제 10 장 전자서명

10.1 전자서명 개요

10.1.1 전자서명의 요구사항

전자서명은 기존 서명을 전자적으로 구현한 것으로서 기존 서명이 갖추어야 하는 요구사항뿐만 아니라 전자적으로 구현하였기 때문에 갖추어야 하는 추가 요구사항을 가지고 있다. 따라서 전자서명의 요구사항은 다음과 같다.

- **요구사항 1.** 인증(authentic): 누가 서명하였는지 확인이 가능해야 한다.
- **요구사항 2.** 위조불가(unforgeable): 위조가 불가능해야 한다.
- **요구사항 3.** 재사용불가(not reusable): 서명을 다시 사용할 수 없어야 한다.
- **요구사항 4.** 변경불가(unalterable): 서명된 문서의 내용을 변경할 수 없어야 한다.
- **요구사항 5.** 부인방지(non-repudiation): 나중에 부인할 수 없어야 한다.

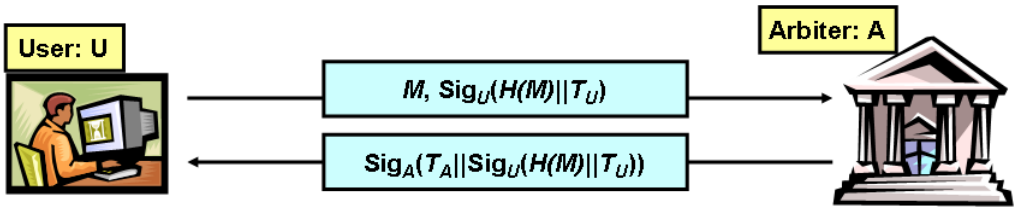
요구사항 1을 충족하기 위해서는 각 서명자마다 서명이 독특해야 한다. 요구사항 3에서 다시 사용할 수 없다는 것은 두 가지 측면으로 해석될 수 있다. 첫 번째 측면은 특정 메시지에 대한 서명 블록을 다시 사용할 수 없어야 한다는 것이다. 즉, 특정 메시지의 서명 블록을 다른 메시지의 서명 블록으로 사용할 수 없어야 한다. 여기서 서명 블록이란 메시지와 별도로 존재하는 서명값을 말한다. 보통 전자서명은 일반서명과 달리 서명한 문서 또는 메시지와 별도로 존재한다. 이 요구조건이 충족되기 위해서 전자서명은 메시지에 의존해야 한다. 두 번째 측면은 특정 메시지와 그것의 서명 블록 전체를 다시 사용할 수 없어야 한다는 것이다. 이 문제는 응용에 따라 차이가 있으며, 응용 차원에서 중복 검사를 통해 해결할 수 있다. 이것을 방어하기 위해 서명에 서명 시간을 포함하여 서명을 확인하는 사용자가 서명의 유효기간을 검사하는 방법을 일반적으로 많이 사용한다. 전체의 재사용이 가능하는 것은 기본적으로 전자서명이 디지털 데이터이기 때문이며, 디지털 데이터는 원본과 복사본을 구분할 수 없다. 하지만 전자서명에서는 원본과 복사본을 구분하는 것보다는 전자서명으로부터 서명자를 확인할 수 있고, 서명된 내용이 변경할 수 없다는 것이 더 중요한 요구사항이다. 요구사항 4는 요구사항 3의 첫 번째 측면과 동일한 의미이다. 따라서 전자서명이 메시지에 의존하면 요구사항 4를 충족시킬 수 있다.

전자서명은 다음과 같은 측면에서 일반 서명과 다르다. 첫째, 전자서명은 수학적으로 서명자를 검증할 수 있다. 하지만 일반 서명은 보통 원 서명과 눈으로 대조하여 검증하며, 각 서명자의 서명마다 위조의 어려움이 다를 수 있다. 따라서 일반 서명보다 전자서명이 상대적으로 안전하다고 할 수 있다. 둘째, 일반 서명은 문서 위에 하지만 전자서명은 앞서 언급한 바와 같이 문서와 보통 별도로 존재한다. 셋째, 일반 서명은 서명마다 동일한 형태이지만 전자서명은 메시지에 의존해야 하므로 문서마다 다른 형태이어야 한다. 넷째, 전자서명은 원본과 복사본을 구분하기가 어렵다. 따라서 전체 재사용을 방지하기 위해 서명시간을 서명에 포함해야 하며, 검증자는 이미 처리한 서명인지 중복검사를 해야 한다.

10.1.2 전자서명 방식의 분류

전자서명 방식은 서명할 때 중재자를 사용하느냐에 따라 **직접 서명**(direct digital signature) 방식과 **중재 서명**(arbitrated digital signature) 방식으로 구분할 수 있다. 직접 서명 방식은 서명자가 홀로 서명 알고리즘을 수행하여 서명하는 방식이다. 이 방식은 서명키가 분실 또는 도난 되었을 경우에 부인방지 요구사항을 충족시키는 것이 어려워진다는 문제점을 가지고 있다. 특히, 서명자가 직접 서명시간을 서명에 포함하고, 이 시간을 통해 서명이 이루어진 시점을 판단할 경우에는 서명이 원 서명자가 한 것인지 서명키를 획득한 공격자가 한 것인지 구분하기가 어렵다. 중재 서명 방식은 중재자와 프로토콜을 수행하여 서명하는 방식으로 앞서 언급한 직접 서명 방식의 문제점을 극복할 수 있다. 이 방식에서는 신뢰할 수 있는 중재자가 서명의 증인 역할을 수행하게 되며, 그림 10.1처럼 프로토콜이 수행된다. 즉, 서명자가 서명 알고리즘을 통해 서명을 한 후에 이것을 중재자에게 전달하면 중재자는 이 서명에서 다시 서명을 하는 방식을 취한다. 따라서 중재자의 서명시간이 서명에 포함되게 되며, 이 때문에 서명자가 서명 시간에 대한 부정을 할 수 없게 된다.

