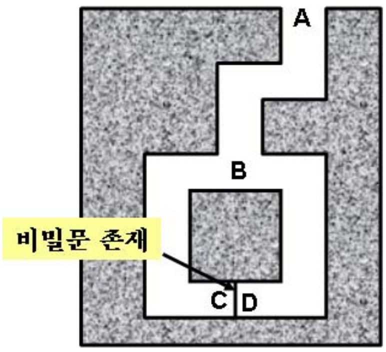
제 11 장 영지식 증명과 전자서명 프로토콜

11.1 영지식 증명

11.1.1 개요

영지식 증명(zero-knowledge proof)이란 어떤 내용을 알고 있을 때 그 내용을 직접 보여주지 않고 그것을 알고 있음을 증명하는 방법을 말하며, 이 때 증명하고자 하는 자가 그 지식을 알고 있다는 것을 제외하고는 어떤 정보도 확인자에게 노출되지 않아야 한다. 예를 들어 그림 11.1처럼 어떤 동굴 내부에 비밀문이 있다. Alice는 동굴의 비밀문을 알고 있으며, 이 문을 여는 비밀 주문을 알고 있다. 이것을 Bob에게 증명하고 싶지만 Bob에게는 비밀 주문을 가르쳐주고 싶지 않다.



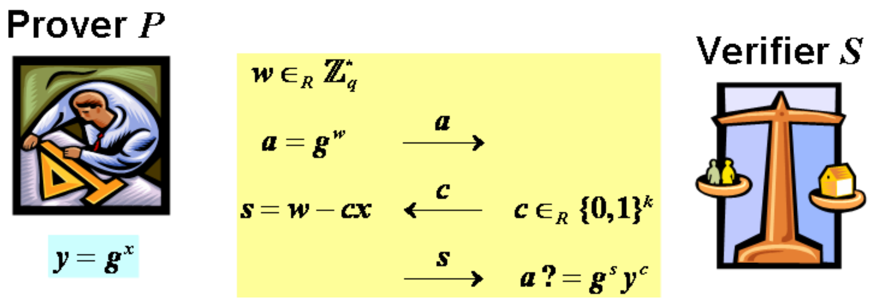
<그림 11.1> 동굴의 비밀문에 대한 영지식 증명

Alice는 Bob와 다음과 같은 프로토콜을 수행하여 Bob에게 비밀 주문을 가르쳐주지 않고 자신이 비밀 주문을 알고 있다는 것을 증명할 수 있다.

- **단계 1.** Bob은 A 위치에서 기다린다.
- **단계 2.** Alice는 C나 D까지 이동한다.
- **단계 3.** Bob은 B 까지 이동한다.
- **단계 4.** Bob은 왼쪽 또는 오른쪽 중 한 쪽으로 나오라고 Alice에게 외친다.
- **단계 5.** Alice는 비밀주문을 이용하여 Bob의 지시를 따른다.
- **단계 6.** 이 과정을 n번 반복한다.

이 프로토콜을 한 번 수행할 경우 Alice는 50%의 확률로 비밀 주문을 모르더라도 프로토콜 수행에 성공할 수 있다. 따라서 n번 모두 성공할 확률은 $1/2^n$이다. 그러므로 안전성을 높이기 위해서는 프로토콜의 수행 횟수를 늘려야 한다. 하지만 이 안전성은 Alice가 비밀 주문을 모르는 상태에서 프로토콜 수행에 성공할 확률을 말한다. Bob은 1회를 수행하나 여러 번 수행하나 비밀 주문을 절대 얻을 수 없다. 또한 Bob은 이 과정을 비디오로 녹화하여도 다른 사람에게 Alice가 이 사실을 알고 있다고 증명할 수 없다. 이것은 Bob과 Alice가 공모하여 가짜 비디오를 쉽게 만들 수 있기 때문이다.

