제 8 장 공개키 암호알고리즘 관련 수학적 배경

공개키 암호알고리즘은 비밀키 암호알고리즘과 달리 대부분 풀기 어려운 계산 문제(computational problem)에 기반을 두고 있다. 이와 같은 계산 문제들은 풀기 어렵다고 증명되어 있지는 않다. 즉, 이와 같은 문제를 해결하여 주는 알고리즘들이 존재하지만 가장 좋은 알고리즘을 사용하더라도 그것을 기반으로 하는 공개키 암호알고리즘의 안전성을 해칠 정도로 적은 비용과 빠른 시간 내에 해결할 수 없다는 것이다. 그러나 반대로 이와 같은 문제를 효율적으로 해결할 수 있으면 해당 문제에 의존하는 공개키 암호알고리즘의 안전성은 더 이상 보장될 수 없다. 따라서 공개키 암호알고리즘들은 컴퓨팅 기술의 발달로 키의 길이 등을 지속적으로 늘려주어야 하는 문제점을 공통적으로 가지고 있다.

현재 공개키 암호알고리즘에서 가장 많이 사용되는 계산 문제는 인수분해 문제, 이산대수 문제이다. 이 장에서 이와 같은 문제들의 특성을 살펴본다.

8.1 인수분해 문제

정수의 인수분해 문제(integer factorization problem)란 주어진 합성수의 소인수를 모두 찾는 문제를 말한다. 수학적으로 이 문제를 정의하면 다음과 같다.

정의 8.1 (정수의 인수분해 문제) 양의 정수 n이 주어졌을 때 $n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$를 만족하는 서로 다른 소수 p_i와 양의 정수 e_i를 찾는 문제를 말한다.

정수의 인수분해 문제와 **소수 검사 문제**(primality decision problem)를 같은 문제로 생각하는 경우도 있다. 하지만 어떤 정수가 합성수인지 소수인지 결정하는 것은 그것을 소인수분해하는 것보다 쉽다. 주어진 정수를 인수분해하는 가장 단순한 방법은 그 수보다 작은 모든 수로 나누어 보는 것이다. 이 때 다음 정리를 이용하면 모든 수로 나누어보지 않고 인수분해를 할 수 있다.

정리 8.1. n이 합성수이면 n은 $\sqrt{n}$보다 작은 소인수를 가진다.

증명. n이 합성수이므로 $n = ab(1 < a \le b < n)$로 표현할 수 있다. 그런데 이 때 $a < \sqrt{n}$이어야 한다. $b \ge a > \sqrt{n}$이면 $ab > \sqrt{n} \cdot \sqrt{n} = n$이므로 $a < \sqrt{n}$이 되어야 한다는 것을 알 수 있다. 뿐만 아니라 1보다 큰 모든 양의 정수는 소인수를 가지므로 a는 소인수를 가지며, 이 소인수는 $\sqrt{n}$보다 작으므로 n은 $\sqrt{n}$보다 작은 소인수를 가진다.

따라서 n를 소인수분해하기 위해서는 $\sqrt{n}$보다 작은 소수들로 나누어 보면 된다. 이와 같은 방법으로 인수분해하는 것을 “trial division"이라 한다. 이 방법을 사용할 경우 최악에는 $O(\sqrt{n})$번 나눗셈을 시도해야 한다.

Pollard는 합동을 이용한 두 가지 인수분해 알고리즘을 제안하였다. 첫 번째 알고리즘은 rho 알고리즘이라 하며, 그것의 원리는 다음과 같다. p가 n의 소인수일 때 $x_i \equiv x_j \pmod{p}$

이지만 $x_i \not\equiv x_j \pmod n$이면 $\gcd(x_i - x_j, n)$은 n의 자명하지 않은 약수가 된다. 이것은 $x_i - x_j$는 p의 배수이지만 n의 배수는 아니므로 $x_i - x_j$와 n은 공통된 약수를 가지기 때문이다. 그런데 최대공약수는 유크리드 알고리즘을 통해 빠르게 계산할 수 있다. 따라서 $x_i \equiv x_j \pmod p$이지만 $x_i \not\equiv x_j \pmod n$인 x_i와 x_j를 찾을 수 있으면 n을 인수분해할 수 있다. 그러므로 이 방법의 복잡성은 앞서 설명한 조건을 만족하는 x_i와 x_j를 어떻게 빠르게 찾느냐에 의해 결정되며, Pollard가 제시한 방법을 사용하면 복잡성이 $O(n^{1/4})$이다.

