通信情報保護學會誌
第3卷 第1號, 1993. 3

디지틀 서명에 관한 고찰

이아란* · 송주석**

1. 디지틀 서명의 개요

현재의 컴퓨터 네트워크는 수천개의 터미날을 통해 액세스가 가능한 전세계적인 규모를 갖는 것으로 발전되어 왔다. 이에 따라 보안에 대한 요구는 점차로 중요한 문제가 되고 있다. 부가적으로 보안이 보장되지 않는 링크를 따라 송신이 이루어지는 네트워크내에서 송신자와 수신자가 가치있는 중요한 정보를 교환하기 위해서는 내용의 확인(validation)과 인증(authentication)이 보장되는 안전한 교환 방법이 요구된다. 내용의 확인(validation)이란 송신자가 보낸 정보의 내용이 통신도중에 제3자에 의해 변경됨이 없이 본래의 정보 그대로 임을 확인할 수 있는 방법을 말하며, 인증(authentication)이란 그 정보를 보낸 송신자가 A라는 것을 주장했을 때, 그것을 증명할 수 있는 방법을 말한다. 이 두가지 기능을 동시에 달성할 수 있도록 해주는 것이 디지틀 서명(digital signature)으로, 이는 송신되는 메세지에 덧붙여지거나 또는 메세지의 일부분으로 포함되어 메세지와 함께 수신자로 보내어진다. 디지틀 서명이 가져야하는 특성들을 요약하면 다음과 같다[10].

- 유일함 : 주어지는 디지틀 서명은 그 사용자만이 생성할 수 있어야 한다.
- 위조 불가능 : 디지틀 서명을 위조하고자 하는 사용자가 다른 사용자의 디지틀 서명을 생성할 수 없어야 한다.
- 인증의 용이함 : 수신자 또는 중재자(수신자와 송신자 사이의 논쟁(dispute)를 해결하는 제3자)가 사용자의 인증을 하기 쉬워야 한다.
- 부정의 불가능 : 디지틀 서명을 한 사용자가 자시이 한 디지틀 서명에 대해 부정할 수 없어야 한다.
- 생성의 용이함.

위의 특성은 손으로 쓰여진 서명이 가져야 하는 특징과 같다고도 할 수 있다. 그러나 디지틀 서명은 '0'과 '1'로 이루어진 비트열이므로 손으로 쓰여지는 서명과 다른 특성도 가져야 한다[1]. 즉, 손으로 쓰여지는 서명은 그 사용자에 따라 항상 일정하나 디지틀 서명은 각 메세지에 따라 동일한 송신자라 할지라도 생성되는 결과가 달라야 한다. 이러한 메세지 의존적인 특성에 의하여 디지틀 서명이 변하지 않는한 메세지의 내용중의 한 비트도 변화시킬 수 없으므로, 메세지가 전달되는 과정에서 내용이 변경되지 않았다는 것을 수신자는 디지틀 서명을 통해 알 수 있게 된다.

디지틀 서명의 방법은 인증에 참여하는 참가자들이 누구냐에 따라 크게 두가지로 나누어진다. 그 첫째는

* 연세대학교 전산과학과 석사과정
** 연세대학교 전산과학과 부교수

직접서명 또는 순수 서명 방법으로, 인증은 송신자 S와 수신자 R 사이에서만 이루어진다. 두번째는 간접 서명 또는 중재된 서명 방법으로 송신자 S와 수신자 R 사이에 중재자 A가 끼여 A는 S로부터 메세지와 디지틀 서명을 받아 인증을 한 후 그 결과와 메세지를 R로 보낸다. 이 방법에서 S와 R 사이에 논쟁이 발생할 경우는 A에 의해 해결이 가능하다.

디지틀 서명의 또 하나의 다른 분류 방법은 디지틀 서명에 사용되어지는 암호 시스템에 의한 분류로 비밀키 암호 시스템에 기반을 둔 디지틀 서명 방법과 공개키 암호 시스템에 기반을 둔 디지틀 서명 방법으로 나누어진다. 각각의 디지틀 서명 방법에 대해 자세히 살펴 보기 전에, 다음 절에서는 이에 사용되어지는 암호 시스템을 간략하게 알아보고 넘어가도록 한다. 또, 메세지 그 자체에 암호화를 적용하여 디지틀 서명을 생성할 경우에는 원문과 똑같은 길이 또는 그 이상의 디지틀 서명이 생성되므로 이는 시간적으로도 공간적으로도 비효율적이다. 그러므로 보통의 경우 디지틀 서명은 본문을 압축하여 적당한 길이로 줄인 후 암호 시스템을 적용하는 것이 일반적이므로, 이에 필요한 압축방법에 대해서도 알아보기로 한다.

2. 디지틀 서명의 기반을 이루는 기법

2.1. 비밀키와 공개키 암호 시스템

현재 컴퓨터의 보안을 해결하기 위해 사용되는 암호화 시스템은 크게 비밀키 암호 시스템과 공개키 암호 시스템으로 나눌 수 있다[4]. 공개키 암호 시스템은 비밀키 암호 시스템에 비해 비교적 최근에 발표된 것으로 1976년, Diffie와 Hellman에 의해 새로운 암호 시스템으로써 제안되었다. 비밀키 암호 시스템은 원문을 암호화(encryption), 복호화(decryption) 하는데 있어서 동일한 키(key)를 사용하며, 이 키는 암호문을 주고 받는 당사자들만이 고유하는 비밀키이다. 사용되는 키가 하나이므로 비밀키 암호 시스템은 단일키 암호문 시스템이라고도 불린다. 이에 반해 공개키 암호 시스템은 암호와 하는 절차(키)와 복호화하는 절차(키)가 서로 틀리고 암호화하는 절차(키)는 일반에게 공개되어지지만 복호화하는 절차(키)는 암호문을 받아 암호를 푸는 측밖에 알지 못한다.