11.1.2 영지식 증명의 예



<그림 11.2> 이산대수 영지식 증명 프로토콜

그림 11.2에 기술된 프로토콜을 사용하면 어떤 군 원소의 이산대수를 알고 있다는 것을 그 값을 노출시키지 않고 증명할 수 있다. 만약 증명자가 확인자가 전달하는 c를 미리 예측할 수 있으면 y의 g에 대한 이산대수를 모르더라도 다음과 같이 증명에 성공할 수 있다.

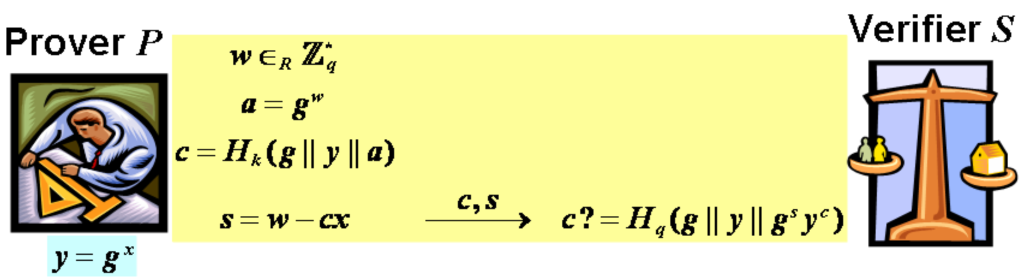
- $s \in_R \mathbb{Z}_q^*$를 임의로 선택한 다음에 $a = g^s y^c$ 값을 전달한다.

따라서 c의 길이가 k 비트이면 이산대수를 모르는 상태에서 증명에 성공할 확률은 $1/2^k$이다. 또 확인자는 유사한 다음과 같은 방법으로 가짜 트랜스크립트를 쉽게 구성할 수 있다.

- $s \in_R \mathbb{Z}_q^*$와 $c \in_R \{0,1\}^k$를 임의로 선택한 다음에 $a = g^s y^c$를 계산하고, a, c, s를 트랜스크립트로 사용한다.

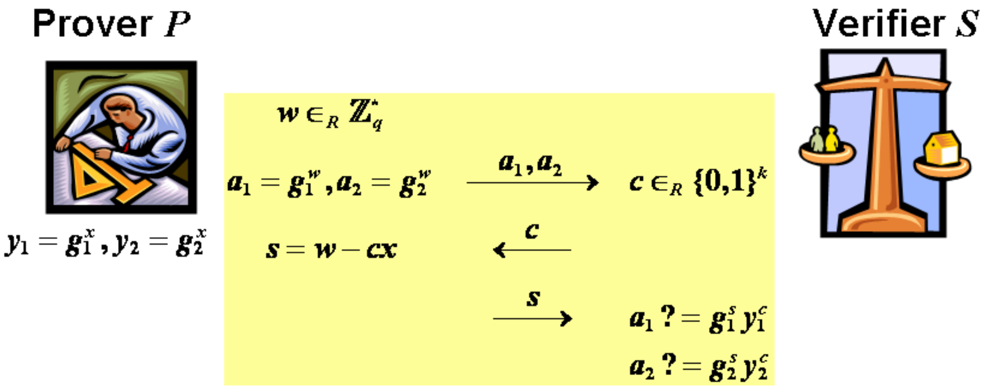
즉, 확인자는 이 프로토콜에 참여하여 증명자가 y의 g에 대한 이산대수를 알고 있음을 확인할 수 있지만 다른 사람에게 이 사실을 증명할 수 없다. 그림 11.2의 프로토콜에서 주의할 점은 증명자는 항상 다른 w를 사용해야 한다. 만약 두 개의 다른 프로토콜 수행에서 같은 w를 사용하면 공격자들은 $s = w - cx$와 $s' = w - c'x$를 이용하여 이산대수 x를 계산할 수 있다.

그림 11.2에 기술된 프로토콜은 증명자가 확인자와 상호작용을 해야 한다. 하지만 상호작용 없이 증명자가 홀로 증명을 그림 11.3과 같이 만들 수도 있다.