.



10.1.3 전자서명 알고리즘의 분류

전자서명을 생성하기 위해 사용되는 전자서명 알고리즘은 메시지 복구 여부에 따라 크게 **메시지 복구가능 전자서명 알고리즘**(DSS(Digital Signature Scheme) with recovery)과 **원 메시지 필요 전자서명 알고리즘**(DSS with appendix)로 분류된다. 전자는 서명값으로부터 메시지를 추출할 수 있기 때문에 서명값을 확인하기 위해 서명한 원 메시지가 필요 없다. 후자는 서명값으로부터 메시지를 추출할 수 없으므로 서명을 검증하기 위해서는 원 메시지가 필요하다. 매우 큰 메시지의 경우에는 전자 방식을 사용하기가 어렵다.

전자서명 알고리즘은 크게 결정 방식과 확률 방식으로 나누어질 수도 있다. 결정적 서명 알고리즘은 메시지와 서명키가 같으면 항상 같은 서명이 생성되는 알고리즘을 말하며, 확률적 서명 알고리즘은 메시지와 서명키가 같아도 매번 그 결과가 달라지는 알고리즘을 말한다. 여기서 메시지는 서명키 외에 타임스템프와 같이 전자서명할 때 포함되는 모든 정보를 말한다.

10.1.4 인증서의 필요성

전자서명은 보통 공개키 암호알고리즘을 사용하며, 이 때 개인키가 서명키가 되고, 공개키가 확인키가 된다. 공개키 암호알고리즘에서 가장 중요한 요소는 공개키의 인증이다. 따라서 전자서명에서도 공개키의 인증이 가장 중요하다. 특히, 확인키가 인증되지 않으면 주어진 서명이 누구의 서명인지 확신할 수 없으며, 이것을 속일 수 있으면 다른 사용자가 서

명키 없이도 자신의 서명을 다른 사용자의 서명으로 속일 수 있다. 공개키가 누구의 공개키인지 확인할 수 있도록 사용하는 가장 대표적인 메커니즘은 **인증서**(certificate)이다. 인증서는 신뢰할 수 있는 기관에서 발행한 전자 보증서이며, 공개키와 그것의 소유자를 바인딩하여 주는 역할을 한다. 인증서는 보통 **인증기관**(CA, Certification Authority)이라고 하는 신뢰할 수 있는 기관의 서명키로 공개키와 그것의 소유자를 서명하여 만들어진다.

10.1.5 전자서명에 대한 공격

전자서명에 대한 공격의 가장 큰 목표는 서명을 위조하는 것이다. 이 측면에서 위조에 대한 공격의 결과를 다음과 같이 분류한다. 이 분류는 대칭/비대칭 암호알고리즘에 대한 일반적인 공격 결과에 대한 분류와 유사하다.

- **완전 파괴**(total break): 공격자가 서명자의 서명키를 회득한 경우
- **완전 위조**(universal forgery): 서명자의 서명 알고리즘과 등가인 또 다른 효율적인 서명 알고리즘을 찾은 경우
- **선택 위조**(selective forgery): 특정한 메시지 또는 특정한 종류의 메시지에 대해서는 서명을 위조할 수 있는 경우
- **존재 위조**(existential forgery): 최소한 하나의 메시지에 대한 서명을 위조한 경우로서, 공격자가 위조 가능한 메시지를 선택할 수 없는 경우

전자서명에 대한 공격방법도 대칭/비대칭 암호알고리즘에 대한 일반적인 공격 방법과 유사하게 분류할 수 있다. 크게 분류하면 공격자가 확인키만 가지고 있는 경우와 확인키 뿐만 아니라 메시지와 서명 쌍을 가지고 있는 경우로 분류할 수 있다. 전자를 키단독 공격(key-only attack)이라 하고, 후자를 메시지 공격(message attack)이라 한다. 메시지 공격은 어떻게 메시지와 서명 쌍을 얻을 수 있는지에 따라 다음과 같이 세부적으로 분류할 수 있다.

- **알려진 메시지 공격**(known-message attack): 공격자가 메시지를 선택할 능력이 없는 경우
- **선택 메시지 공격**(chosen-message attack): 공격자가 메시지를 선택할 능력이 있지만 메시지에 대한 서명을 얻기 전에 서명 받을 모든 메시지를 선택해야 하는 경우
- **적응적 선택 메시지 공격**(adaptive chosen-message attack): 공격자는 메시지를 선택할 능력이 있을 뿐만 아니라 지난 요청의 결과에 따라 다음 요청을 선택할 수 있는 경우

위 내용에서 알 수 있듯이 실제 상황에서 가능한 공격 방법의 분류라기보다는 서명 알고리즘의 안전성을 측정하기 위한 공격 분류라는 것을 알 수 있다. 따라서 적응적 선택 메시지 공격을 하여도 안전한 서명 알고리즘이 가장 안전한 서명 알고리즘이 된다.

10.2 전자서명 알고리즘의 특성

10.2.1 전자서명의 구성요소

전자서명의 구성요소는 다음과 같다.

