로 RSA와 ElGamal의 결합을 들 수 있다.

기존 디지털 서명 방식은 PKI 구조를 기초로 하고 있다. 새로운 디지털 서명 방식도 마찬가지이다. 일반적으로

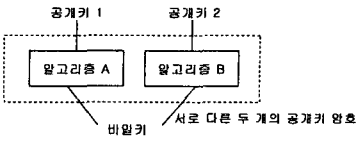

그림 1 두 개의 공개키 암호

PKI 구조에서 인증기관은 End-Entity에게 인증서 발행을 책임지며, 인증서 발행 후에는 인증서의 라이프 싸이클(Life Cycle)에 따라 관리를 책임진다. 새로운 방식에서는 인증기관에게 서명자의 서명을 확인하고 인증을 위해 메시지에 서명하는 기능을 추가한다. 검증자는 서명자의 서명의 유효성을 인증기관의 서명을 통해 확인한다. 새로운 디지털 서명 방식은 서명 과정, 인증 과정, 검증 과정 등 3 가지 과정으로 이루어진다.

P_A : 알고리즘 A의 공개키; 모든 객체에게 공개됨

P_B : 알고리즘 B의 준공개키(Semipublic Key); 인증 기관들에게만 공개됨

P_{AB}^{-1} : 두 개의 공개키 암호의 비밀키

{ } AKey : 알고리즘 A에 의한 암호화 또는 복호화

H: 임의 길이의 입력을 일정 크기의 값으로 늘리는 알고리즘

h: 임의 길이의 입력을 일정 크기의 값으로 압축시키는 알고리즘

서명자에 의한 서명 과정은 다음과 같다. 먼저 서명자는 랜덤키(Random Key) K를 생성하여 인증기관 및 제 3 자가 메시지 내용을 볼 수 없도록 메시지를 대칭 암호 알고리즘으로 암호화한다. 만일 메시지 내용이 중요하지 않으면 이 과정은 생략해도 된다. 서명자는 랜덤넘버(RN: Random Number)의 해쉬값 H(RN)을 계산한다. H(RN)의 값의 크기는 대칭 암호의 블록 크기가 적당할 것이다. 서명자는 암호화된 메시지에 H값을 추가하여 해쉬값 h({M}K, H(RN))의 값을 구한다. 여기서 h를 계산할 때, H의 값은 서명자가 선택한 암호화된 메시지의 특정 블록에 위치한다. 만일 메시지가 암호화되지 않았다면, 메시지를 블록 단위로 나누어 위치시킨다. 서명자는 h값, H값을 메시지에 위치시킨 블록 위치(BP: Block Position), RN를 비밀키를 이용하여 알고리즘 B로 암호화해서 디지털 서명을 만든다. 그리고 해쉬값 h({M}K)를 계산하여 TSA에게 보내므로 TS token을 요구한다. 그 후, 디지털 서명, 암호화된 메시지, 사용자 ID, 검증자(User B)의 공개키로 암호화된 랜덤키, TS token을 인증기관에 전송한다. 그림 2은 서명 과정을 보이고 있다.

ID A, {M}K, User $A_{Signature}$, {K} $P_{A\_User\ B}$,

TS token → CA

-User $A_{Signature}$: {h({M}K, H(RN))||BP||RN} $BP_{AB\_User\ A}^{-1}$

인증기관에 의한 인증 과정은 다음과 같다. 인증기관은 사용자의 ID를 이용하여 데이터 베이스로부터 사용자의 준공개키 P_B 을 찾는다. 그리고 디지털 서명을 공개키로 복호화하여 BP와 RN을 얻어 h({M}K, H(RN))를 계산하고, 디지털 서명의 일부분인 h 해쉬값과 비교하여 일치

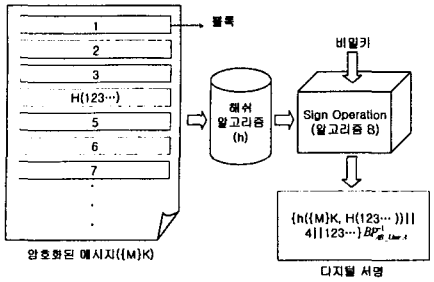

그림 2 서명 과정

여부를 확인한다. 이상이 없다면 인증을 위해 서명 번호(SN: Sign Number)를 추가하여 {M}K와 해쉬값 h(SN, {M}K)을 계산한 다음 User $A_{Signature}$와 h를 Exclusive-OR하여 그 값을 비밀키로 암호화하여 디지털 서명 {User $A_{Signature} \oplus h$} P_{CA}^{-1}을 만든다. 그리고 그 디지털 서명과 인증기관의 인증서, 서명자에게 받은 메시지들을 검증자에게 보낸다. 여기서 인증기관은 두 개의 공개키 암호일 필요는 없다.

SN, {M}K, {K} $P_{A\_User\ B}$, TS token, User $A_{Signature}$, $CA_{Signature}$, CA's Certificate → User B

-$CA_{Signature}$:{User $A_{Signature} \oplus h$(SN, {M}K)} P_{CA}^{-1}

검증자에 의한 검증 과정은 다음과 같다. 검증자는 먼저 h(SN, {M}K)을 계산하고 인증기관의 공개키를 이용하여 디지털 서명을 복호화하여 그 값에 서명자의 서명을 Exclusive-OR한 값과 일치하는지를 검사한다. 그리고 TS token을 확인한다. 이상이 없다면 서명자의 디지털 서명과 메시지에 이상이 없기 때문에 자신의 비밀키로 암호화된 랜덤키를 복호화하여 그 키로 암호화된 메시지를 복호화하여 평문 메시지를 얻는다.