<그림 11.3> 이산대수 영지식 증명 프로토콜 비상호작용 버전

이산대수를 모르는 증명자가 $s \in_R \mathbb{Z}_q^*$를 임의로 선택하여도 $c = H_k(g\|y\|g^s y^c)$를 만족하는 c를 계산할 수 없으므로 속이는 것이 어렵다. 해쉬값의 길이가 k이면 임의의 k비트 길이의 값 c'을 선택하여 $H_k(g\|y\|g^s y^{c'})$이 c'과 같은지 비교할 수 있다. 이와 같은 전사(brute-force) 방법을 통해 c를 찾는 것은 $O(1/2^{k-1})$ 비용이 소요된다.



<그림 11.4> 이산대수 등가 영지식 증명 프로토콜

y_1과 y_2가 각각 g_1과 g_2에 대한 이산대수가 같을 경우에 이것을 영지식으로 그림 11.4의 프로토콜을 이용하여 증명할 수 있다. 이 증명을 이산대수 등가 영지식 증명이라 한다. 이 증명도 그림 11.3과 비슷한 방법으로 비상호작용 버전을 만들 수도 있다.

11.2 은닉 채널

Simmons는 1985년에 ElGamal 서명을 이용하여 은닉 채널(subliminal channel)을 만들 수 있음을 보였다. 은닉 채널이란 정당한 메시지에 정보를 숨겨 상대방에게 보내는 것이다. 외부 사람들은 이 메시지를 보면 정당한 메시지라는 것을 확인할 수 있지만 그 메시지에 숨겨져 있는 내용을 알아내거나 숨겨진 내용이 있다는 사실 조차 알 수 없다. 예를 들어 죄수들이 평범한 내용의 쪽지를 서로 주고받았다고 하자. 간수는 이를 전달해주면서 내용을 검토하였지만 문제가 되는 내용은 없었다. 그러나 이 평범한 쪽지에는 탈옥 계획이 포함되어 있었다면 은닉 채널이 쪽지에 존재한다는 것을 의미한다.

Simmons가 발견한 ElGamal 서명에 대한 은닉채널은 다음과 같다. Alice는 m'에 대한 ElGamal 서명을 보내지만 거기에 m을 숨겨 Bob만 볼 수 있도록 전달하고 싶다. 이 때 Alice의 서명키는 x_A이고, 확인키는 $y_A = g^{x_A}$라 하면 다음과 같은 방법을 통해 m을 은닉하여 Bob에게 전달할 수 있다. 단, Bob도 Alice의 개인키를 알고 있어야 한다.

- 다음에 주어진 (W, s) 쌍은 m'에 대한 Alice의 ElGamal 서명이다.

$$W = g^m \bmod P,\ s = (H_{p-1}(m') - x_A W) m^{-1} \bmod (p-1)$$

  - 이 서명은 다음 식을 이용하여 누구나 확인할 수 있다.

$$y_A^W W^s \ ? \equiv g^{H_{p-1}(m')} \pmod p$$

- Bob은 일반 사용자들과 마찬가지로 위 서명 확인 과정을 통해 서명의 유효성을 확인한 후에 m을 다음 식을 통해 얻을 수 있다.

$$m = s^{-1}(H_{p-1}(m) - x_A W) \bmod (p-1)$$

이 방법의 문제점은 Bob도 Alice의 개인키를 알고 있어야 한다. 현재는 이것이 가능하지 않은 은닉 채널 기법도 있다.

11.3 서버-지원 서명

대부분의 전자서명 기법은 서명자가 직접 서명을 생성한다. 서버-지원 서명 기법에서는 서명자 대신에 신뢰하는 서버가 대신 서명을 해준다. 이렇게 하면 효율성을 높일 수 있고, 시스템의 복잡성을 줄일 수 있다. 기존 서명 기법에서 서명키가 노출되면 공격자가 서명을 위조할 수 있으며, 위조된 서명의 수를 제한할 수 없다. 하지만 서버-지원 서명 기법을 사용하면 서명의 수를 제한할 수 있다. 서버-지원 서명의 기본적인 생각은 그림 11.5와 같다.