- M: 서명할 수 있는 메시지의 유한 집합
- M_S: 서명 알고리즘에 적용할 수 있는 값들의 집합
- S: 고정된 길이의 서명의 유한 집합
- $R: M \rightarrow M_R$: 일대일 함수로서 잉여함수(redundancy function)라 함 ($M_R \subseteq M_S$)
- $H: M \rightarrow M_H$: 충돌회피 해쉬함수 ($M_H \subseteq M_S$)
- R_I: 특정 서명함수를 식별하는 색인 요소들의 집합
- $S_{A,k}: M_S \rightarrow S$: 서명자 A의 서명함수($k \in R_I$)
- $V_A: M \times S \rightarrow \{true, false\}$: 서명자 A의 확인함수

이 구성요소를 통해 알 수 있는 사실은 다음과 같다. 메시지로부터 바로 서명값을 계산하는 것이 아니라 메시지를 서명 알고리즘에 적용할 수 있는 값들의 집합으로 매핑한 후에 계산된다. 이 때 이 매핑은 해쉬함수를 사용하거나 잉여함수를 이용한다. 집합 R_I의 크기가 1이면 이 서명 알고리즘은 결정 방식이고, 크기가 1보다 크면 확률 방식이다.

10.2.2 원 메시지 필요 전자서명 방식

원 메시지 필요 전자서명 방식에서는 해쉬함수를 사용한다. 즉, 메시지를 서명함수에 바로 적용하지 않고, 메시지의 해쉬값을 서명함수에 적용한다. 서명자는 서명을 하기 위해서는 먼저 서명키/확인키 쌍을 생성하여야 한다. 원 메시지 필요 전자서명 방식에서 서명키와 확인키는 각각 다음과 같이 정의된다.

- 개인키: 사용자 A의 개인키는 함수 집합 $S_A = \{S_{A,k} | k \in R_I\}$이다. 이 때 서명함수 $S_{A,k}$는 M_H에서 S로의 일대일 함수이다.
- 확인키: 사용자 A의 확인키는 확인 함수 $V_A(\widetilde{m}, s^*)$이며, 이 함수는 $S_{A,k}(\widetilde{m}) = s^*$이면 참이고 $S_{A,k}(\widetilde{m}) \neq s^*$이면 거짓이 된다. 이 때 $\widetilde{m} = H(m) \in M_H$이고 $s^* \in S$이다.

이 방식에서 서명자 A의 메시지 m에 대한 전자서명 과정은 다음과 같다.

- **단계 1.** $k \in R_I$를 선택한다. (함수집합 중에 하나의 서명함수를 선택한다.)
- **단계 2.** $\widetilde{m} = H(m) \in M_H$를 계산한다.
- **단계 3.** $s^* = S_{A,k}(\widetilde{m})$를 계산한다.

확인자는 메시지 m과 이것에 대한 A의 서명값 s^*를 받으면 다음과 같이 검증할 수 있다.

이 때 확인자는 A의 확인함수를 알고 있어야 한다.

- **단계 1.** $\widetilde{m}= H(m) \in M_H$를 계산한다.
- **단계 2.** $V_A(\widetilde{m}, s^*)$ 값이 참인지 검사한다.

이 방식에서 해쉬함수의 역할은 크게 두 가지이다. 하나는 긴 메시지 대신에 고정된 길이의 짧은 메시지에 서명할 수 있도록 하여 전자서명의 효율성을 높여준다. 다른 하나는 존재 위조가 어렵도록 해준다. 즉, $s^* \in S$를 임의로 선택한 다음에 그것에 해당하는 m을 찾는 것이 어려워진다. 이것이 어려운 이유는 해쉬함수의 일방향성 때문이다. 이 방식에서 해쉬함수는 그러나 충돌회피 해쉬함수이어야 한다. 즉. 일방향성만 만족하면 전자서명에 사용할 수 없다. 그 이유는 $s^* \in S$가 메시지 m에 대한 서명이고 $H(m) = H(m')$인 메시지 m'을 찾을 수 있으면 s^*는 m 뿐만 아니라 m'에 대한 유효한 서명이 되기 때문이다.

10.2.3 메시지 복구가능 서명 방식

메시지 복구 가능 전자서명 방식에서 서명키와 확인키는 각각 다음과 같이 정의된다.

- 개인키: 사용자 A의 개인키는 함수 집합 $S_A = \{S_{A,k} | k \in R_I\}$이다. 이 때 서명함수 $S_{A,k}$는 M_R에서 S로의 일대일 함수이다.
- 확인키: 사용자 A의 확인키는 확인 함수 $V_A(s^*) = \widetilde{m}$이다. 이 때 $\widetilde{m}= R(m) \in M_R$이고 $s^* \in S$이다.

즉, 이 방식에서는 해쉬함수 대신에 잉여함수라는 것을 사용하며, 이 함수는 해쉬함수와 달리 역함수가 존재한다. 뿐만 아니라 이 함수는 공개되어 있어, 누구나 $R(m)$으로부터 m을 계산할 수 있다. 이 방식에서 서명자 A의 메시지 m에 대한 전자서명 과정은 다음과 같다.

- **단계 1.** $k \in R_I$를 선택한다. (함수집합 중에 하나의 서명함수를 선택한다.)
- **단계 2.** $\widetilde{m}= R(m) \in M_R$를 계산한다.
- **단계 3.** $s^* = S_{A,k}(\widetilde{m})$를 계산한다.