비밀키 암호 시스템은 이를 사용하여 통신하는 사용자 둘 사이에 쌍방향(two-way) 채널을 제공할 수 있다는 장점이 있다. 즉, A와 B가 비밀키 암호 시스템을 사용하여 통신을 할 경우 하나의 키를 사용하여 A가 암호화한 메세지를 B가 받아 복호화할 수도 있고, B가 암호화하여 보낸 암호문을 A가 받아 복호화할 수도 있다. 이런 대칭적인 특성 때문에 비밀키 암호 시스템은 대칭적(symmetric) 암호 시스템이라고도 불리고 (공개키 암호 시스템은 비대칭적(asymmetric) 암호 시스템이라고도 불린다), 이런 특성이 비밀키 암호 시스템이 갖는 주요 장점이라 할 수 있다. 또, 키가 공개되지 않고 비밀로 남아 있는 한은 암호화된 메세지를 통해 자동적으로 인증 메카니즘이 제공되어진다. 즉, 비밀키 암호 시스템에서 키를 공유하고 있는 것은 당사자인 A, B 이외에는 없으므로 암호문을 보낸 것은 자연히 암호화를 할 수 있는 키의 소유자라는 것이 증명된다.

이런 장점이 있는 반면, 비밀키 암호 시스템은 다음과 같은 단점을 갖는다. 첫번째로 통신하고자 하는 당사자들 사이의 비밀키의 분배문제가 있으며, 두번째 문제는 키의 수가 통신망을 이용하는 사용자의 수에 제곱에 비례하여 증가한다는 것이다[8].

이러한 문제를 고려 1976년 Diffie와 Hellman이 처음으로 제안한 공개키 암호 시스템은 그 뒤 Rivest, Shamir, Adleman에 의해 제안된 RSA 방법을 통해 발전하였다[9]. 공개키 암호 시스템은 앞에서 든 비밀키 암호 시스템의 단점들을 극복하였다. 여기서, 공개된 암호 절차 E와 사용자만이 아는 복호화 절차 D는 다음과 같은 특성을 가져야 한다.

(a) 메세지 M을 암호화한 뒤 복호화하면 다시 원래의 M을 얻을 수 있어야 한다. 즉, D(E(M)) =M.

(b) 절차 E와 D는 쉽게 계산할 수 있어야 한다.

(c) 절차 E를 공개함으로써 D를 쉽게 계산하는 방법을 드러내지 않도록 고안되어져야 한다. 즉, 절차 D의 정당한 사용자만이 쉽게 D를 계산할 수

있어야 한다.

(d) 메세지 M을 먼저 복호화한 뒤 암호화한 결과는 M이어야 한다. 즉, D(E(M))=M.

(a)-(c)의 특성을 만족하는 E를 trap-door 일방향 함수라고 하고 (d)의 특성까지 포함하여 만족하는 E는 trap-door 일방향 치환(permutation)이라고 한다. 여기서 "일방향(one-way)"이라함은 한방향으로의 계산은 쉽게 할 수 있으나 반대방향으로의 계산은 무척 어려움을 나타낸다. "trap-door"라고 이름에 붙은 것은 trap-door 정보만 알려지는 경우에 역함수의 계산도 쉽게 할 수 있음을 나타낸다. (d)의 특징을 만족하는 경우를 함수라 하지 않고 "치환"이라고 하는 것은 이 특성을 만족함으로 하여 모든 메세지는 어떤 다른 메세지의 암호문이 되고, 따라서 암호문 자체도 다른 암호문의 암호문이 되어 즉, 원 메세지의 치환이라고 볼 수 있기 때문이다. 이 (d)의 특성을 이용함으로써 디지틀 서명에 공개키 방식을 사용할 수 있게 된다.

RSA 암호 시스템은 바로 앞의 특성을 만족하는 trap-door 일방향 치환 암호 시스템이며 그외의 공개키 암호 시스템으로 knapsack 시스템 등이 있고 비밀키 암호 시스템으로는 NBS(Natiolal Bureau of Standards)가 채택한 DES가 대표적이다.

2.2. 메세지의 압축 기법

가장 간단한 디지틀 서명의 방법은 메세지 자체를 암호화하여 이루어진다. 그러나 이 경우에는 앞에서도 말한 바와 같이 수 페이지에 걸친 메세지와 동시에 그와 같은 길이의 디지틀 서명을 통신하는 과정에 다루어야 하므로 여러가지면에서 효율적이지 못하다. 그러므로 디지틀 서명을 하기 전에 일방향 함수에 의한 데이타 압축 기법을 적용하여 더 짧은 길이의 디지틀 서명을 가능하게 한다.

일반적으로 압축 기법은 N비트의 메세지 M을 n비트의 요약(digest) CF(M)으로 변환하는 것이다. 디지틀 서명에 사용되는 압축 기법은 두개의 다른 메세지에 대해서는 각각 다른 요약이 생성되도록 고안되어야 한다. CF(M)은 메세지의 모든 비트의 값에 의해서 결정되는 랜덤한 수이므로 CF(M)을 만들어내는 함수를 해쉬함수(hash function)이라 하기도 한다.