두 번째 알고리즘은 $p-1$ 알고리즘이라 하며, 그 원리는 다음과 같다. 합성수 n이 소인수 p를 가지고 있으며, 양의 정수 k에 대해 $p-1 | k!$이면 $2^{k!} - 1 \bmod n$과 n은 공통된 약수를 가진다. 따라서 $\gcd(2^{k!} - 1 \bmod n, n)$은 n의 자명하지 않은 약수이다. 보다 자세히 이 원리를 살펴보면 다음과 같다. $p-1 | k!$이므로 $k! = (p-1)q$인 정수 q가 존재한다. 이 때 $2^{k!}$을 생각하여 보자. $2^{k!} = 2^{(p-1)q} = (2^{p-1})^q \equiv 1^q \equiv 1 \pmod p$임을 알 수 있다. 즉, $p | 2^{k!} - 1$이다. 따라서 $M = (2^{k!} - 1) \bmod n$이면 $p | 2^{k!} - 1$이고 $p | n$이므로 $p | M$이다. 따라서 $\gcd(M, n)$은 n의 자명하지 않은 소인수이다. 따라서 $r_k = 2^{k!}$라 하고, $\gcd(r_k - 1 \bmod n, n)$을 지속적으로 구하면 n을 인수분해할 수 있다.

다음 원리를 이용하는 인수분해 알고리즘도 있다. $x^2 \equiv y^2 \pmod n$이지만 $x \not\equiv \pm y \pmod n$이면 $n | (x+y)(x-y)$이지만 $n \nmid (x-y)$이다. 따라서 $\gcd(x-y, n)$은 n의 자명하지 않은 인수이다. 이 알고리즘을 "random squaring" 알고리즘이라 하며, 이 알고리즘의 복잡성도 Pollard의 rho 알고리즘과 만찬가지로 $x^2 \equiv y^2 \pmod n$이지만 $x \not\equiv \pm y \pmod n$인 x와 y를 찾는 것에 의해 결정된다.

이 외에도 elliptic curve 인수분해 알고리즘, quadratic sieve 알고리즘, number field sieve 알고리즘 등이 있으며, 이 중에 number field sieve가 지금까지 알려진 가장 빠른 범용 알고리즘이다. 하지만 n이 1024비트 이상의 수이면 이들을 이용하여 그것을 인수분해하는 것은 비용과 시간 측면에서 계산적으로 어렵다.

8.2 RSA 문제

이 문제는 공개키 암호알고리즘 중 현재 가장 널리 사용 중인 RSA 공개키 암호알고리즘이 기반으로 하고 있는 문제이며, 그 정의는 다음과 같다.

정의 8.2 (RSA 문제) 두 개의 서로 다른 소수 p와 q의 곱인 양의 정수 n, $\gcd(e, \phi(n)) = 1$인 양의 정수 e, 양의 정수 c가 주어졌을 때, $m^e \equiv c \pmod n$인 m을 찾는 문제를 RSA 문제라 한다.

이 문제는 n을 인수분해할 수 있으면 쉽게 다음과 같이 해결할 수 있다.

- 단계 1. n을 인수분해할 수 있으므로 $\phi(n) = (p-1)(q-1)$을 계산한다.
- 단계 2. $ed \equiv 1 \pmod{\phi(n)}$인 d를 찾는다. (확장 유크리드 알고리즘 이용)
- 단계 3. 다음 식을 이용하여 m을 계산한다.

$$c^d = m^{ed} = m^{\phi(n)k+1} \equiv m^{\phi(n)k} m \equiv m \pmod n$$

RSA 문제는 다항시간 내에 인수분해 문제로 변환할 수 있다. 즉, 인수분해 문제를 해결할 수 있으면 RSA 문제를 해결할 수 있다. 반대로 RSA 문제를 해결하면 인수분해 문제를 해결할 수 있다는 것은 증명되어 있지 않다. 하지만 많은 사람들은 RSA 문제와 인수분해 문제가 등가라고 믿고 있다.