확인자는 A의 서명값 s^*를 받으면 다음과 같이 검증할 수 있다. 이 때 확인자는 A의 확인함수를 알고 있어야 한다. 그러나 원 메시지 필요 전자서명 방식과 달리 서명 검증에 원 메시지가 필요 없다.

- **단계 1.** $\widetilde{m}= V_A(s^*)$를 계산한다.
- **단계 2.** $\widetilde{m}$가 M_R에 속하는지 검사한다.
- **단계 3.** $m = R^{-1}(\widetilde{m})$을 계산하여 메시지를 확인한다.

메시지 복구 전자서명 방식에서는 잉여함수가 매우 중요하다. 만약 $M_S = M_R$이 같으면 $M = M_R$이라는 것을 의미한다. 이 경우 임의의 $s \in S$를 선택하면 V_A를 이용하여 $\widetilde{m}$를 계산할 수 있고, $\widetilde{m}$로부터 다시 m을 계산할 수 있다. 즉, 존재 위조를 쉽게 할 수 있다. 그러므로 R을 잘 선택하여 M_R이 M_S의 작은 일부가 되도록 하여야 한다. 이 경우에는 임의의 $s \in S$를 선택하면 V_A를 이용하여 $\widetilde{m}$를 계산하더라도 $\widetilde{m} \in M_R$일 확률이 작다. 따라서 존재 위조를 하기가 어렵다.

10.3 RSA 전자서명

RSA 공개키 암호알고리즘에서 사용했던 공개키가 전자서명에서는 확인키가 되고, 개인키는 서명키가 된다. RSA 전자서명 방식은 메시지 복구가능 방식으로도 사용할 수 있고, 원 메시지 필요 방식으로도 사용할 수 있다. 메시지 복구가능 방식으로 사용할 경우 전자서명 과정은 다음과 같다.

- **단계 1.** $\widetilde{m} = R(m)$를 계산한다.
- **단계 2.** $s = \widetilde{m}^d \bmod n$를 계산한다.

검증과정은 다음과 같다.

- **단계 1.** $\widetilde{m} = s^e \bmod n$를 계산한다.
- **단계 2.** $\widetilde{m} \in M_R$인지 검사한다.
- **단계 3.** $m = R^{-1}(\widetilde{m})$을 계산하여 서명 메시지를 확인한다.

원 메시지 필요 방식의 경우에 전자서명 과정은 다음과 같다.

- **단계 1.** $\widetilde{m} = H(m)$를 계산한다.
- **단계 2.** $s = \widetilde{m}^d \bmod n$를 계산한다.

검증과정은 다음과 같다.

- **단계 1.** $\widetilde{m} = H(m)$를 계산한다.
- **단계 2.** $\widetilde{m} \stackrel{?}{\equiv} s^e \pmod n$이 성립하는지 확인한다.

RSA 공개키 암호알고리즘은 준동형 특성을 가지고 있으므로 RSA 전자서명도 이를 이용한 존재 위조가 가능하다. 따라서 메시지 복구가능 방식보다는 원 메시지 필요 방식을 사용하는 것이 더 바람직하다.

10.4 ElGamal 전자서명

ElGamal 전자서명은 ElGamal 공개키 암호알고리즘과 마찬가지로 이산대수 문제에 기반을 두고 있다. 따라서 이 서명 알고리즘의 시스템 설정은 ElGamal 공개키 암호알고리즘과 동일하게 $\mathbb{Z}_p^*$를 사용하거나 이 군의 위수가 소수 q인 부분군 선택한 다음의 그 군의 생성자 g를 선택해야 한다. 후자 방식으로 시스템 설정을 했다고 가정하면 추가적으로 충돌회피 해쉬함수 $H_q : \{0,1\}^* \rightarrow \mathbb{Z}_q$가 필요하다. 따라서 매우 큰 소수 p, $p-1$의 소인수 q, 생성자 g, 충돌회피 해쉬함수 $H_q : \{0,1\}^* \rightarrow \mathbb{Z}_q$를 이 시스템의 파라미터(parameter)라 한다. 서명자의 서명키/확인키 쌍은 ElGamal 공개키 암호알고리즘의 공개키 쌍과 동일하다. 즉, 서명자 A의 서명키는 $x \in \mathbb{Z}_q$가 되며, 그것의 대응되는 확인키 $y = g^x$가 된다. 이 때 메시지 m에 대한 서명 과정은 다음과 같다.

- **단계 1.** $w \in_R \mathbb{Z}_q$를 선택한다.
- **단계 2.** $W = g^w \bmod p$를 계산한다.
- **단계 3.** $s = (H_q(m) - xW)w^{-1} \bmod q$를 계산한다.

결과 서명은 쌍 $\langle W, s \rangle$가 된다. 이 서명은 서명 검증에 원 메시지가 필요한 서명이므로 확인자에게는 $\langle m, W, s \rangle$를 전달해야 한다. 이것을 받은 확인자는 서명자의 공개키 y를 이용하여 다음과 같이 확인한다.

- **단계 1.** $\widetilde{m} = H_q(m)$를 선택한다.
- **단계 2.** $g^{\widetilde{m}} ? \equiv y^W W^s \pmod p$를 검사한다. 두 값이 법 p에서 합동이면 이 서명은 메시지 m에 대한 A의 서명이 틀림이 없다.

