제 9 장 공개키 암호알고리즘

9.1 소수 생성

대부분의 공개키 암호알고리즘에서는 공개키 쌍을 생성하거나 유한순환군을 생성하기 위해 매우 큰 소수를 필요로 한다. 하지만 소수를 생성하여 주는 결정적 알고리즘은 존재하지 않는다. 따라서 임의의 수를 선택한 후에 그것이 소수인지 판단하는 방식을 사용한다. 소수인지 정확하게 판단하기 위해서는 인수분해를 해야 하지만 인수분해하는 것은 계산적으로 어렵기 때문에 확률적인 판단 방법을 사용하는 것이다. 즉, 몇 가지 조건을 검사하여 통과하면 소수라고 결론을 내리고 사용하는 것이다. 어떤 수가 소수인지 판단하기 위해 사용할 수 있는 몇 가지 판단 방법은 다음과 같다.

- **조건 1.** n보다 작은 임의의 정수 a에 $\gcd(a,n)\neq 1$이면 n은 소수가 아니다.
- **조건 2,** n보다 작은 임의의 정수 a에 대해 $a^{n-1}\not\equiv 1 \pmod n$이면 n은 소수가 아니다.
- **조건 3.** $x = a^{(n-1)/2}$이면 $x^2 = a^{n-1} \equiv 1 \pmod n$이므로 x가 1 또는 -1이 아니면 n은 소수가 아니다.

조건 2는 페르마의 작은 정리를 이용한 것이다. 조건 3은 제곱근 문제에서 p가 소수일 때 $x^2 \equiv a \pmod p$의 해가 존재하면 정확하게 두 개의 해를 가진다는 원리를 이용하고 있다. 따라서 $x^2 \equiv 1 \pmod p$는 해가 항상 존재하며, 그 해는 1과 -1이다.

주어진 정수가 소수인지 판별하는 알고리즘은 오일러 규준(Euler Criterion)을 이용하는 Soloway와 Strassen의 방법과 위에 제시된 조건 3을 응용한 Rabin과 Miller 방법 등이 있다. 임의로 선택한 수가 합성수일 때, Soloway와 Strassen 시험을 한 번 통과할 확률은 50%이며, Rabin과 Miller 방법은 25%이다. 따라서 Rabin과 Miller 방법이 더 우수하다.

오일러 규준이란 p가 홀수인 소수이면 모든 a에 대해 $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p$가 성립한다는 것이며, 여기서 $\left(\frac{a}{p}\right)$는 르장드로 기호이다. 이 규준은 소수가 법일 때 주어진 수가 이차잉여인지 아닌지 판단할 때 사용된다. 좀 더 자세히 설명하면 위에 제시된 조건 3에서 설명한 바와 같이 p가 소수이면 $a^{(p-1)/2} \bmod p$는 1 아니면 -1이다. a가 이차잉여이면 제곱근 b을 가지며, a 자리에 b^2를 대입하면 $a^{(p-1)/2} \bmod p$는 반드시 1이어야 한다는 것을 알 수 있다. Soloway와 Strassen은 임의의 a에 대해 오일러 규준이 만족되지 않으면 주어진 수가 소수가 아닌 것으로 판정하는 방법을 사용한다.

Rabin과 Miller는 $n-1 = 2^k m$을 만족하는 홀수 m을 구한 다음 임의의 a에 대해 $a^m \bmod n$, $a^{2m} \bmod n$, ..., $a^{2^{k-1}m} \bmod n$을 계산하여 이 값들을 통해 n이 소수인지 판별한다. Rabin과 Miller는 $a^m \equiv \pm 1 \pmod n$이거나 $a^{2m} \bmod n$, ..., $a^{2^{k-1}m} \bmod n$ 중 하나가 -1과 합동인지 검사하여 만족하면 n이 소수일 확률이 높다고 판단한다. 이 조건을 만족하면 $a^{n-1} \equiv 1 \pmod n$이 만족된다는 것을 알 수 있다. 특히 n이 소수이면 이 조건을 무조건 만족해야 한다. 그 이유는 n 이하의 수의 위수는 $n-1$의 약수가 되기 때문이다. 거꾸로 n이

합성수이어도 이 검사에 통과할 수 있는데 그것의 확률은 25%이하라고 증명이 되어 있다. 따라서 여러 수를 이용하여 검사하여도 통과하면 n이 소수일 확률이 매우 높다.

실제 응용에서 소수는 다음과 같이 생성된다.