<그림 11.5> 서버-지원 서명 방식

서버는 사용자를 인증한 다음에 서명해야 할 문서를 서버에 전달한다. 서버는 서명을 생성하여 사용자에게 전달한다. 이 서명을 확인하고 싶은 확인자들은 서버에게 요청하여 서명을 확인한다. 여기서 사용자를 인증하는 방법은 중요하지 않다. 또한 서명 자체를 생성하지 않을 수도 있다. 단지 데이터베이스에 그 사실만 기록할 수 있다.

Asokan 등은 해쉬체인(hash chain)을 이용한 서버-지원 서명 기법을 제안하였다. 해쉬체인은 1981년에 Lamport가 처음 제안하였다. 해쉬체인은 다음과 같이 생성한다.

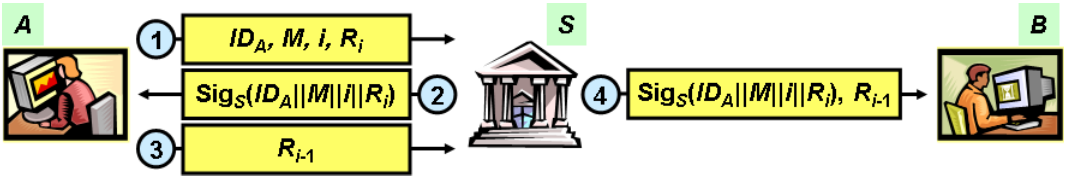
- 해쉬체인 생성 방법: 랜덤 값 R_0을 임의로 선택한 후에 $R_i = H(R_{i-1})$를 반복적으로 계산한다. 최종값 R_n은 해쉬체인의 루트(root)라 하고, 각 R_i는 토큰이라고 한다.

해쉬함수의 일방향성 때문에 R_i를 알아도 R_{i-1}를 계산할 수 없다. 만약 R_n이 사용자와 바인딩되어 있다면 R_{n-1}, R_{n-2}, ..., R_0 순으로 사용하여 사용자를 인증할 수 있다.

Asokan 등이 제안한 서버-지원 서명 기법은 다음과 같다.

- **단계 1.** Alice는 인증 서버에 해쉬체인의 루트인 R_n을 제출하고, $(ID_A \| n \| R_n \| ID_S)$에 대한 인증서 $Cert_A$를 발급 받는다. 여기서 ID_S는 사용자가 사용할 서명 서버의 식별자이다.
- **단계 2.** Alice는 $Cert_A$를 서명 서버 S에게 전달한다. 서버는 인증서를 확인하여 사용자를 인증한다.

● **단계 3.** Alice는 서명 서버 S와 그림 11.6의 프로토콜을 진행한다.



<그림 11.6> Asokan 등의 서버-지원 서명 방식

Alice는 유효한 서명을 받을 경우에만 그 다음 토큰을 제공한다. 서버는 언제든지 서명을 생성할 수 있지만 유효한 서명이 되기 위한 토큰 R_{i-1}은 Alice가 주지 않으면 계산할 수 없으므로 독자적으로 서명을 위조할 수 없다. 이 방식의 한 가지 단점은 서버는 i번째 서명을 여러 개 생성할 수 있다.