단계 2에서 유효한 서명일 경우 합동식이 성립한다는 것은 다음 식을 통해 확인할 수 있다.

$$\begin{aligned}
g^{\widetilde{m}} \; ? &\equiv y^W W^s \\
g^{H_q(m)} ? &\equiv g^{xW} g^{ws} \\
g^{H_q(m)} ? &\equiv g^{xW} g^{w(H_q(m) - xW)w^{-1}} \\
g^{H_q(m)} ? &\equiv g^{xW} g^{H_q(m) - xW} \\
g^{H_q(m)} ? &\equiv g^{H_q(m)}
\end{aligned}$$

ElGamal 전자서명의 안전성은 이산대수 문제에 의존한다. 즉, 공격자가 서명자의 확인키 y로부터 서명키 x를 계산할 수 있으면 이 공격자는 서명자의 서명을 항상 위조할 수 있다. 하지만 확인키 y로부터 서명키 x를 계산하는 것은 이산대수 문제이므로 계산적으로 어렵다. 서명을 위조할 수 있는 다른 가능성을 생각해 보면 다음과 같다.

- $W \in_R G_q$를 선택한 다음에 $g^{H_q(m)} \equiv y^W W^s \pmod p$를 만족하는 s를 찾으면 메시지 m에 대한 서명을 위조할 수 있다. 하지만 이것은 $\log_W(g^{H_q(m)} y^{-W})$를 계산하는 것이므로 이 또한 이산대수 문제이다. 따라서 계산적으로 어렵다.

- $s \in_R \mathbb{Z}_q$를 선택한 다음에 $g^{H_q(m)} \equiv y^W W^s \pmod p$를 만족하는 W를 찾으면 메시지 m에 대한 서명을 위조할 수 있다. 하지만 W가 지수로도 사용되고, 군 원소로도 사용되었으므로 이산대수보다 더 어려운 문제이다.
- $W \in_R G_q$와 $s \in_R \mathbb{Z}_q$를 선택한 다음에 $g^{H_q(m)} \equiv y^W W^s \pmod p$를 만족하는 $H_q(m)$를 찾은 다음에 이 해쉬값으로 계산되는 m을 찾으면 메시지 m에 대한 서명을 위조할 수 있다. W와 s를 결정한 다음에 $H_q(m)$을 계산하는 문제는 이산대수 문제일 뿐만 아니라 이것을 해결하였다 하더라도 해쉬함수의 일방향성 때문에 이와 같은 공격은 가능하지 않다.

ElGamal 서명을 할 때 한 가지 주의할 점이 있다. 전자서명을 할 때 서명자는 단계 1에서 임의의 $w \in_R \mathbb{Z}_q$를 선택한다. 이 때문에 이 알고리즘은 확률 알고리즘이 된다. 즉, 이 값에 따라 메시지와 서명자가 동일하더라도 결과 서명값은 다르게 된다. 만약 서로 다른 메시지 m_1과 m_2에 대해 서명할 때 서명자가 동일한 $w \in_R \mathbb{Z}_q$를 사용하여 만든 서명이 각각 $\langle m_1, W, s_1 \rangle$과 $\langle m_2, W, s_2 \rangle$라 하자. 그러면 서명의 확인식에 의해 다음이 성립한다.

$$g^{H_q(m_1)} \equiv y^W W^{s_1} \pmod p$$
$$g^{H_q(m_2)} \equiv y^W W^{s_2} \pmod p$$

이 경우 공격자는 서명자의 개인키를 다음과 같이 확보할 수 있다. 위 식에 의해 다음이 성립하므로

$$g^{H_q(m_1) - H_q(m_2)} \equiv W^{s_1 - s_2} \equiv g^{w(s_1 - s_2)} \pmod p$$

$H_q(m_1) - H_q(m_2) = w(s_1 - s_2) \pmod q$가 성립한다. 따라서 공격자는 이 식으로부터 w를 계산할 수 있고, w를 알면 s_1 또는 s_2를 이용하여 서명자의 서명키 x를 계산할 수 있다. 따라서 ElGamal 전자서명에서는 항상 다른 w를 사용해야 한다.

ElGamal 전자서명의 경우에는 존재위조의 가능성도 존재한다. 다음 과정을 통해 공격자는 존재위조할 가능성도 존재한다.

- **단계 1.** $a, b \in_R \mathbb{Z}_q$를 선택하고 $W = g^a y^b$를 계산한다.
- **단계 2.** $A - as \equiv 0 \pmod q$, $W + bs \equiv 0 \pmod q$가 되도록 $s (= -Wb^{-1}) \in \mathbb{Z}_q$와 $A (= -Wab^{-1}) \in \mathbb{Z}_q$를 선택한다.
- **단계 3.** $H_q(m) = A$를 만족하는 m을 찾는다. m을 찾으면 $\langle m, W, s \rangle$는 유효한 서명이 된다.

위 공격의 결과가 유효한 서명이라는 것은 다음 식을 통해 확인할 수 있다.

$$\begin{aligned} g^A &\stackrel{?}{\equiv} y^W W^s \\ g^A &\stackrel{?}{\equiv} y^W (g^a y^b)^s \\ g^A &\stackrel{?}{\equiv} g^{as} y^{W+bs} \\ g^{A-as} &\stackrel{?}{\equiv} y^{W+bs} \\ g^0 &\stackrel{?}{\equiv} g^0 \end{aligned}$$

단계 3에서 $H_q(m) = A$를 만족하는 m을 찾는 것은 해쉬함수의 일방향성 때문에 계산적으로 어렵다. 따라서 이 공격이 ElGamal 안전성에 큰 영향을 주는 것은 아니다. 또한 이 공격은 명백하게 선택 위조 공격은 아니다. 즉, m을 선택한 다음에 $H_q(m) = A$로 두고, 공격을

시작할 수 없기 때문이다.