- **단계 1.** n 비트 수 p를 임의로 생성한다.
- **단계 2.** 최상위 비트와 최하위 비트를 1로 설정한다.
- **단계 3.** 3, 5, 7, 11과 같은 작은 소수로 p를 나누어 본다.
- **단계 4.** Rabin-Miller 시험을 5번 수행한다.

단계 2에서 최상위 비트를 1로 설정하는 것은 필요한 크기의 소수를 선택하기 위함이고, 최하위 비트를 1로 설정하는 것은 홀수를 선택하기 위함이다. 선택한 p가 합성수이면 3, 5, 7로 나누어만 보아도 50% 이상은 이 단계를 통과하지 못한다. p가 합성수인데 5번 Rabin과 Miller 시험을 통과할 확률은 0.1% 이하가 된다.

9.2 RSA 공개키 암호알고리즘

9.2.1 공개키 쌍 생성

RSA 공개키 암호알고리즘은 MIT의 Rivest, Shamir, Adleman이 제안한 알고리즘으로서, 현재 가장 널리 사용되고 있다. RSA는 RSA 문제의 어려움에 기반을 두고 있다. 이 알고리즘은 다음과 같은 과정을 통해 공개키 쌍을 생성한다.

- **단계 1.** 두 개의 소수 p와 q를 선택한다.
- **단계 2.** $n = pq$를 계산한다.
- **단계 3.** $\phi(n) = (p-1)(q-1)$을 계산한다.
- **단계 4.** $\gcd(e, \phi(n)) = 1$인 e를 선택한다.
- **단계 5.** $ed \equiv 1 \pmod{\phi(n)}$를 만족하는 d를 계산한다.

위 과정을 통해 생성된 $\langle n,e \rangle$ 쌍이 공개키가 되고, $\langle n,d \rangle$ 쌍이 개인키가 된다. 단계 1에서 두 개의 소수를 선택해야 하는데, 이 소수는 앞 절에서 언급한 방법을 사용하여 선택해야 한다. 또 n의 크기를 고려하여 p와 q의 크기를 결정하는데, 이 때 p와 q의 크기는 보통 같도록 만든다. 단계 3를 계산한 후에는 더 이상 p와 q를 유지할 필요가 없다. 안전성을 위해서는 이들을 폐기하는 것이 바람직하다. 단계 5는 확장 유크리드 알고리즘을 사용하여 계산된다.

RSA 공개키 쌍을 생성할 때에는 두 개의 소수가 필요하다. 또한 두 사용자가 동일한 두 소수를 사용하게 되면 한 사용자가 다른 사용자의 개인키를 계산할 수 있게 된다. 따라서 사용자마다 다른 쌍의 소수가 필요하다. 이 때문에 정수론 지식이 부족한 경우에는 다음 사항들에 대해 궁금해 하는 경우가 있다.

- **궁금증 1.** 모든 사람이 다른 소수를 필요로 한다. 소수가 부족하지 않을까?

- **궁금증 2.** 두 사람이 우연히 같은 소수를 선택하는 경우도 있지 않을까?
- **궁금증 3.** 모든 소수에 대한 데이터베이스를 구축하면 공개키 암호알고리즘을 해독할 수 있지 않을까

위 모두에 대해 걱정할 필요가 없다. 그 이유는 소수는 무수히 많기 때문이다. 512 비트 이하의 수만 고려하면 대략 10^{151}개의 소수가 존재한다. 따라서 소수가 부족하지도 않으며, 같은 소수를 선택할 확률도 낮고, 이와 같은 데이터베이스를 구축하는 것도 현실적으로 불가능하다.

9.2.2 암호화와 복호화 알고리즘

RSA 공개키 암호알고리즘에서 개인키가 (n,d)이고 공개키가 (n,e)일 때, $M<n$의 암호화는 다음과 같이 이루어진다.

$$C = M^e \bmod n$$

즉, 주어진 평문 M을 공개키를 이용하여 거듭제곱하여 암호화한다. 복호화는 암호문을 개인키로 다음과 같이 거듭제곱하여 이루어진다.

$$M = C^d \bmod n$$

그런데 어떤 이유에서 이와 같은 과정을 통해 복호화가 가능할까? 이것은 다음과 같이 증명할 수 있다.