메세지의 요약이 64비트로 표현되어진다고 가정하자. 이 때, 제3자가 의도적으로 원 메세지 M과 같은 요약을 갖는 위조 메세지 M′, 즉 CF(M)=CF(M′)과 M<>M′을 만족하는 M′을 찾고자 한다면 다음과 같은 방법으로 이를 수행할 수가 있다.

우선 첫번째 단계로 원래의 메세지 M의 일부의 단어들을 다른 단어와 치환하여 다양한 변형들을 만들어 낸다. 앞의 조건들을 만족하는 위조 메세지를 찾아낼 성공률을 높이기 위해서는, 이렇게 만들어 내는 변형들의 수를 생성 가능한 요약의 수만큼(이 예의 경우는 2^{64}개) 작성해 낸다. 또, 위조 메세지의 변형도 같은 수만큼 비슷한 절차에 의해 만들어 낸다. 이어, 각각에 대해 2^{32}개씩의 요약을 생성한다. 이렇게 생성한 M의 변형의 요약의 집합과 M′의 변형의 요약의 집합을 비교하여, 같은 값을 갖는 요약을 찾아냄으로써, 이 침입자는 요약이 동일한 위조 메세지 M′을 얻을 수 있게 된다. 이렇게 하여 성공할 수 있는 확률은 거의 0.5에 다르며 이와같은 종류의 공격(attack)을 생일 패러독스(birthday paradox)에 기반을 둔 공격이라고 부른다[3].

요약의 길이가 n인 경우 각 요약의 집합은 $2^{n/2}$개의 원소를 포함하도록 만들어야 하며, 이 두 집합을 비교할 때에는 각 집합을 크기에 따라 작은 것부터 정렬(sort)시킨 후에 비교한다. N개의 데이타를 정렬하는 것은 O(nlogn)의 복잡성(complexity)의 문제이므로 이 공격은 허용 가능한 시간내에 충분히 성공할 수 있다. 그러므로 이 공격을 막기 위해서는 두가지 방안이 제안된다. 첫째, 보통 사용되는 압축 과정을 두 차례에 걸쳐서 적용하며, 두번째로 랜덤한 패러미터값을 갖는 초기치를 주어 압축을 수행한다. 이 두 가지를 모두 채택한 것이 그림 1이다.

여기서 E_k는 비밀키 암호방식을 사용한 암호 함수이며 IC는 초기치이다. 메세지 M은 각가 64비트 메세지(M_2 …, M_k)로 나누어지고 앞 암호 함수의 결과가 그 다음 이어지는 함수의 입력이 된다. 즉 IC는 M_1을 키로하여 암호화되고 그 결과 C1은 M2를 키로하여 암호화된다. 따라서 일반적으로 암호문

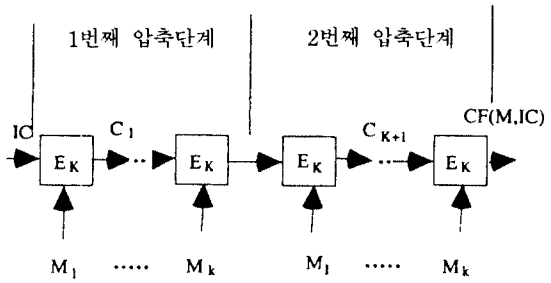


그림 1. 간단한 압축 시스템

C_i는 메세지 M_{i+1}를 키로하여 암호화되고 그 결과인 C_{i+1}은 그다음 함수의 입력이 된다. 이런 과정을 두번 반복한 뒤 마지막으로 64비트의 요약 CF(M, IC)를 얻을 수 있으며, 이는 초기치인 IC와 메세지 M에만 의존하는 값이다.

또 다른 비밀키 암호기법을 이용한 압축 기법이 그림 2에 나타나 있다.

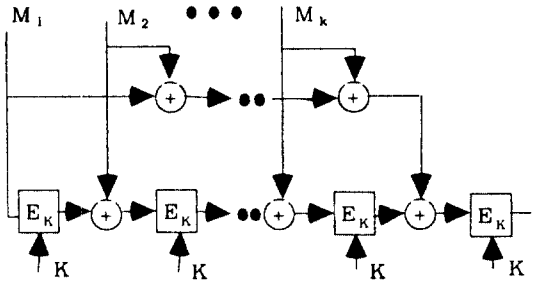


그림 2. 하나의 단계로 이루어진 압축 시스템

이 기법에서는 모든 E_k에 대한 키는 동일하다. 이 기법에서는 다음과 같은 과정을 통해 요약 CF_k(M)이 생성된다.

$$\begin{aligned} &C1=E_k(M_1)\\ &C2=E_k(M_2 \oplus C_1)\\ &\cdots\\ &Ck=Ek(Mk \oplus Ck\text{-}1)\\ &C2=Ek(M1 \oplus M2 \cdots Mk \oplus Ck) \end{aligned} \quad (1)$$

여기서 Mi(1<=i<=K)는 M을 요약과 같은 길이의 메세지로 나눈 것이다. 이때 요약 CF_k(M)는 전적으로 메세지에만 의존하는 결과로 나타난다. 이 방법에서는 생일 패러독스류의 공격을 막기 위해 마지막 암호화 함수의 입력 C_k에는 모든 M_i(1<=i<=K)가 포함된다. 이런 압축기법에 사용하기에 적절한 방법으로 DES를 들 수 있다[2].