10.5 Schnorr 전자서명

Schnorr 전자서명은 ElGamal 전자서명과 마찬가지로 이산대수 문제에 기반을 두고 있는 서명 알고리즘이다. 따라서 ElGamal와 동일한 시스템 설정 과정이 필요하며, 서명키/확인키 쌍의 생성도 동일하다. $\mathbb{Z}_p^*$의 위수가 소수 q인 부분군 G_q, 이 군의 생성자 g, 충돌회피 해쉬함수 $H_q : \{0,1\}^* \rightarrow \mathbb{Z}_q$가 이 시스템의 시스템 파라미터라 하자. 또한 서명자 A의 서명키는 $x \in \mathbb{Z}_q$이며, 대응되는 확인키 $y = g^x$라 하자. 그러면 서명자 A의 메시지 m에 대한 전자서명 과정은 다음과 같다.

- **단계 1.** $w \in_R \mathbb{Z}_q$를 선택한다.
- **단계 2.** $c = H_q(m \| g^w)$를 계산한다.
- **단계 3.** $s = w - cx \bmod q$를 계산한다.

결과 서명은 쌍 $\langle c, s \rangle$가 되며, 이 서명 역시 서명 검증에 원 메시지가 필요한 서명이므로 확인자에게는 $\langle m, c, s \rangle$를 전달해야 한다. 이것을 받은 확인자는 서명자의 공개키 y를 이용하여 다음과 같이 확인한다.

- **단계 1.** $c ? \equiv H_q(m \| g^s y^c) \pmod q$를 검사한다. 두 값이 합동이면 이 서명은 메시지 m에 대한 A의 서명이 틀림이 없다.

유효한 서명이면 단계 1의 검사를 통과한다는 것은 다음 식을 통해 확인할 수 있다.

$$\begin{aligned} g^s y^c \quad ? &\equiv g^w \\ g^{w-cx} g^{xc} ? &\equiv g^w \end{aligned}$$

이 전자서명의 안전성도 ElGamal 전자서명의 안전성과 유사하게 분석할 수 있다. 즉, 공격자가 서명자의 확인키 y로부터 서명키 x를 계산할 수 있으면 이 공격자는 서명자의 서명을 항상 위조할 수 있다. 하지만 확인키 y로부터 서명키 x를 계산하는 것은 이산대수 문제이므로 계산적으로 어렵다. 서명을 위조할 수 있는 다른 가능성은 ElGamal와 유사하다.

- $w \in_R G_q$를 선택한 다음에 c를 계산하고 $g^s g^c \equiv g^w \pmod p$를 만족하는 s를 찾으면 메시지 m에 대한 서명을 위조할 수 있다. 하지만 이것은 $\log_g(g^w y^{-c})$를 계산하는 것이므로 이 또한 이산대수 문제이다. 따라서 계산적으로 어렵다.

ElGamal 서명을 할 때 한 가지 주의할 점은 전자서명할 때마다 다른 w를 사용해야 한단, 는 것이다. Schnorr 전자서명도 마찬가지로 매 번 다른 w를 사용해야 한다. 만약 같은 것을 사용하면 ElGamal 전자서명과 마찬가지로 공격자는 서명자의 서명키를 얻을 수 있다. 예를 들어 서로 다른 메시지 m_1과 m_2에 대해 서명할 때 서명자가 동일한 $w \in_R \mathbb{Z}_q$를 사용하여

만든 서명이 각각 $\langle m_1, c_1, s_1 \rangle$과 $\langle m_2, c_2, s_2 \rangle$라 하자. 그러면 서명의 확인식에 의해 다음이 성립한다.

$$g^w \equiv g^{s_1} y^{c_1} \pmod{p}$$
$$g^w \equiv g^{s_2} y^{c_2} \pmod{p}$$

따라서 다음이 성립하므로

$$g^{s_1 - s_2} \equiv g^{x(c_2 - c_1)} \pmod{p}$$

$x \equiv (s_1 - s_2)(c_2 - c_1)^{-1} \pmod{q}$이다. 따라서 공격자는 서명키 x를 계산할 수 있다.

10.6 DSA 전자서명

DSA(Digital Signature Algorithm)는 1991년 NIST에서 전사서명 표준(DSS, Digital Signature Standard)에 사용하기 위해 제안된 알고리즘이다. 이 알고리즘은 ElGamal 서명의 변형으로서 이산대수 문제에 기반을 두고 있다. 따라서 이 시스템의 설정은 ElGamal와 동일하다. 서명자 A의 서명키가 $x \in \mathbb{Z}_q$이고 대응되는 확인키가 $y = g^x$일 때 메시지 m에 대한 A의 서명 과정은 다음과 같다.

- **단계 1.** $w \in_R \mathbb{Z}_q$를 선택한다.
- **단계 2.** $W = (g^w \bmod p) \bmod q$를 계산한다.
- **단계 3.** $s = (H_q(m) + x W) w^{-1} \bmod q$를 계산한다.