- $\gcd(M,n)=1$인 경우: 다음과 같이 오일러 정리에 의해 성립함을 알 수 있다.
$$C^d \equiv (M^e)^d \equiv M^{ed} \equiv M^{k\phi(n)+1} \equiv (M^{\phi(n)})^k M \equiv M \pmod n$$
- $\gcd(M,n)\neq 1$인 경우: $M<n$이므로 M은 p의 배수이거나 q의 배수이다. p의 배수라고 가정하면 $C^d \equiv M^{ed} \equiv M \pmod p$가 성립한다. 또 $M<n$이므로 M이 p의 배수이면 q의 배수일 수 없다. 따라서 $C^d \equiv (M^{\phi(n)})^k M \equiv (M^{\phi(q)})^{\phi(p)k} M \equiv M \pmod q$가 성립한다. 그러므로 $C^d \equiv M \pmod n$이 성립한다. M이 q의 배수일 경우에도 동일하게 증명할 수 있다.

따라서 $M<n$이면 RSA의 복호화하는 항상 올바르게 동작한다.

RSA는 원래 $\phi(n)$을 사용하도록 제안되었지만 $\phi(n)$ 대신에 보편지수(universal exponent) λ를 사용할 수도 있다. 보편지수란 모든 $a \in \mathbb{Z}_n^*$에 대하여 $a^\lambda \equiv 1 \pmod n$를 만족하는 가장 작은 정수를 말한다. 이 λ는 모든 a의 위수의 최소공배수이며 $\lambda | \phi(n)$이다. 특히, $n=pq$이면 $\lambda = \mathrm{lcm}(p-1, q-1)$이다. 이 사실들로부터 알 수 있듯이 $\lambda < \phi(n)$보다 작다. 따라서 $\phi(n)$ 대신에 λ를 사용하면 선택하는 공개키 e와 d 값이 작아진다. 그러므로 λ를 사용하면 암호화와 복호화의 성능을 향상시킬 수 있으며, 안전성에는 차이가 없다. 그 이유는 λ를 계산하기 위해서는 n을 인수분해할 수 있어야 한다.

9.2.3 안전성 분석

대칭키 암호알고리즘의 경우에는 안전성을 논할 때 가장 먼저 고려하는 것이 가능한 모든

키로 검사하는 전사공격이다. 이 공격에 의해 대칭키의 길이가 결정된다. 하지만 공개키 암호알고리즘에서는 대칭키와 달리 사용하는 키의 길이가 매우 크기 때문에 가능한 모든 키를 검사하는 것은 계산적으로 가능하지 않을 뿐만 아니라 가능한 모든 키를 검사하는 것보다 공개키 암호알고리즘이 의존하는 계산 문제를 푸는 것이 더 빠르다. 따라서 공개키 암호알고리즘에서는 전사공격은 의미가 없다.

RSA 공개키 암호알고리즘은 RSA 문제에 의존하며, 이 문제는 인수분해 문제에 의존한다. 즉, 공개키에 포함된 n을 인수분해할 수 있으면 개인키를 구할 수 있다. 미래를 고려하였을 때 n은 1024비트 이상이 되어야 한다.

RSA는 준동형(homomorphic) 특성을 가지고 있다. 어떤 함수 f가 항상 $f(xy)=f(x)f(y)$ 만족하면 준동형 특성을 가지고 있다고 말한다. RSA의 경우에는 다음이 성립하므로 이 특성을 가지고 있다고 말한다.

$$(M_1M_2)^e \equiv M_1^e M_2^e \equiv C_1C_2 \pmod n$$

즉, 두 개의 값을 곱하여 암호화한 결과는 각 값을 개별적으로 암호화한 값을 곱한 것과 같아진다. 즉, 2와 5를 각각 암호화한 암호문을 가지고 있으면 이것을 이용하여 10을 암호화한 암호문을 만들 수 있다. 이 특성을 이용하여 다음과 같은 공격이 가능하다.

- 공격자의 목표: 암호문 C를 복호화하는 것
- 가정. 이것을 복호화할 수 있는 사용자에게 C가 아닌 다른 암호문의 복호화는 요청할 수 있다.
- 공격방법
  - 공격자는 $r \in_R \mathbb{Z}_n$를 선택하고 $C' = r^e C \bmod n$을 계산하여 이 값에 대한 복호화를 요청하여 $M' \equiv C'^d \pmod n$을 받는다.
  - 공격자는 $t \equiv r^{-1} \pmod n$을 구한 후에 다음을 계산하면 C를 복호화할 수 있다.

$$tM' \equiv r^{-1}C'^d \equiv r^{-1}(r^eC)^d \equiv r^{-1}r^{ed}C^d \equiv r^{-1}rM \equiv M \pmod n$$

$f(x)=x$가 되는 x를 우리는 보통 고정점(fixed-point)이라 한다. RSA에는 고정점이 존재한다. 즉, $M^e \equiv M \pmod n$인 M이 존재한다. 대표적으로 0, 1, $n-1$은 모두 고정점이다. M이 고정점이 되기 위해서는 $M^{e-1} \equiv 1 \pmod n$이어야 하며, 최소 9개는 항상 존재한다. 그러나 p, q, e를 임의로 잘 선택하면 고정점의 수는 무시할 정도로 적어 실제 응용에서는 크게 고려하지 않고 RSA를 사용할 수 있다.