3. 암호 시스템에 의한 디지틀 서명의 분류

3.1. 비밀키 암호 시스템을 기반으로 한 디지틀 서명

3.1.1. Diffie-Lamport 서명 기법

이 기법에서는 n비트의 메세지에 대해 어떠한 압축방법도 작용하지 않고 디지틀 서명을 생성한다[7]. 송신자가 n비트의 메세지를 보내고자 할 때, 우선, 다음과 같이 n개의 키의 쌍을 선택한다.

$$(K_{10}, K_{11}), (K_{20}, K_{21}), \cdots, (K_{n0}, K_{n1}) \quad (2)$$

이들은 송시자만이 알고 있는 키이다. 그리고나서 송신자는 S와 R이라고 하는 다음과 같은 열을 생성한다.

$$\begin{aligned} &S=[(K_{10}, K_{11}), (K_{20}, K_{21}), \cdots, (K_{n0}, K_{n1})]\\ &R=[(R_{10}, R_{11}), (R_{20}, R_{21}), \cdots, (R_{n0}, R_{n1})] \end{aligned} \quad (3)$$

이들은 후에 디지틀 서명을 인증하는 과정에서 사용된다. 송신자가 이들을 생성하는 방법은 우선 S의 원소들을 임의로 선택하고, 선택된 S로부터 다음과 같은 암호화를 거쳐 R을 생성한다.

$$R_{ij}=E_{kij}(S_{ij}) \quad i=1,\cdots,n, j=0,1 \quad (4)$$

암호시스템의 구조에 의해 S와 R의 길이는 정해진다. 이 S와 R은 사전에 수신자에 알려져야 한다. 그러므로 S와 R은 공공 레지스터(public register)에 저장되고 송신자와 수신자 이외의 제3자는 이 데이타를 읽을 수는 있으나 내용을 변경할 수는 없다.

n비트의 메세지 M이 $M=(m_1,\cdots,m_n)$, m_i=0, 1 (1<=i<=n)이라고 할 때 M의 디지틀 서명은 다음과 같은 비밀키의 열로 이루어진다.

$$SG(M)=(K_{1i_1}, K_{2i_2}, \cdots, K_{ni_n}) \quad (5)$$

이때 $m_j=0$이면 $i_j=0$이고 $m_j=1$이면 $i_j=1(1<=j<=n)$이다. 예를 들어, 6비트의 메세지 M=(1,0,1,1,0,0)이 있다고 할때 디지틀 서명은 다음과 같다.

$$SG(M)=(K_{11}, K_{20}, K_{31}, K_{41}, K_{50}, K_{60})$$

수신자는 SG(M)을 받아 해당하는 S와 R의 쌍이 그 키에 의해 암호화하면 서로 일치하는가를 보아 인증을 할 수 있다. 앞의 예의 경우에는 송신자로부터 알려진 키($K_{11}, K_{20}, K_{31}, K_{41}, K_{50}, K_{60}$)를 가지고, 수신자는 공공 레지스터로부터 이 키들의 인덱스에 해당하는 S의 원소를 읽어와 다음과 같이 암호화한다.

$$E_{11}(S_{11})$$
$$E_{20}(S_{20})$$
$$E_{31}(S_{31})$$
$$E_{41}(S_{41})$$
$$E_{50}(S_{50})$$
$$E_{60}(S_{60})$$

이 값들의 결과가($R_{11}, R_{20}, R_{31}, R_{41}, R_{50}, R_{60}$)과 같다면 수신자는 디지틀 서명 SG(M)을 진짜로 받아들인다.

이 방법의 가장 큰 단점은 서명의 길이가 길다는 것이다. 2.2. 절에서도 언급한 바와 같이 이런 단점을 극복하기 위해서는 Diffie-Lamport기법에 압축을 적용하여 n비트의 메세지를 r비트의 요약으로 줄인 후 디지틀 서명을 생성할 수 있다[10]. (그림 3)

이 경우 수신자의 인증 과정도 몇 단계를 더 거쳐서 이루어져야 한다. 첫째로, 수신자는 메세지의 요약을 생성해야 한다. 즉, 압축 기법은 공개되어 있어야 할 것이다. 그 다음 단계로 수신자는 공공 레지스터에서 필요한 r개의 S의 원소를 읽어와 SG(M)에 포함된 키로 암호화하여 같이 읽어온 r개의 R의 원소와 비교한다(그림 4). 비교한 결과가 같으면 인증은 성공하여 그 서명은 받아들여진다.

이 기법에서는 전송되는 SG(M)을 통해 r개의 키가 노출된다. 반면, 그것들과 쌍을 이루는 또다른 r개의 키는 아직은 비밀이 보장된 상태이다. 그러나

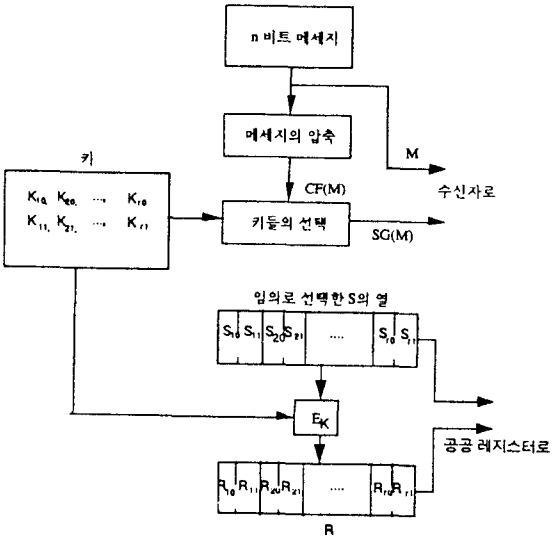


그림 3. Diffie-Lamport 서명 절차(송신자측)