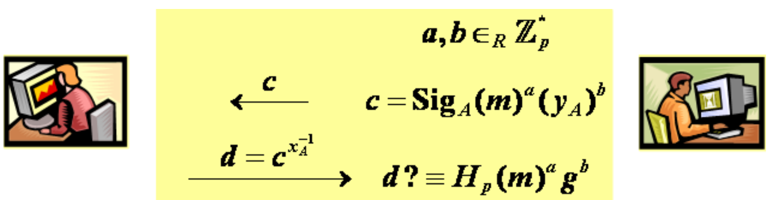
11.4 부인불가 서명

일반 전자서명은 원본과 복사본을 구분하기가 어렵다. 이 특성이 유용하게 사용될 수 있지만 반대로 부작용도 있다. 예를 들어 어떤 소프트웨어 회사에서 X라는 소프트웨어를 판매한다. 이 소프트웨어에 바이러스가 없음을 보장하기 위해 회사는 배포되는 소프트웨어마다 전자 서명을 포함한다. 그러나 이 서명은 오직 적법한 사용자만 확인할 수 있어야 하며, 만약 배포된 소프트웨어 중에 바이러스가 있을 경우에는 회사가 서명을 부인할 수 없어야 한다. 이것을 가능하게 해주는 것이 **부인불가 서명**(undeniable signature)이다. 부인불가 서명 방식도 일반 전자 서명처럼 서명은 서명할 문서와 서명키에 의존하지만 일반 전자 서명과 달리 서명자의 동의 없이는 서명을 확인할 수 없다.

Chaum은 다음과 같은 부인불가 서명을 제안하였다. 서명자 Alice의 서명키는 x_A이고, 확인키는 $y_A = g^{x_A}$이면 서명은 다음과 같이 한다.

$$\mathrm{Sig}_A(m) = H_p(m)^{x_A}$$

이 서명은 그림 11.7의 프로토콜을 수행하여 확인할 수 있다.



<그림 11.7> Chaum의 부인 불가 서명 방식의 서명 확인 프로토콜

즉, x_A를 모르는 사용자는 $\mathrm{Sig}_A(m)$를 확인시켜 줄 수 없다. Bob은 그림 11.7의 서명 확인 프로토콜을 직접 Alice와 수행하면 이 서명이 Alice의 서명임을 확인할 수 있다. 하지만 Bob은 이 과정을 다른 사람에게 보여주어도 그 사람은 Bob을 믿을 수 없다. 이것은 아무나 쉽게 다음과 같이 가짜 프로토콜을 만들 수 있기 때문이다.

- $a,b \in_R \mathbb{Z}_p^*$를 선택한 다음에 $c = \mathrm{Sig}_A(m)^a (y_A)^b$와 $d = H_p(m)^a g^b$를 계산한다.

11.5 지정된 확인자 서명

지정된 확인자 서명(designated confirmer signature)은 부인불가 서명과 일반 서명의 절충안이다. 부인불가 서명은 서명자의 도움 없이는 서명을 확인할 수 없는 문제점이 있다. 반면에 일반 서명은 누구나 항상 서명을 확인할 수 있다. 반면에 지정된 확인자 서명은 다음과 같이 진행된다.

- 서명자는 확인자와 프로토콜을 수행하여 서명을 전달한다.
  - 부인불가 서명과 마찬가지로 확인자는 이 프로토콜의 트랜스크립트을 이용하여 다른 사람들에게 서명의 유효성을 증명할 수 없다.
- 서명자가 지정한 사용자는 이 서명을 다른 사용자들에게 확인해 줄 수 있다.

11.6 프록시 서명

프록시 서명(proxy signature)은 서명자가 자신의 서명키를 지정된 프록시에게 주지 않고, 자신을 대신하여 서명할 수 있도록 해준다. 프록시 서명의 요구사항은 다음과 같다.

- **요구사항 1.** 구별가능성: 프록시 서명은 일반 서명과 구분될 수 있어야 한다.
- **요구사항 2.** 위조불가능성: 원 서명자와 지정된 프록시 서명자만 유효한 프록시 서명을 생성할 수 있다.
- **요구사항 3.** 프록시 서명자는 프록시 서명이 아닌 실제 서명은 할 수 없어야 한다.
- **요구사항 4.** 확인가능성: 확인자는 프록시 서명으로부터 원 서명자의 서명된 메시지에 대한 동의를 확인할 수 있어야 한다.
- **요구사항 5.** 식별가능성: 원래 서명자는 프록시 서명을 통해 프록시 서명자를 확인할 수 있어야 한다.
- **요구사항 6.** 부인불가능성: 프록시 서명자는 자신이 서명한 프록시 서명을 부인할 수 없어야 한다.