법 $\phi(n)$에서 e의 위수가 k이면 $M^{e^k} \equiv M \pmod n$이 성립하며, 이 때 k를 M의 주기라 한다. 암호문 C의 주기가 짧으면 공격자는 개인키를 모르는 상태에서 C를 쉽게 복호화할 수 있다. 예를 들어 C의 주기가 k이면 $C^{e^k} \equiv C \pmod n$이므로 $M \equiv C^{e^{k-1}} \pmod n$이 된다. 즉, 공격자는 C를 그 결과가 C가 될 때까지 e로 반복적으로 거듭제곱하면 C를 복호화할 수 있다. 따라서 주기가 작으면 위험하다.

끝으로 RSA의 경우에는 두 사용자가 절대 같은 n을 사용하지 않아야 한다. 만약 두 사용자 Alice와 Bob이 같은 n을 사용한다고 가정하자. 이 때 두 사용자가 직접 공개키를 생성하였다면 둘 다 n의 인수분해를 알고 있음을 의미한다. 따라서 서로 상대방의 개인키를 계산할 수 있다. 이 문제뿐만 아니라 공격자가 두 사용자가 같은 n을 사용한다는 것을 알게 되면 이 공격자는 개인키를 모르는 상태에서 메시지를 복호화할 수 있는 가능성이 있다. 예

를 들어 Alice의 공개키가 (n,e_1)이고, Bob은 (n,e_2)이며, $\gcd(e_1,e_2)=1$이라 하자. 이 때 공격자가 메시지의 내용은 모르지만 다음과 같은 동일한 메시지를 Alice와 Bob이 각각 암호화한 값을 확보하였다고 하자.

$$C_1 \equiv M^{e_1} \pmod{n}$$

$$C_2 \equiv M^{e_2} \pmod{n}$$

그러면 공격자는 Alice나 Bob의 개인키를 몰라도 위 암호문을 복호화할 수 있다. 공격자가 위 암호문을 복호화하는 과정은 다음과 같다.

단계 1. 확장 유크리드 알고리즘을 사용하여 $re_1+se_2=1$인 r과 s를 찾는다.
단계 2. r과 s 중에 r이 음수라 하면 다음과 같이 복호화할 수 있다.

$$(C_1^{-1})^{-r} C_2^s \equiv M^{re_1} M^{se_2} \equiv M \pmod{n}$$

따라서 n은 절대 사용자 간에 공유하면 안 된다.

9.3 이산대수 기반 공개키 암호알고리즘

9.3.1 유한순환군과 생성자

이산대수 문제에 기반을 두고 있는 알고리즘은 보통 군 연산을 수행할 유한순환군이 필요하다. 이 순환군을 생성하는 과정을 시스템 설정이라 한다. 암호기술에서 유한순환군이 필요할 경우 가장 많이 사용하는 것이 p가 소수일 때 $\mathbb{Z}_p^*$ 군 또는 위수가 소수인 $\mathbb{Z}_p^*$의 부분군이다. 유한순환군을 선택한 다음에는 이 군의 생성자를 선택해야 시스템 설정이 완료된다. $\mathbb{Z}_p^*$ 군을 생성하기 위해서는 적당한 크기의 소수 하나만 선택하면 된다. 하지만 이것의 부분군을 사용하기 위해서는 $p-1$의 약수인 소수 q를 추가적으로 선택해야 한다. p를 선택한 후에 q를 선택하기 위해서는 $p-1$를 인수분해해야 한다. 하지만 매우 큰 수를 인수분해하는 것은 계산적으로 쉬운 것이 아니므로 거꾸로 생성하는 방식을 사용한다. 즉, q를 먼저 선택한 후에 이 수로부터 p를 만들고 p가 소수인지 검사한다. 만약 이렇게 하여 생성한 p가 소수가 아니면 이 과정을 다시 반복해야 하므로 그렇게 효율적인 방법은 아니다. 하지만 시스템 설정은 최초에 한 번 설정된 이후에는 반복적으로 해야 하는 것은 아니므로 큰 문제가 되지는 않는다. 또 이와 같은 방법으로 군을 생성하여야 군의 생성자를 선택하기 쉽다.