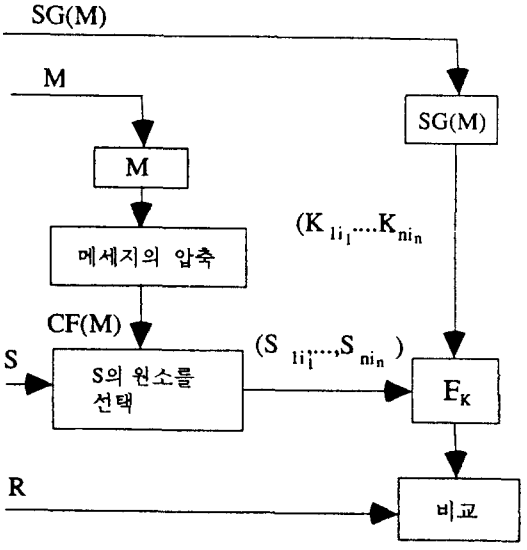


그림 4. Diffie-Lamport의 서명의 인증 절차

이를 반복하여 사용할 경우, 궁극적으로 모든 키가 노출이 될 수도 있다. 그런 위험을 막기 위해서 이 기법을 사용할 경우에는 생성된 키는 단 한번만 사용하도록 한다.

3.1.2. Matyas-Mayer 서명 기법

1981년에 Matyas와 Mayer에 의해 제안된 이 기법은 DES 앨거리듬을 기반으로 하고 있다. 디지틀 서명을 만들어 내기 위해 송신자는 적절한 패러미터를 결정한다. 첫째로 다음과 같은 행렬을 난수 발생기에 의해서 생성해낸다.

$$U=[u(i, j)] \quad 1<=i<=30, \quad 1<=j<=31 \qquad (6)$$

U의 각 원소는 64비트의 열이다. 그 다음으로 송신자는 다음과 같은 비밀키의 행렬을 생성한다.

$$K=[k(i, j)] \quad 1<=i, j<=31 \qquad (7)$$

이 행렬의 첫째 행, 즉 $k(1, 1)\cdots, k(1, 31)$은 임의로 선택되어지는 비밀키이지만, 그외의 다른 열들은 다음과 같은 방정식에 따라 계산되어진다.

$$k(i+1, j)=E_k(i, j)\ [u(i, j)] \quad 1<=i<=30, \quad 1<=j<=31 \qquad (8)$$

여기서 E_K는 DES의 암호화 함수를 뜻한다. K를 계산한 후 U의 모든 원소와 K의 마지막 행, 즉 $k(31, 1)\cdots, k(31, 31)$을 공공 레지스터와 수신자에게로 보낸다. 여기까지가 디지틀 서명을 하기 위한 준비 단계이다.

이 초기 단계 끝나면 송신자는 보내고자 하는 메세지 M을 압축하여 요약 CF(M)을 생성한다. 여기서 사용되는 압축기법은 2.2.절에서 두번째로 설명된 방법(식 (1))에 의한다.

얻어진 CF(M)로부터 송신자는 다음과 같은 열을 계산해 낸다.

$$b_i=E_{ki}(CF(M)) \quad 1<=i<=31 \qquad (9)$$

여기서 사용되는 31개의 키 $(K_1\cdots, K_{31})$은 공개되어 있는 것으로 공공 레지스터에 저장되어 있다. 이제 bit($1<=i<=31$)의 값을 이진수의 숫자로 보고 그 값의 크기에 따라 정렬시킨다. 이 정렬된 b_i의 열에서 얻은 정보로 행렬 K의 원소 중 31개의 키를 선택함으로써 디지틀 서명을 구한다. 만약 정렬된 b_i의 열이 $(b_{i_1}\cdots, b_{i_{31}})$과 같다면 디지틀 서명은 다음과 같다.

$$[k(i_1, 1), k(i_2, 2), \cdots, k(i_{31}, 31)] \qquad (10)$$

수신자는 송신자가 거친 과정을 반복 수행하여 디지틀 서명을 인증한다(그림 5). 우선 수신자는 메세지 M의 요약 CF(M)을 생성하여 b_i의 열을 계산한다. 계산한 b_i는 크기에 따라 정렬한다. 그런 다음 송신자가 보낸 디지틀 서명내의 키들을 빈 행렬 K에 $(b_{i_1}\cdots, b_{i_{31}})$가 나타내는 값에 따라 배치한다. 수신자는 식 (8)에 의한 행렬 K의 계산 방법을 알고 있기 때문에 각 열의 배치된 키 밑의 값들을 계산할 수 있다. 이렇게 해서 행렬 K의 마지막 행의 모든 값을 얻을 수 있고, 이 결과와 공공 레지스터에 저장된 K의 마지막 행의 값을 비교하여 인증을 할 수 있다.