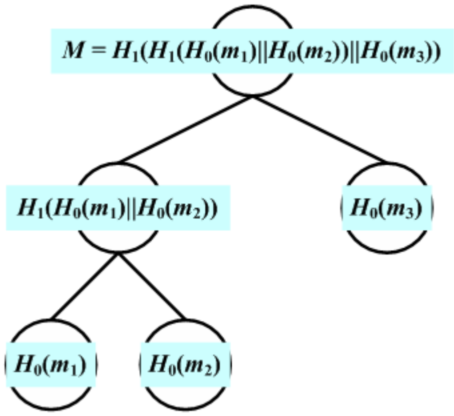
11.7 그룹 서명

다음과 같은 요구사항을 충족하는 서명을 그룹 서명(group signature)라 한다.

- **요구사항 1.** 그룹의 멤버들만 서명을 할 수 있다.
- **요구사항 2.** 확인자들은 그룹의 멤버 중 한 명이 서명하였다는 것을 확인할 수 있지만 누가 실제로 서명하였는지는 알 수 없다.
- **요구사항 3.** 분쟁이 발생한 경우에는 실제 서명자를 밝힐 수 있어야 한다.

11.8 일괄 서명

일괄 서명(batch signature)는 Fiat가 1989년에 처음으로 제안하였다. 이 서명 방식에서는 한 서명자가 하나의 메시지에 대한 서명 비용 정도로 여러 개의 다른 메시지들을 한꺼번에 서명할 수 있으며, 이 서명으로부터 저렴한 비용으로 개별 메시지 서명을 추출하여 다양한 사용자들에게 전달할 수 있다.



<그림 11.8> Pavloski와 Boyd의 일괄 서명 기법에서 사용하는 해쉬 이진 트리

Pavloski와 Boyd는 1999년에 해쉬 이진 트리를 이용한 일괄 서명 기법을 제안하였다. 이들이 제안한 서명 기법을 이용하여 메시지 m_1, m_2, m_3에 대한 일괄 서명을 하기 위해서는 그림 11.8과 같은 해쉬 이진 트리를 구성한다. 그 다음에 이 트리의 루트 값인 M에 대해 전자서명을 한다. 즉, 동시에 서명하는 메시지 수와 상관없이 한 번의 전자서명 비용만 소요된다. 일괄 서명 값에서 개별 서명 값은 다음과 같이 구성한다.

- 메시지 m_1에 대한 서명: $Sig_A(M)$, $[H_0(m_3),R]$, $[H_0(m_2),R]$, $[H_0(m_1),L]$
- 메시지 m_3에 대한 서명: $Sig_A(M)$, $[H_0(m_3),R]$, $[H_1(H_0(m_1)\|H_0(m_2)),L]$

11.9 다중 서명과 결합 서명

다중 서명(multi-signature) 기법이란 n명의 서로 다른 서명자가 같은 메시지에 대해 서명하지만 그 결과가 n개의 서명이 아니라 하나의 서명을 얻게 되는 기법을 말한다. 따라서 n개의 서명 대신에 하나의 서명을 확인하여 n명이 서명한 사실을 확인할 수 있다.

Boldyreva의 겹선형 쌍함수을 이용한 다중 서명 기법은 다음과 같다. 여기서 $G_1=\langle P\rangle$은 위수가 소수 q인 타원곡선 군이고, G_2는 위수가 소수 q인 곱셈군이며, $H:\{0,1\}^*\rightarrow G_1$과 $\hat{e}: G_1\times G_1\rightarrow G_2$는 각각 충돌회피 해쉬함수와 겹선형 쌍함수이다.