군의 생성자는 다음 원리를 이용하여 선택한다. 유한순환군의 모든 부분군은 순환군이며, 부분군의 위수는 원 군의 약수이다. 따라서 원 군의 모든 원소는 이들 부분군 중 하나의 군의 생성자가 된다. 따라서 어떤 원소가 어느 군의 생성자인지 확인하기 위해서는 가장 작은 약수부터 사용하여 거듭제곱한 값이 1인 되는지 확인하면 된다. 하지만 보통 우리가 찾고자 하는 생성자는 원 군의 생성자이거나 이 군의 부분군 중 위수가 소수인 특정 군이다. 군의 위수가 n이고 이것의 소인수분해가 $p_1^{e_1}p_2^{e_2}\cdots p_r^{e_r}$일 때 원소 g가 원 군의 생성자인지 여부는 g^{n/p_i}가 1인지 검토하여 확인할 수 있다. g^{n/p_i} 값이 모든 n의 소인수에 대해 1이 아니면 이 값은 원 군의 생성자가 된다. $g^{n/p_i} \neq 1$라고 하는 것은 $i \neq j$에 대해 g^{p_j}가 1이 아니라는 것

을 말한다. 따라서 g^{n/p_i} 값이 모두 1이 아니면 오직 g^n만 1이 된다는 것을 의미함으로 g는 원 군의 생성자가 된다. 군의 위수가 n이고, q가 n의 소인수를 일 때 위수가 q인 부분군의 생성자를 선택하기 위해서는 임의의 g를 선택한 후에 $g^{n/q}$를 계산한다. 이 때 $g^{n/q} \neq 1$이면 $g^{n/q}$은 위수는 q가 된다. 그 이유는 $g^{n/q} \neq 1$이지만 $(g^{n/q})^q = 1$이기 때문이다.

9.3.2 ElGamal 공개키 암호알고리즘

RSA 공개키 암호알고리즘은 인수분해 문제에 기반을 두고 있는 가장 대표적인 알고리즘이라면 ElGamal는 이산대수 문제에 기반을 두고 있는 가장 대표적인 알고리즘이다. 시스템 설정이 되어 유한순환군과 군의 생성자 g가 주어지면 각 사용자는 자신의 공개키 쌍을 생성할 수 있다. 예를 들어 $\mathbb{Z}_p^* = \langle g \rangle$를 사용하기로 설정되었으면 사용자는 자신의 개인키가 될 임의의 $0 < x \leq p-1$를 선택한 다음에 자신의 공개키 $y = g^x$를 계산한다. 이 사용자에게 메시지 $M < p$은 다음과 같이 암호화하여 전달한다.

- **단계 1.** $r \in_R \mathbb{Z}_q$를 선택한 다음에 $A = g^r$를 계산한다.
- **단계 2.** y^r를 계산한 후에 $B = y^r \bullet M$을 계산한다.

쌍 $\langle A, B \rangle$가 결과 암호문이 된다. 이것을 수신한 사용자는 다음과 같이 복호화한다.

- **단계 1.** $R = A^x$을 계산한다. 그 다음 법 p에서 R의 곱셈에 대한 역원 R'을 계산한다.
- **단계 2.** $B \bullet R'$ 계산하면 평문 M을 얻을 수 있다.

단계 2를 통해 복호화가 가능한 이유는 다음 식을 통해 확인할 수 있다.

$$B \bullet R' = y^r \bullet M \bullet A^{-x} = g^{rx} \bullet M \bullet g^{-rx} = M$$

ElGamal 공개키 암호알고리즘은 RSA와 마찬가지로 준동형 특성을 가지고 있다. 예를 들어 어떤 특정 사용자의 공개키를 이용한 메시지 M_1에 대한 암호문이 $\langle A_1, B_1 \rangle = \langle g^{r_1}, y^{r_1} M_1 \rangle$이고, 메시지 M_2에 대한 암호문이 $\langle A_2, B_2 \rangle = \langle g^{r_2}, y^{r_2} M_2 \rangle$이라 하자. 그러면 두 암호문을 곱한 결과는 다음과 같이 두 평문 메시지를 곱한 결과를 같은 사용자가 암호화한 결과와 같아진다.

$$\langle A, B \rangle = \langle A_1 A_2, B_1 B_2 \rangle = \langle g^{r_1 + r_2}, y^{r_1 + r_2} M_1 M_2 \rangle$$