결과 서명은 쌍 $\langle W, s \rangle$가 된다. 기존 ElGamal 전자서명과 비교하면 단계 2와 단계 3이 조금씩 바뀌었다. 단계 2에서는 군 연산을 수행한 후에 다시 q로 나머지 연산을 수행하였으며, 단계 3에서는 뺄셈 연산이 덧셈 연산으로 바뀌었다. 이 서명 역시 서명 검증에 원 메시지가 필요한 서명이므로 확인자에게는 $\langle m, W, s \rangle$를 전달해야 한다. 이것을 받은 확인자는 서명자의 공개키 y를 이용하여 다음과 같이 확인한다.

- **단계 1.** $e_1 = H_q(m) s^{-1} \bmod q$를 계산한다.
- **단계 2.** $e_2 = W s^{-1} \bmod q$를 계산한다.
- **단계 3.** $g^{e_1} y^{e_2} ? \equiv W \pmod{q}$를 검사한다. 두 값이 법 q에서 합동이면 이 서명은 메시지 m에 대한 A의 서명이 틀림이 없다.

단계 2에서 유효한 서명일 경우 합동식이 성립한다는 것은 다음 식을 통해 확인할 수 있다.

$$\begin{aligned}
g^{e_1} y^{e_2} &? \equiv W \\
g^{H_q(m) s^{-1}} y^{W s^{-1}} &? \equiv g^w \\
g^{H_q(m) s^{-1}} g^{x W s^{-1}} &? \equiv g^w \\
g^{(H_q(m) + x W) s^{-1}} &? \equiv g^w \\
g^{(H_q(m) + x W)(H_q(m) + x W)^{-1} w} &? \equiv g^w
\end{aligned}$$

10.7 ECDSA 전자서명

ECDSA(Elliptic Curve DSA)는 기존 DSA 표준을 타원곡선을 이용하여 재정의한 알고리즘으로서, 2000년 미국 표준 FIPS 186-2로 채택되었다. 타원곡선 기반 알고리즘이므로 시스템 설정을 위해서는 p가 소수인 유한체 F_p 상의 타원곡선 E를 하나 선택한 다음에 이 곡선 위에 위수가 매우 큰 소수 q인 점 P를 선택하고, 충돌회피 해쉬함수 $H_q: \{0,1\}^* \rightarrow \mathbb{Z}_q$를 선택하여 공개해야 한다. 서명자 A의 서명키가 $x \in \mathbb{Z}_q$이고 대응되는 확인키가 $Y = xP$일 때 메시지 m에 대한 A의 서명 과정은 다음과 같다.

- **단계 1.** $w \in_R \mathbb{Z}_q$를 선택한다.
- **단계 2.** $(u,v) = wP$를 계산한다.
- **단계 3.** $r = u \bmod q$를 계산한다.
- **단계 4.** $s = (H_q(m) + rx)w^{-1} \bmod q$를 계산한다.

결과 서명은 쌍 $\langle r,s \rangle$가 된다. 이 서명 역시 서명 검증에 원 메시지가 필요한 서명이므로 확인자에게는 $\langle m,r,s \rangle$를 전달해야 한다. 이것을 받은 확인자는 서명자의 공개키 Y를 이용하여 다음과 같이 확인한다.

- **단계 1.** $i = s^{-1} H_q(m) \bmod q$를 계산한다.
- **단계 2.** $j = s^{-1} r \bmod q$를 계산한다.
- **단계 3.** $(u,v) = iP + jY$를 계산한다.
- **단계 4.** $u \stackrel{?}{\equiv} r \pmod q$를 검사한다. 두 값이 법 q에서 합동이면 이 서명은 메시지 m에 대한 A의 서명이 틀림이 없다.

단계 4에서 유효한 서명일 경우 합동식이 성립한다는 것은 다음 식을 통해 확인할 수 있다.

$$\begin{aligned}
(u,v) &\stackrel{?}{\equiv} iP + jY \\
wP &\stackrel{?}{\equiv} iP + jxP \\
wP &\stackrel{?}{\equiv} s^{-1}H_q(m)P + s^{-1}rxP \\
wP &\stackrel{?}{\equiv} s^{-1}(H_q(m)P + rx)P \\
wP &\stackrel{?}{\equiv} (H_q(m) + rx)^{-1} w (H_q(m) + rx)P \\
wP &\stackrel{?}{\equiv} wP
\end{aligned}$$

10.8 겹선형 쌍함수를 이용한 전자서명

겹선형 쌍함수를 이용하여 매우 간단한 전자서명 알고리즘을 만들 수 있다. 다음 알고리즘은 Boneh, Lynn, Shacham이 제안한 전자서명 알고리즘이다. 겹선형 쌍함수를 사용하기 위해서는 타원곡선 위의 덧셈군과 유한체 위의 곱셈군 두 가지 군이 필요하다. 따라서 시스템 설정은 다음과 같이 이루어진다. 위수가 소수 q인 타원곡선을 이용한 순환군 $G_1 = \langle P \rangle$과 위수가 소수 q인 유한체 상의 순환군 G_2를 생성한 다음에 겹선형 쌍함수 $\hat{e}: G_1 \times G_1 \rightarrow G_2$와 충돌회피 해쉬함수 $H_1: \{0,1\}^* \rightarrow G_1$를 선택하여 공개해야 한다. 서명자 A

의 서명키가 $x \in \mathbb{Z}_q$이고 대응되는 확인키가 $Y = xP$일 때 메시지 m에 대한 A의 서명 과정은 다음과 같다.

- **단계 1.** $s = xH_1(m)$를 계산한다.

결과 서명은 s가 되며, 이 서명 역시 서명 검증에 원 메시지가 필요한 서명이므로 확인자에게는 $\langle m, s \rangle$를 전달해야 한다. 이것을 받은 확인자는 서명자의 공개키 Y를 이용하여 다음과 같이 확인한다.