- 각 사용자의 개인키/공개키: $x_i\in_R\mathbb{Z}_q^*$, $Y_i=x_iP$

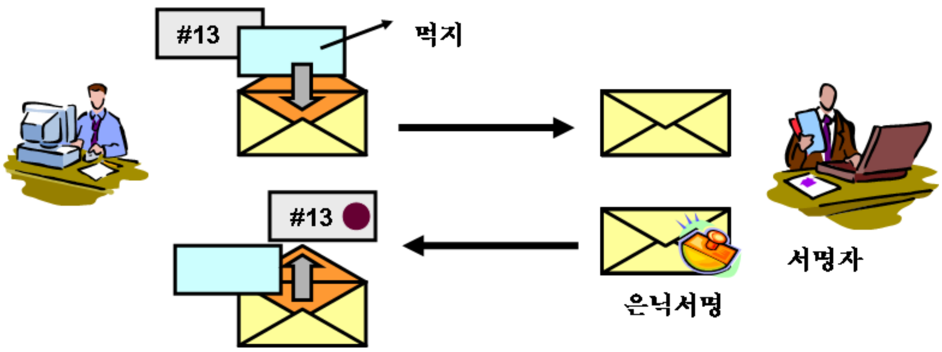
- 다중 서명 공개키: $Y = \sum_{i=1}^{n} Y_i = \Sigma_i x_i P$
- 메시지 m에 대한 각 사용자의 서명: $S_i = x_i H(m)$
- 다중 서명: $S = \sum_{i=1}^{n} S_i = (\Sigma_i x_i) H(m)$
- 다중 서명에 대한 확인: $\hat{e}(P, S) ?= \hat{e}(Y, H(m))$

결합 서명(aggregate signature) 기법이란 n명의 서로 다른 서명자가 서로 다른 n개의 메시지에 대해 서명하지만 그 결과가 n개의 서명이 아니라 하나의 서명을 얻게 되는 기법을 말한다. 따라서 n개의 서명 대신에 하나의 서명을 확인하여 n개의 서로 다른 메시지에 대한 n명의 서명을 확인할 수 있다.

Boneh의 겹선형 쌍함수을 이용한 다중 서명 기법은 다음과 같다. 각 사용자 i는 메시지 m_i에 대해 결합 서명을 한다. 여기서 $G_1 = \langle P \rangle$은 위수가 소수 q인 타원곡선 군이고, G_2는 위수가 소수 q인 곱셈군이며, $H: \{0,1\}^* \rightarrow G_1$과 $\hat{e}: G_1 \times G_1 \rightarrow G_2$는 각각 충돌회피 해쉬함수와 겹선형 쌍함수이다.

- 각 사용자의 개인키/공개키: $x_i \in_R \mathbb{Z}_q^*$, $Y_i = x_i P$
- 다중 서명 공개키: $Y = \sum_{i=1}^{n} Y_i = \Sigma_i x_i P$
- 메시지 m에 대한 각 사용자의 서명: $S_i = x_i H(m_i)$
- 다중 서명: $S = \sum_{i=1}^{n} S_i = \Sigma_i x_i H(m_i)$
- 다중 서명에 대한 확인: $\hat{e}(P, S) ?= \prod_{i=1}^{n} \hat{e}(Y, H(m))$

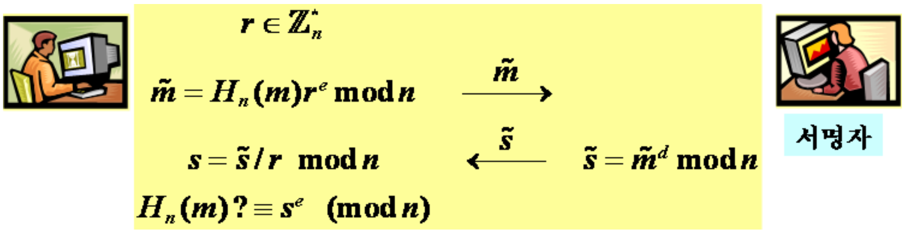
11.10 은닉 서명



<그림 11.9> 은닉 서명의 원리

본질적으로 서명자는 자신이 서명하는 내용을 알고 있어야 한다. 하지만 **은닉 서명**(blind signature)에서 서명자는 자신이 서명하는 내용을 전혀 알 수 없다. 이 서명 기법은 주로 전