즉, ElGamal는 $E(M_1 \times M_2) = E(M_1) \times E(M_2)$를 만족한다. ElGamal은 이와 같은 특성뿐만 아니라 $E(M_1 + M_2) = E(M_1) \times E(M_2)$를 만족하도록 구성할 수도 있다. 이 경우 M을 기존대로 암호화하지 않고 M 대신에 g^M을 암호화해야 한다. 물론 이 경우 g^M으로부터 M을 얻을 수는 없지만 이 특성이 전자선거와 같은 다른 응용에 유용하게 사용되고 있다. 좀 더 구체적으로 살펴보자. 어떤 특정 사용자의 공개키를 이용한 메시지 M_1에 대한 암호문이 $\langle A_1, B_1 \rangle = \langle g^{r_1}, y^{r_1} g^{M_1} \rangle$이고, 메시지 M_2에 대한 암호문이 $\langle A_2, B_2 \rangle = \langle g^{r_2}, y^{r_2} g^{M_2} \rangle$이라 하자. 그러면 두 암호문을 곱한 결과는 다음과 같이 두 평문 메시지를 곱한 결과를 같은 사용

자의 공개키로 암호화한 결과와 같아진다.

$$\langle A, B\rangle = \langle A_1 A_2, B_1 B_2\rangle = \langle g^{r_1+r_2}, y^{r_1+r_2} g^{M_1+M_2}\rangle$$

9.4 타원곡선을 이용한 ElGamal 공개키 암호알고리즘

유한순환 곱셈군에서 이산대수 문제에 기반을 두고 있는 암호알고리즘들은 쉽게 타원곡선 기반 덧셈군을 사용하는 타원곡선 이산대수 문제에 기반을 두는 암호알고리즘으로 변형할 수 있다. 타원곡선을 이용한 ElGamal 공개키 암호알고리즘의 시스템 설정은 다음과 같다. 먼저 소수 p를 선택하고 $\mathbb{Z}_p$에서 타원곡선 E를 하나 선택한다. 그 다음에 소수 q인 이 곡선 위에 점 P를 하나 선택한다. 즉, P는 위수가 소수 q인 유한 덧셈순환군의 생성자가 된다. 이와 같은 시스템 환경에서 사용자의 개인키는 $x \in \mathbb{Z}_q$가 되고 대응되는 공개키는 $Q = xP$가 된다. 이 때 곡선의 임의의 점 A의 암호화는 다음과 같다.

- 단계 1. $r \in_R \mathbb{Z}_q$를 선택하고 $U = rP$를 계산한다.
- 단계 2. $r \in_R \mathbb{Z}_q$를 선택하고 $V = A + rQ$를 계산한다.

결과 쌍 $\langle U, V\rangle$가 점 A에 대한 암호문이 된다. 이것의 복호화는 사용자의 개인키 x를 이용하여 다음과 같이 이루어진다.

- 단계 1. $V - xU$를 계산한다.

이 결과가 A가 된다는 것은 다음 식을 통해 확인할 수 있다.

$$V - xU = A + rQ - xrP = A + rxP - xrP = A$$

실제 메시지를 이와 같은 방식으로 암호화하기 위해서는 일반 메시지를 타원곡선 점으로 매핑해야 한다.

9.5 신원기반 암호시스템

9.5.1 신원기반 암호시스템의 개요

1984년 Shamir는 전자우편주소나 주민등록번호처럼 사용자의 잘 알려진 신원정보로부터 그 사용자의 공개키를 유도해내는 신원기반 공개키 암호시스템(identity-based public-key cryptosystem)을 처음으로 소개하였다. 공개키를 그 사용자의 신원정보로부터 계산할 수 있음으로 기존 공개키 시스템과 달리 공개키가 누구의 공개키인지 확인하기 위한 별도의 메커니즘이 필요 없다. 즉, 어떤 사용자의 주민번호로 만들어진 공개키는 그 사용자의 공개키일 수밖에 없으며, 다른 사용자의 공개키가 될 수 없다. 따라서 기존 시스템과 달리 공개키 인증서가 필요 없으며, 공개 인증서와 관련된 여러 가지 요소가 필요 없다. 따라서 이 시스템의 가장 큰 장점은 통신하고자 하는 상대방이나 제3의 서버에 문의하지 않고 상대방의 신원을 알고 있다면 상대방의 공개키를 생성할 수 있다는 것이다. 하지만 이 장점은 이론상으로만 또는 이상적인 경우에만 이렇다는 것이다.