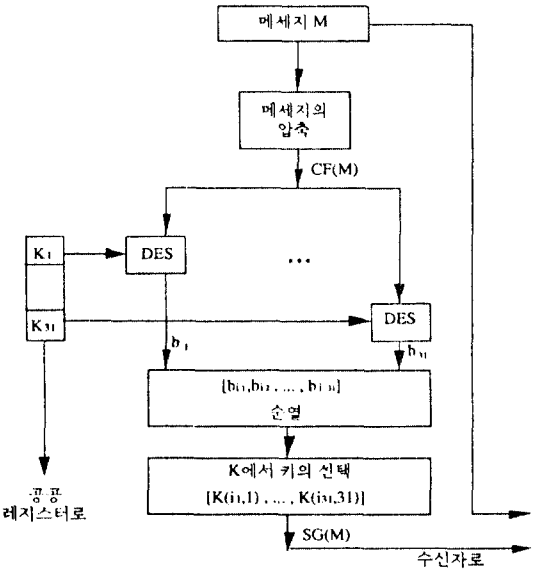


그림 5. Matyas-Meyer의 서명 기법

이 기법과 Diffie-Lamport 서명 기법은 둘다 비밀키 암호 시스템을 기반으로 하고 있는데, 두 경우 다 서명을 인증하기 위해서는 비밀 정보의 일부분을 노출해야 한다. 또한 실제 디지틀 서명을 하기 전에 공공 레지스터에 필요한 정보를 미리 저장해 놓아야 한다. 이런 점외에, 이 기법의 가장 큰 단점이라고 할 수 있는 것은 이러한 방법에서 생성되는 패러미터, 키들의 값등은 한번밖에 사용하지 못한다는 점이다.

즉 각각의 디지틀 서명에 대해, 송신자는 새로운 패러미터를 생성하여 저장해야 한다.

3.2. 공개키 암호 시스템을 기반으로한 디지틀 서명

3.2.1. RSA 서명 기법

RSA 암호 시스템을 사용하는 사용자는 우선 두 개의 큰 소수 p와 q를 임의로 선택하여, p와 q의 곱 m을 계산한다. m은 공개키의 일부분으로 공개되지만 p와 q는 비밀키의 소유자만이 알고 있는 값이다. 여기서 m은 충분히 큰 값으로 암호를 깨고자 하는 제3자가 m으로부터 p와 q를 유도해내기가 거의 불가능할 정도여야 한다. 이 요구를 만족시키기 위해서는 m이 이진수로 표현될 때 500 자리수 이상이어야 한다.

암호화해야할 메세지 x는 0과 m-1 사이의 수여야 하고(이 범위를 넘어가는 메세지는 적절히 나누어 이 범위안에 들도록 한다). x의 암호문 y는 $y=y^e \pmod m$으로 얻어지고, y로 부터 x로 복호화하는 함수는 $x=y^d \pmod m$이 된다. 여기서 e와 d는 p-1과 q-1에 대해 서로 소이며, 또 $ed=1 \pmod n$을 만족해야 한다. 단 n=(p-1)(q-1)이다. 이를 만족하는 e와 d에 대해서 앞의 두 함수 사이에는 서로 역함수의 관계가 성립된다. 이렇게 정해진 e는 m과 함께 공개키로서 공개되어지고 d는 비밀키로 사용된다.

RSA 암호 시스템에서 A의 공개키 K로 M을 암호화하는 것을 $E_{KA}(M)$으로, 복호화하는 것을 $D_{KA}(M)$로 나타내기로 할때, 다음과 같은 방법으로 디지틀 서명의 기능과 메세지의 보안 기능을 제공할 수 있다.

송신자 A는 보내고자 하는 메세지 M을 자신의 비밀키로 암호화한다. 그리고나서 수신자 B의 공개키로 암호화하여 이 결과인 $E_{KB}(D_{KA}(M))$을 송신한다(그림 6). 수신자는 $E_{KB}(D_{KA}(M))$을 받아 자신의 비밀키로 복호화를 하고, 그 결과를 송신자 A의 공개키로 암호화하여 따로 보내어진 메세지 M과 계산의 결과가 같은 경우 서명은 정당한 것으로 받아들여진다.

위와 같은 디지틀 서명 방법에서는 한가지 문제가

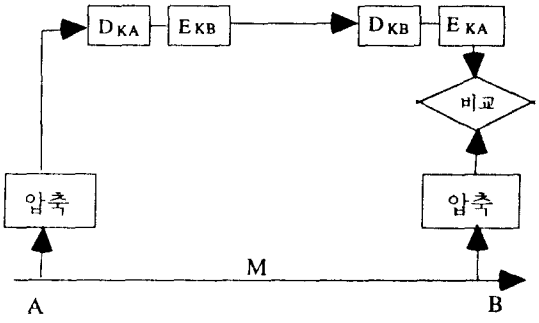


그림 6. RSA 서명 기법

발생한다. 즉 A와 B 사이에서 사용되는 m_A와 m_B는 서로 값이 다르므로 복호화 과정 D_{KA}, 즉 디지틀 서명 과정의 결과가 암호화 과정 E_{KB}의 입력의 범위를 벗어날 수도 있다. 이런 블럭킹(blocking) 문제를 해결하기 위해 Kohnfelder는 다음과 같은 해결방안을 제시하였다[6]. 즉 modulus 값이 적은 것을 먼저 계산하고 큰 값을 나중에 계산하도록 두 값 m_A와 m_B의 대소를 비교하여 암호화와 디지틀 서명의 순서를 조정해 준다.

modulus값 m_A와 m_B은 일반적으로 공개된 데이타이므로 송신자와 수신자는 둘다 사용하는 modulus값을 알고 있다. 따라서 이 해결방법은 이론적으로는 수행이 가능하다. 그러나, 실제 구현에서는 이런 식으로 modulus 값의 대소에 따라 디지틀 서명과 암호화의 문제를 바꾸어서 문제를 해결할 수는 없다. 암호화는 메세지가 전달되는 도중의 보안을 위하여 행하는 것이므로 전달이 끝난 후에는 즉시 복호화가 되고 그에 대해서는 더이상 고려를 하지 않으나, 디지틀 서명의 경우는 틀리다. 디지틀 서명은 마치 보통 서류에 손으로 기입된 서명과 같이 통신이 끝난 후에도 서명된 상태로 메세지와 함께 남아서 후에 그 메세지를 사용하고자 하는 사용자가 필요한 때에 인증을 할 수 있어야 한다. 그러므로 실제로는 디지틀 서명이 암호화 보다 항상 먼저 행해져야 한다. 즉, 디지틀 서명은 원 메세지에 직접 행해져서 통신 중에 행해진 암호화가 풀린 후에도 그 메세지의 사용자에 의해 언제나 인증이 가능해야 한다.