8.3 이차잉여 문제

이차잉여 문제(QRP, Quadratic Residuosity Problem)도 인수분해와 매우 관련이 있는 문제 중 하나이다. $\gcd(a,n)=1$인 정수 a에 대해 이차합동식 $x^2 \equiv a \pmod n$이 해를 가지면 a를 법 n에 관한 **이차잉여**(quadratic residue)라 하고, 해를 갖지 않으면 이차비잉여(quadratic nonresidue)라 한다. 즉, 쉽게 표현하면 어떤 정수 a가 법 n에서 제곱근을 가지면 법 n에서 a는 이차잉여라 한다. 이 때 법 n의 모든 이차잉여들의 집합을 Q_n이라 하며. 법 n의 모든 이차비잉여들의 집합을 $\overline{Q}_n$이라 한다. 이차잉여와 관련하여 다음과 같은 기호가 정의되어 있으며, 이 기호를 **르장드로 기호**(Legendre symbol)라 한다.

$$\left(\frac{a}{p}\right)=\begin{cases} 0, & \text{if } p|a \\ 1, & \text{if } a \in Q_p \\ -1, & \text{if } a \in \overline{Q}_p \end{cases}$$

이 기호는 법이 소수일 때 의미가 있는 기호이다. 법의 소수가 아닐 경우에는 대신 **야코비 기호**(Yacobi symbol)가 있다. 이 기호는 르장드로 기호를 이용하여 정의되며, $n(\geq 3)$의 소인수분해가 $p_1^{e_1}p_2^{e_2}\cdots p_r^{e_r}$이고 a가 정수일 때, 야코비 기호는 다음과 같이 정의된다.

$$\left(\frac{a}{n}\right)=\left(\frac{a}{p_1}\right)^{e_1}\left(\frac{a}{p_2}\right)^{e_2}\cdots\left(\frac{a}{p_r}\right)^{e_r}$$

여기서 $\left(\frac{a}{p_i}\right)$는 르장드로 기호이다. 르장드로 기호는 야코비 기호들의 곱이므로 결과는 르장드로 기호와 마찬가지로 0, -1, 1이다. 하지만 르장드로 기호와 달리 야코비 기호 $\left(\frac{a}{n}\right)$이 1이어도 a가 법 n에서 이차잉여가 아닐 수 있다. 한편 이 값이 -1이나 0이면 a는 법 n에서 비이차잉여가 확실하다. 따라서 J_n이 야코비 기호 값이 1인 모든 $a \in \mathbb{Z}_n^*$들의 집합이라고 할 때, 이차잉여 문제의 정의는 다음과 같다.

정의 8.3 (**이차잉여 문제**) 홀수인 양의 정수 n과 $a \in J_n$이 주어졌을 때, a가 법 n에서 이차잉여인지를 결정하는 문제를 이차잉여 문제라 한다.

르장드로 기호나 야코비 기호 값을 계산하여 주는 효율적인 알고리즘이 존재한다. 즉, n을 인수분해하지 않고도 야코비 기호 값을 계산하여 주는 효율적인 알고리즘이 존재한다. 하지만 야코비 기호 $\left(\frac{a}{n}\right)$이 1이어도 a가 법 n에서 이차잉여가 아닐 수 있기 때문에 이 문제가 어려운 문제이다. 여기서 n이 일반적인 합성수일 때는 문제가 복잡해지므로 암호기술에서 널리 사용되는 n이 서로 다른 두 소수 p와 q의 곱일 때로 한정하여 이 문제를 생각하

면 다음과 같다. $n=pq$이므로 $\left(\frac{a}{n}\right)=\left(\frac{a}{p}\right)\left(\frac{a}{q}\right)$이다. 따라서 야코비 기호 $\left(\frac{a}{n}\right)$이 1이면 $\left(\frac{a}{p}\right)$와 $\left(\frac{a}{q}\right)$가 모두 1이거나 $\left(\frac{a}{p}\right)$와 $\left(\frac{a}{q}\right)$가 모두 -1이어야 한다.

정리 8.1 n이 서로 다른 두 소수 p와 q의 곱일 때 $a \in Q_n$이기 위한 필요충분조건은 $a \in Q_p$이고 $a \in Q_q$이어야 한다.