공개키는 사용자의 신원 정보로부터 계산할 수 있다. 그러면 대응되는 개인키는 어떻게 계산하는가? 개인키를 사용자들이 공개키로부터 직접 계산할 수 있다고 하면 다른 사용자들도 다른 사용자의 개인키를 계산할 수 있다는 것을 의미한다. 그러나 이렇게 되면 공개키 시스템으로 사용할 수 없다. 따라서 신원기반 공개키 암호시스템에서 개인키는 특별한 신뢰기관이 생성하여 주며, 이 기관을 PKG(Private Key Generator)라 한다. 그런데 PKG는 모든 사용자의 개인키를 만들 수 있으므로 이 기관은 모든 암호문을 복호화할 수 있고, 모든 사용자의 서명을 위조할 수 있다. 즉, 너무 강력한 권한을 가지는 기관이 된다. 따라서 이 문제를 해결하고자 여러 기관에 발급 권한을 분배하여 n개의 기관 중 t개의 기관 이상이 협조해야 사용자의 개인키를 만들 수 있도록 하는 방법을 사용한다. 즉, 권한 남용을 어렵도록 만들며, 이와 같은 기법을 암호기술에서는 threshold 기법이라 한다.

Shamir가 1984년에 처음 신원기반 공개키 시스템을 소개한 후에 지속적인 연구가 이루어졌지만 주로 신원기반 전자서명 알고리즘에 제한되었으며, 신원기반 공개키 암호알고리즘에 대한 성공적인 연구가 많지 않았다. 그런데 2001년에 Boneh와 Franklin이 겹선형 쌍함수를 이용한 신원기반 암호시스템을 제안한 이후로 이 분야에 대한 연구가 다시 활발하게 이루어지고 있다.

9.5.2 신원기반 암호시스템에서 신원정보

앞서 언급한 바와 같이 공개키는 사용자의 신원정보로부터 계산되고, 대응되는 개인키는 PKG에 의해 생성된다. 이 때 PKG는 PKG의 개인키와 사용자의 공개키를 이용하여 사용자의 개인키를 생성한다. 그런데 공개키는 사용자마다 달라야 하므로 공개키를 생성하기 위해 사용되는 사용자의 신원정보는 사용자마다 독특해야 한다. 또한 신원기반의 장점인 제3자에게 물어보지 않고 다른 사용자의 공개키를 생성할 수 있기 위해서는 누구나 쉽게 알 수 있는 신원정보이어야 한다. 따라서 전자우편주소와 같은 신원정보가 가장 적합한 신원정보이다. 하지만 전자우편주소나 주민번호는 불변정보이다. 즉, 바꿀 수 없는 정보이다. 물론 전자우편주소는 바꿀 수 있는 정보이지만 공개키를 바꾸기 위해 전자우편주소를 바꿀 수는 없다. 즉, 이와 같은 신원정보만을 사용하여 공개키를 생성할 경우에 나중에 공개키를 변경해야 한다면 변경하기가 매우 어렵다.

보다 구체적으로 살펴보면 공개키를 변경하기 위해서는 사용된 신원정보를 변경하거나 PKG의 개인키를 변경해야 한다. 하지만 전자의 경우에는 불변정보를 변경해야 하며, 후자는 모든 사용자의 개인키를 변경해야 한다. 따라서 둘 다 가능한 시나리오가 아니다. 이를 해결하기 위해 공개키를 생성할 때 사용자의 고유한 불변정보뿐만 아니라 발급시간과 같은 동적 정보를 사용할 수 있다. 이 경우에 공개키를 변경해야 할 경우 불변정보는 그대로 둔 상태에서 동적 정보만 변경하면 된다. 하지만 동적 정보를 사용하면 신원기반 시스템의 장점인 제3자에게 문의하지 않고 공개키를 생성할 수 있다는 장점은 퇴색된다. 즉, 사용자들은 이제 다른 사용자의 공개키를 생성하기 위해 다른 사용자의 신원정보뿐만 아니라 공개키 생성할 때 사용된 동적 정보까지 필요하다. 발급 시간을 동적 정보로 사용한다면 고정된 발급 시간을 사용함으로써 동적 정보의 사용 문제를 일부 완화할 수는 있다. 하지만 번거로운 측면이 여전히 존재한다. 이 교재에서는 공개키 생성에서 동적 정보의 포함은 알고리즘을 기술할 때 생략한다.