그래서 제시된 두번째 방법에서는 메세지 M을 압축하여 요약 CF(M)을 생성한 뒤 이를 비밀키로 암호화 함으로써 디지틀 서명 $D_{KA}(CF(M))$를 생성하여 메세지 M과 함께 수신자로 보낸다. 수신자는 이를 받아 M은 알려진 압축 기법에 의해 압축을 하고 디지틀 서명 $D_{KA}(CF(M))$는 송신자의 공개키로 암호를 푼다. 압축과 암호를 푼 두 결과가 같으면 디지틀 서명은 인증이 된 것이다.

3.2.2. Elgamal의 이산대수 문제를 기반으로한 서명 기법

이 기법은 Diffie-Hellman의 키 분배 방식을 기반으로 한다. 그러므로 서명 기법을 설명하기 전에 Diffie-Hellman 키분배 방식을 먼저 알아보기로 한다.

A와 B 사이에 비밀키 K_{AB}를 공유하고자 할 때 A와 B는 각각 임의의 수 x_A와 x_B를 선택한다. x_A와 x_B는 각각 자신만이 아는 비밀 값이다. p는 매우 큰 값의 소수이고 a는 GF(p)의 원시근이며 이 두 값은 공개되어 있어 공공 레지스터에 저장되어 있다. 그 다음, A는 $y_A=a^{x_A} \bmod p$를 계산하여 B로 보내고 B는 $y_B=a^{x_B} \bmod p$를 계산하여 A로 보내다. 그뒤 둘 사이의 비밀키 K_{AB}는 다음과 같이 계산된다.

$$\begin{aligned} K_{AB} &= a^{x_A x_B} \bmod p \\ &= y^{x_B} \bmod p \\ &= a^{x_A} \bmod p \end{aligned} \qquad (11)$$

A와 B는 위의 계산을 쉽게 할 수 있으나 다른 제 3 자는 x_A 또는 x_B값을 모르는 한 계산하기는 거의 불가능하다. 단 이 p를 선택할 때, p-1이 적어도 한 개 이상의 인수를 갖도록 정해야 한다. 만약 이 조건이 만족되지 않는 P를 택했을 때, 위의 이산대수 문제는 비교적 쉽게 풀려 암호가 깨질 위험성이 있다.

A가 B로 메세지 m(0<=m<=p-1)을 보내고자 할 때 위의 키분배 방식을 기반으로한 서명 기법은 다음과 같이 이루어진다[5]. 이때 A의 공개키는 a, p와 함께 $y_A=a^{x_A} \bmod p$가 되고 메세지 m의 디지틀 서명에 사용되는 비밀키는 x_A가 되며 x_A를 모르는 한 아무도 서명을 위조할 수는 없다.

메세지 m에 대한 서명은 (r, s)(0<=r, s<=p-1)의 쌍으로 이루어지며 이때 r과 s는 다음 방정식을 만족시키는 값이다.

$$a^m = y_A^r r^s \bmod p \qquad (12)$$

자세한 디지틀 서명의 절차는 다음과 같다. 우선 gcd(k, p-1)=1을 만족하는 0과 p-1 사이의 임의의 수 k를 하나 선택한다. 이 k값을 가지고 다음과 같이 r을 계산한다.

$$r=a^k \bmod p \qquad (13)$$

이때 (12)의 식은 다음과 같이 쓸 수 있다.

$$a^m = a^{xr} a^{ks} \bmod p \qquad (14)$$

그러므로 s는 다음 식을 이용해 풀 수 있다.

$$m=xr+ks \bmod (p-1) \qquad (15)$$

만약 k가 gcd(k, p-1)=1의 조건을 만족하도록 선택되어졌다면 (15)의 식은 근을 갖는다. 따라서 이렇게 구해진 r과 s를 디지틀 서명으로 하여 메세지 m과 함께 수신자 B에게 보내어진다. B는 m, (r, s)를 받아서 서명의 인증을 하게 되는데 식(12)을 이용하면 이는 쉽게 수행할 수 있다. 즉 a와 p는 공공 레지스터 내지는 공개된 화일등에 각 사용자마다 저장되어 있는 값이므로 식(12)의 양변을 이 값들을 이용해 각각 계산하여 얻어진 결과와 비교하면 된다. 이렇게 해서 얻어진 디지틀 서명은 m의 두배의 크기가 된다.

4. 디지틀 서명의 응용 분야

디지틀 서명의 가장 기본적인 기능은 통신 네트워크의 채널을 통해 교환되는 전자 메세지의 인증을 제공해 주는 것이다. 이 기능의 전형적인 응용분야로는 은행과 고객간 또는 중개 사업자와 고객간의 사무처리, 사령관으로부터 그의 부대로의 군사적 명령, 그리고 개인과 단체 사이의 계약, 합의등의 내용을 포함하는 다양한 문서의 교환등을 들 수 있다. 그러한 경우의 대표적인 예로 전자식 자금 이동(EFT : Electronic Funds Transfer) 시스템을 들 수 있다.