- **단계 1.** $\hat{e}(P,s) ? \equiv \hat{e}(Y, H_1(m))$를 검사한다. 두 값이 일치하면 이 서명은 메시지 m에 대한 A의 서명이 틀림이 없다.

유효한 서명일 경우 단계 1의 두 값은 다음 식을 통해 일치한다는 것을 확인할 수 있다.

$$\begin{aligned} \hat{e}(P,s) \quad &? \equiv \quad \hat{e}(y, H_1(m)) \\ \hat{e}(P, xH_1(m)) &? \equiv \hat{e}(xP, H_1(m)) \\ \hat{e}(P, H_1(m))^x &? \equiv \hat{e}(P, H_1(m))^x \end{aligned}$$

10.9 신원기반 전자서명 알고리즘

신원기반 공개키 암호시스템은 공개키 생성 방식에 대한 기존 공개키 암호시스템의 대안으로 등장한 시스템이다. 따라서 신원기반 전자서명 알고리즘은 신원기반 암호시스템이 등장한 이후 많은 알고리즘이 제안되고 있다. 9장에서 언급한 바와 같이 겹선형 쌍함수를 기반한 신원기반 시스템이 제안된 이후 신원기반 시스템에 대한 연구가 다시 활발하게 이루어지고 있다. 이 절에서 소개하는 알고리즘은 Hess가 2003년도에 제안한 알고리즘으로서, 겹선형 쌍함수를 기반한 신원기반 시스템이므로 PKG가 서명키를 사용자에게 생성해 주어야 한다. 따라서 이 알고리즘의 시스템 설정은 다음과 같다. PKG는 위수가 소수 q인 타원곡선을 이용한 순환군 $G_1 = \langle P \rangle$과 위수가 소수 q인 유한체 상의 순환군 G_2를 생성한 다음에 겹선형 쌍함수 $\hat{e}: G_1 \times G_1 \rightarrow G_2$, 충돌회피 해쉬함수 $H_1 : \{0,1\}^* \rightarrow G_1$과 $H_2 : \{0,1\}^* \times G_2 \rightarrow \mathbb{Z}_q^*$를 선택하여 공개한다. 끝으로 PKG는 마스터 키 $s \in \mathbb{Z}_q$를 선택한 다음 공개키 $P_{pub} = sP$를 계산하여 공개키는 시스템 파라미터와 함께 공개한다. PKG는 서명자 A의 신원정보가 ID_A일 때 이 사용자의 공개키 $Q_{ID} = H_1(ID_A)$를 이용하여 서명키 $d_{ID} = sQ_{ID}$를 계산하여 A에게 전달한다. 메시지 m에 대한 A의 서명 과정은 다음과 같다.

- **단계 1.** $w \in_R \mathbb{Z}_q$와 $P_1 \in_R G_1$를 선택한다.
- **단계 2.** $r = \hat{e}(P_1, P)^w$를 계산한다.
- **단계 3.** $v = H_2(m, r)$를 계산한다.
- **단계 4.** $u = vd_{ID} + wP_1$를 계산한다.

결과 서명은 $\langle u, v \rangle$가 되며, 이 서명 역시 서명 검증에 원 메시지가 필요한 서명이므로 확

인자에게는 $\langle m, u, v \rangle$를 전달해야 한다. 이것을 받은 확인자는 서명자의 공개키 Q_{ID}를 이용하여 다음과 같이 확인한다.

- **단계 1.** $r = \hat{e}(u, P)\hat{e}(Q_{ID}, -P_{pub})^v$를 계산한다.
- **단계 2.** $v? \equiv H_2(m, r)$를 검사한다. 두 값이 일치하면 이 서명은 메시지 m에 대한 A의 서명이 틀림이 없다.

유효한 서명일 경우 단계 2의 두 값은 다음 식을 통해 일치한다는 것을 확인할 수 있다.

$$\begin{aligned}
\hat{e}(u, P)\hat{e}(Q_{ID}, -P_{pub})^v ? &\equiv \hat{e}(vd_{ID} + wP_1, P)\hat{e}(Q_{ID}, -sP)^v \\
&? \equiv \hat{e}(vsQ_{ID}, P)\hat{e}(wP_1, P)\hat{e}(Q_{ID}, P)^{-vs} \\
&? \equiv \hat{e}(Q_{ID}, P)^{vs}\hat{e}(wP_1, P)\hat{e}(Q_{ID}, P)^{-vs} \\
&? \equiv \hat{e}(wP_1, P)
\end{aligned}$$

참고문헌

[1] S.G. Akl, "Digital Signatures: A Tutorial Survey", IEEE Computer, Vol. 16, No. 2, pp. 15-24, 1983.

[2] C.P. Schnorr, "Efficient Signature Generation by Smart Cards", J. of Cryptology, Vol. 4, pp. 161-174, 1991.

[3] F. Hess, "Efficient Identity based Signature Scheme based on Parings", Proc. of the 9th Annual Int. Workshop on Selected Areas in Cryptography, SAC 2002, LNCS 2595, Springer, pp. 310-324, 2003.

연습문제

1. 전자서명과 MAC은 모두 메시지 작성자를 인증할 수 있는 기능을 제공할 수 있다. 그 차이점에 대해 설명하시오.
2. 전자서명 중 원 메시지 필요 전자서명에서 잉여함수의 역할을 설명하시오.