계약서처럼 두 명의 사용자가 서명해야 하는 상호 서명 경우는 다음과 같다. 상호 서명은 위에서 설명한 검증 과정까지는 똑같다. 나머지 부분은 서명자의 서명 과정처럼 검증자가 받은 메시지에 서명한 후 다음과 같이 자신의 서명과 ID, TS token 등을 인증기관에게 보낸다.

ID B, User $A_{Signature}$, $CA_{Signature}$, User $B_{Signature}$, {M}K, TS token A, TS token B → CA

-User $B_{Signature}$:{h({M}K, H(RN'))||BP'||RN'} $BP_{AB\_User\ B}^{-1}$

인증기관은 인증 과정처럼 검증자의 서명 및 메시지의 이상 유무를 검사한다. 만일 이상이 없다면, 인증기관은 서명자와 검증자의 서명을 기존의 SN과 동시에 해쉬 알고리즘에 입력하여 해쉬값 h(SN, User $A_{Signature}$, User $B_{Signature}$)를 계산한다. 그 다음 기존의 자신의 서명을 제거하고, 그 해쉬값을 자신의 비밀키로 암호화하므로서 상호 서명을 인증하기 위한 서명 만든 다음 메시지에 첨부한다. 이때부터, 메시지의 효력이 발생할 것이다. 이후 인증기관은 서명들과 메시지, 그리고 자신의 인증서를 서명자와 검증자에게 보낸다.

SN, TS token A, TS token B, User $A_{Signature}$, $CA_{Signature}$, User $B_{Signature}$, {M}K, CA's Certificate → User A, User B

- $CA_{Signature}$: {h(SN, User $A_{Signature}$, User $B_{Signature}$)} P_{CA}^{-1}

4. 안전성

새로운 디지털 서명 방식은 2장에서 설명한 문제들을 해결한다. 다시 말해서 PKI의 어떤 객체라도 해쉬 알고리즘의 충돌 문제를 이용하여 서명자의 디지털 서명이 유효하도록 메시지를 위조할 수 없다. 검증자의 경우, 서명자의 RN과 BP를 모르기 때문에 해쉬값 h를 얻을 수 없어 위조를 할 수 없다. 상호서명의 경우, 서명자는 검증자의 RN, BP를 모르기 때문에 디지털 서명들이 모두다 유효하게 위조를 할 수는 없다. 제 3자인 인증기관 경우는 메시지의 내용을 알 수 없기 때문에 위조할 수 없다. 그러므로 새로운 디지털 서명 방식의 안전성은 해쉬 알고리즘과 상관없이 오직 공개키 암호에 연관된다.

두 개의 공개키 암호의 안전성은 결합된 서로 다른 공개키 암호의 안전성에 연관된다. 서로 다른 수학적 문제들을 바탕으로 하고 있기 때문에 공개키 암호들의 안전성이 비슷하다면 두 개의 공개키 암호는 하나의 공개키 암호의 안전성과 같다. 다시 말해서 두 개의 공개키 암호를 구성하는 공개키 암호들은 안전성이 비슷해야만 한다.

5. 결론

본 논문에서는 기존 디지털 서명 알고리즘의 문제점인 해쉬 알고리즘의 충돌 문제를 이용하여 위조하는 문제점을 해결하는 새로운 디지털 서명 방식을 제안하였다. 이 방식의 안전성은 해쉬 알고리즘의 안전성과 관계가 없으며, 오직 공개키 암호의 안전성에 연관 된다. 그러므로 이 새로운 방식은 기존 디지털 서명 방식에서 보안성은 낮지만 압축속도가 빠른 해쉬 알고리즘을 사용할 수 있다. 또한 기존 방식과도 혼용하여 사용할 수 있다. 즉, 디지털 서명에 강력한 보안성이 필요로 하면 새로운 방식을, 보안성이 필요 없으면 기존 방식을 사용 할 수 있다. 이에 따라 이 새로운 디지털 서명 방식은 E-commerce를 더욱 활성화 시킬 것으로 예상된다.

참고문헌

1. R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems" Communications of the ACM, 21(2):120-126, February 1978.
2. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Trans. Info. Theory, IT 31, 1985.
3. Digital Signature Standard (DSS). FIPS PUB 186, 1994.
4. I.B. Damgard, "A design principle for hash functions", Advances in Cryptology-Crypto '89, Lecture Notes in Computer Science, vol. 435, Springer-Verlag, 1990, pp. 416-427
5. H. Dobbertin, "The status of MD5 after a recent attack", RSA Laboratories, CryptoBytes, 2(2), 1996, The main result of this paper was announced at the Eurocrypt '96 rump session.
6. C. Adams, P. Cain, D. Pinkas, R. Zuccherato, "Internet X.509 Public Key Infrastructure Time Stamp Protocol", draft-ietf-pkix-time-stamp-00.txt, Sep. 1998