그 외에 디지틀 서명이 이용되는 분야에 소프트웨어의 배분이 있다. 네트워크상의 개개의 모든 노드들에게 개선된 소프트웨어를 배분하는 기능을 수행하는 중앙 소프트웨어 지급 센터가 있다고 하자. 개선된 소프트웨어를 각 노드에 배분할 때에는 항상 중앙 지급 센터가 디지틀 서명을 하고, 소프트웨어를 수신한 노드는 반드시 인증을 한 뒤 사용을 한다. 이렇게 함으로써 고의로 잘못된 소프트웨어를 네트워크내에 퍼뜨리려는 침입자로부터 각 노드를 보호할 수 있고, 또 실수로 다른 소프트웨어를 실행시키는 것을 막을 수도 있다.

디지틀 서명은 운영체제의 보안에도 응용될 수 있다. 즉, 정당한 권한이 부여되지 않은 프로그램이 수행되는 것으로 부터 운영체제를 보호하기 위하여 각 프로그램은 그 프로그램의 작성자에 의한 서명과 authority에 의한 서명을 포함하게 한다. 하드웨어는 정당한 서명이 없는 프로그램의 수행은 하지 못하게 함으로써, 위의 목적을 달성할 수 있고 아울러 이 서명은 소프트웨어 바이러스를 감지하는 한 수단이 되기도 한다.

앞의 예들로부터도 알 수 있듯이 디지틀 서명의 중요성은 나날이 증가할 것으로 보인다. 이에 따른 디지틀 서명의 표준화 문제도 고려되어져야 할 것이다. 1991년 8월 미국에서는 NIST(National Institute of Standards and Technology)에 의해 DSA (Digital Signature Algorithm)이 기밀이 아닌 정부의 정보와 보안이 요구되는 공공의 데이타를 위한 디지틀 서명의 표준안으로 제안되었다. 그러나 다수의 암호학자들과 그 분야에 종사하는 사람들에 의해 여러 문제점이 지적되어 표준안으로 채택되기에는 어려움이 있을 것같다[11]. 무엇보다도, 현재 미국 컴퓨터 산업시장의 3분의 2이상이 RSA를 사용하고 있고, ISO, CCITT, SWIFT등의 국제 표준 조직들도 표준안으로써 RSA를 채택하고 있다는 점도, 새로운 독자적인 앨거리듬을 이용한 DSA를 표준안으로 채택하는 것의 장애가 된다. 디지틀 서명의 표준화에 대한 문제는 앞으로도 더욱 고려되어야 할 문제이다.

참 고 문 헌

1. Akl, S. "Digital Signaturea : A Tutorial Survey", Computer, Vol. 16, No. 2, Feb. 1983. pp. 15-26.

2. Davies, D. "Applying the RSA Digital Signature to Electronic Mail", Computer, Vol. 16, No. 2, Feb. 1983, pp. 55-62.

3. Davies, D., and Price, W. Security for Computer Networks, Wiley 1984.

4. Diffie, W., and Hellman, M. "New Direction in Cryptography", IEEE Trans. Info. Theory, Vol. IT-22, No. 6, Nov. 1976, pp. 644-654.

5. Elgamal, T. "A Public Key Cryptosystem and a Signature Scheme Based on Discrte Logarithms", IEEE Trans. Info. Theory. Vol. IT-31, No. 4, July. 1985, pp. 469-472.

6. Kohnfelder, L. "On The Signature Reblocking Problem in Public-Ky Cryptosystems", Comm ACM, Vol. 21, No. 2, Feb. 1978, pp. 179.

7. Meyer, C., and Matyas, S. Cryptography : A New Dimension in Computer Data Security, Wiley 1982.

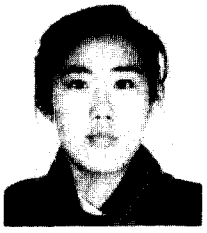
8. Pfleeger, C. Security in computing, Prentice-Hall 1989.

9. Rivest, R. et al. "A Method of Obtaining Digital Signatures and Public-Key Cryptosystems", Comm ACM, Vol. 21, No. 2, Feb. 1978, pp. 120-126.

10. Seberry, J., and Pieprzyk, J. Cryptography-An Introduction to Computer Security, Prentice-Hall 1989.

11. "Debating Encryption Standards", 'The Digital Signature Standard Proposed by NIST", "Responses to NIST'S Proposal", Comm ACM, Vol. 35, No. 7, July. 1992, pp. 23-54.

□ 著者紹介



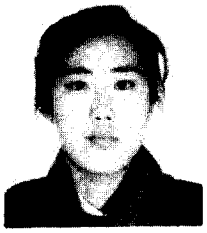
이 아 란

1992년 2월 연세대학교 전산과학과 학사
1992년 3월~현재 연세대학교 전산과학과 석사과정

송 주 석

1976년 2월 서울대학교 전기공학과 학사
1979년 2월 한국과학원 전기 및 전자 석사
1988년 8월 Univ. of California at Berkeley 박사
1988년 8월~1989년 9월 Naval Postgraduate School Assistant Professor
1989년 2월~현재 연세대학교 전산과학과 부교수

□ 著者紹介



이 아 란

1992년 2월 연세대학교 전산과학과 학사
1992년 3월~현재 연세대학교 전산과학과 석사과정

송 주 석

1976년 2월 서울대학교 전기공학과 학사
1979년 2월 한국과학원 전기 및 전자 석사
1988년 8월 Univ. of California at Berkeley 박사
1988년 8월~1989년 9월 Naval Postgraduate School Assistant Professor
1989년 2월~현재 연세대학교 전산과학과 부교수