9.5.3 Boneh와 Franklin의 신원기반 암호시스템

Boneh와 Franklin은 겹선형 쌍함수를 활용하는 신원기반 암호알고리즘을 제안하였다. 겹선형 쌍함수를 활용하기 때문에 이 시스템을 설정하기 위해서는 위수가 동일한 타원곡선 기반 덧셈군 G_1과 유한순환 곱셈군 G_2가 필요하다. 따라서 Boneh와 Franklin의 신원기반 암호시스템의 시스템 설정은 다음과 같이 이루어진다.

- **단계 1.** 소수 p를 선택하고 $\mathbb{Z}_p$에서 타원곡선 E를 하나 선택한다. 그 다음에 소수 q인 이 곡선 위에 점 P를 하나 선택하여 G_1군을 생성한다.
- **단계 2.** 위수가 동일한 소수 q인 유한순환 곱셈군 G_2를 생성한다.
- **단계 3.** 겹선형 쌍함수 $\hat{e}: G_1 \times G_1 \rightarrow G_2$를 생성한다.
- **단계 4.** PKG의 개인키 $s \in \mathbb{Z}_q$를 선택하고, 공개키 $P_{pub} = sP$를 계산한다.
- **단계 5.** 충돌회피 해쉬함수 $H_1 : \{0,1\}^* \rightarrow G_1^*$과 $H_2 : G_2 \rightarrow \{0,1\}^n$를 선택한다.

단계 4에서 PKG의 개인키를 신원기반 시스템에서는 마스터 키(master key)라고도 한다. PKG의 마스터 키를 제외하고는 시스템 설정을 통해 생성된 모든 정보를 공개한다. 사용자의 공개키는 해쉬함수 H_1을 이용하여 생성한다. 사용자의 신원정보가 sangjin@kut.ac.kr이면 이 사용자의 공개키는 $Q_{ID} = H_1(\mathrm{sangjin@kut.ac.kr})$가 된다. 이 사용자의 개인키는 PKG가 자신의 마스터 키 s를 이용하여 다음과 같이 계산한다,

$$d_{ID} = s\,Q_{ID}$$

사용자의 공개키 Q_{ID}를 이용한 메시지 $m \in \{0,1\}^n$의 암호화는 다음과 같이 이루어진다.

- **단계 1.** $r \in_R \mathbb{Z}_q$를 선택하고 $U = rP$를 계산한다.
- **단계 2.** $g_{ID} = \hat{e}(Q_{ID}, P_{pub})$를 계산한 후에 $V = m \oplus H_2(g_{ID}^r)$를 계산한다.

결과 쌍 $C = \langle U, V \rangle$가 메시지 $m \in \{0,1\}^n$에 대한 암호문이다. 이 암호문을 복호화하기 위해서는 개인키 d_{ID}가 필요하며, 복호화하는 다음과 같이 이루어진다.

- 단계 1. $g_{ID}^r = \hat{e}(d_{ID}, U)$를 계산한다.
- 단계 2. $V \oplus H_2(g_{ID}^r) = m$을 계산한다.

복호화 과정의 정확성은 다음과 같은 식을 통해 확인할 수 있다.

$$g_{ID}^r = \hat{e}(Q_{ID}, P_{pub})^r = \hat{e}(Q_{ID}, sP)^r = \hat{e}(Q_{ID}, P)^{sr} = \hat{e}(s\,Q_{ID}, rP) = \hat{e}(d_{ID}, U)$$

참고문헌

[1] R. Soloway and V. Strassen, "A Fast Monte-Carlo Test for Primality," SIAM J. on Computing, Vol. 6, pp. 84-85, 1977.

[2] M.O. Rabin, "Probabilistic Algorithms for Testing Primarity," J. of Number Theory, Vol. 12, pp. 128-138, 1980.
[3] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of ACM, Vol. 21, pp. 120-126, 1978.
[4] Whitfield Diffie and Martin E. Hellman, "New Directions in Cryptography," IEEE Trans. on Information Theory, Vol. 22, pp. 644-654, 1976.
[5] Adi Shamir, "Identity-based Cryptosystems and Signature Schemes," Advances in Cryptology, Crypto 1984, LNCS 196, pp. 47–53, 1985.
[6] Dan Boneh and Matthew K. Franklin, "Identity-based Encryption from Weil pairing," Advances in Cryptology, Crypto 2001, LNCS 2139, pp. 213–229, 2001.

연습문제

1. $p = 11$, $q = 13$을 사용하여 RSA 공개키 쌍을 생성하시오.
2. $\mathbb{Z}_{31}^{*}$의 생성자를 찾는 효과적인 방법을 설명하시오.