자화폐나 전자선거 등에서 사용자의 익명성을 제공하기 위해 많이 사용된다. 은닉서명 프로토콜의 원리는 그림 11.9와 같다. 은닉 서명을 받고 싶은 사용자는 그 내용을 먹지와 함께 봉투에 봉합하여 서명자에게 전달한다. 서명자는 봉투를 개봉하지 않은 상태에서 서명을 한다. 먹지가 포함되어 있으므로 문서에 서명자에 서명이 기록된다. 요청한 사용자는 봉투를 개봉하여 먹지를 제거하고 문서만 사용하게 된다.



<그림 11.10> RSA 기반 은닉서명 프로토콜

RSA 기반 은닉서명 프로토콜은 그림 11.10과 같다. 여기서 r은 은닉 요소(blinding factor)라 한다. 서명자는 이 요소를 알 수 없으므로 나중에 s를 보더라도 이 s와 서명 순간을 연결시킬 수 없다. 은닉 서명은 서명자가 서명하는 내용을 볼 수 없으므로 매우 위험한 서명 기법이다. 이와 같은 위험을 줄이기 위해 사용할 수 있는 기법 중 하나가 cut-and-choose 기법이다. 이 기법은 케이크를 두 사용자 간에 공평하게 나누는 방법에서 유래되었다. Alice는 케이크를 두 조각으로 나눈 후, 각 조각을 동일한 크기의 박스에 포장한다. Bob은 두 박수 중 하나를 선택하고, 선택하지 않은 박스는 Alice가 가지게 된다. 따라서 Alice는 나중에 Bob이 어떤 박스를 선택할지 모르기 때문에 공평하게 나누지 않으면 본인이 손해를 보게 된다.

참고문헌

[1] C.P. Schnorr, "Efficient Signature Generation for Smart Cards," Advances in Cryptology, Crypto 1988, LNCS 403, pp. 239−252, 1990.

[2] G.J. Simmons, "The Subliminal Channel and Digital Signatures," Advances in Cryptology, Eurocrypt 1984, LNCS 209, pp. 364-378, 1985.

[3] N. Asokan, Gene Tsudik, and Michael Waidner, "Server-supported Signatures," J. of Computer Security, Vol. 5, No. 1, pp. 91−108, 1997.

[4] L. Lamport, "Password Identification with Insecure Communications," Communications of ACM, Vol. 24, No. 11, pp. 770−772, 1981.

[5] D. Chaum, "Zero-Knowledge Undeniable Signatures," Advances in Cryptology, Eurocrypt 1990, LNCS 473, pp. 458−464, 1991.

[6] D. Chaum, "Designated Confirmer Signatures," Advances in Cryptology, Eurocrypt 1994, LNCS 950, pp. 86−91, 1995.

[7] D. Chaum and Eugene van Heyst, "Group Signatures," Advances in Cryptology, Eurocrypt 1991, LNCS 547, pp. 257−265, 1991.

[8] Amos Fiat, "Batch RSA," Advances in Cryptology, Crypto 1989, LNCS 435, pp. 175−

185, 1991.

[9] Chris Pavlovski and Colin Boyd, "Efficient Batch Signature Generation using Tree Structures", Int. Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99), City University of Hong Kong Press, pp.70-77, 1999.

[10] Alexandra Boldyreva, "Threshold Signatures, Multisignatures, and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signatures Scheme," Proc. of PKC 2003, LNCS 2567, pp. 31–46, 2003.

[11] Dan Boneh, Craig Gentry, Ben Lynn and Hovav Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," Advances in Cryptology, Eurocrypt 2003, LNCS 2656, pp. 416–432, 2003.

[12] David Chaum, "Blind Signatures for Untraceable Payments," Advances in Cryptology, Proc. of Crypto 1982, pp. 199–203, 1983.