증명. $a \in Q_n$라는 것은 $f(x)=x^2-a$일 때 다항합동식 $f(x) \equiv 0 \pmod{n}$이 해가 존재한다는 것을 말한다. 이 때 c가 다항합동식 $f(x) \equiv 0 \pmod{n}$의 해이면 이 c는 $f(x) \equiv 0 \pmod{p}$와 $f(x) \equiv 0 \pmod{q}$의 해이다. 즉, $a \in Q_n$이면 $a \in Q_p$이고 $a \in Q_q$이다. 거꾸로 $a \in Q_p$이고 $a \in Q_q$이면 $f(x) \equiv 0 \pmod{p}$와 $f(x) \equiv 0 \pmod{q}$가 각각 해를 가진다. 이 때 이 해가 c_p와 c_q라 하자. 그러면 중국인 나머지 정리에 이용하여 $x \equiv c_p \pmod{p}$와 $x \equiv c_q \pmod{q}$를 동시에 만족하는 법 n에서 c를 찾을 수 있다. 그런데 $c^2 \equiv a \pmod{p}$이고 $c^2 \equiv a \pmod{q}$이므로 $c^2 \equiv a \pmod{n}$이다. 따라서 이 c는 $f(x) \equiv 0 \pmod{n}$의 해가 되므로 $a \in Q_n$이다.

따라서 야코비 기호 $\left(\frac{a}{n}\right)$가 1일 때, $\left(\frac{a}{p}\right)$와 $\left(\frac{a}{q}\right)$가 모두 1인지 확인할 수 있다면 $a \in Q_n$인지 알 수 있다. 즉, n을 인수분해할 수 있으면 $a \in J_n$이 주어졌을 때, a가 법 n에서 이차잉여인지를 결정할 수 있다. 여기서 야코비 기호 $\left(\frac{a}{n}\right)$가 1일 때 $a \in Q_n$일 확률은 50%이다. 그 이유는 다음과 같다. $\mathbb{Z}_p^*$는 생성자를 가진 순환군이므로 이 군의 생성자가 g일 때 이 군의 모든 원소는 g^i로 표현할 수 있다. 이 때 i가 짝수이면 g^i는 법 p에서 이차잉여가 된다. 따라서 $|Q_p|=(p-1)/2$이다. 즉, $\mathbb{Z}_p^*$의 원소의 반은 이차잉여이고, 나머지 반은 비이차잉여이다. 마찬가지로 $|Q_q|=(q-1)/2$이다. a가 $\mathbb{Z}_p^*$의 이차잉여이고 b가 $\mathbb{Z}_q^*$의 이차잉여일 때, 중국인 나머지 정리를 이용하여 다음 연립합동식의 해를 구하면

$$x \equiv a \pmod{p}$$
$$x \equiv b \pmod{q},$$

이 해는 $\mathbb{Z}_{pq}^*$의 이차잉여가 된다. 그 이유는 이 해는 법 p에서 이차잉여이고, 법 q에서 이차잉여이므로 정리 8.1에 의해 법 n에서 이차잉여가 된다. 즉, $\mathbb{Z}_p^*$와 $\mathbb{Z}_q^*$의 이차잉여를 하나씩 가지면 $\mathbb{Z}_{pq}^*$의 이차잉여를 독특하게 하나 만들 수 있다. 따라서 $\mathbb{Z}_{pq}^*$의 이차잉여의 개수는 $|Q_{pq}|=|Q_p||Q_p|=(p-1)(q-1)/4$이다. 이 사실은 다음과 같은 원리를 이용해서도 성립함을 알 수 있다. 법 n에서 $a \in \mathbb{Z}_n^*$가 이차잉여이면 $x^2-a \equiv 0 \pmod{n}$는 정확하게 4개의 해를 가진다. 이들 4개가 하나의 이차잉여를 나타내므로 법 n에서 이차잉여는 총 $(p-1)(q-1)/4$개 존재한다. 같은 이유에서 $\mathbb{Z}_p^*$의 비이차잉여와 $\mathbb{Z}_q^*$의 비이차잉여를 하나씩 이용하여 J_n에 속하지만 Q_n에 속하지 않는 $\mathbb{Z}_{pq}^*$의 원소를 독특하게 하나 만들 수 있다. 따라서 J_n의 크기는 $(p-1)(q-1)/2$이고 Q_n는 $(p-1)(q-1)/4$이므로 야코비 기호 $\left(\frac{a}{n}\right)$가 1일 때 $a \in Q_n$일 확률은 50%가 된다.

8.4 법 n에서 제곱근 문제

이차잉여 문제는 주어진 수가 제곱근을 가지고 있는지 결정하는 문제이다. 반면 제곱근 문제는 법 n의 이차잉여가 주어졌을 때 제곱근 자체를 계산하는 문제를 말한다. 이 문제의 정의는 다음과 같다.

정의 8.4 (법 n에서 제곱근 문제) 합성수 n과 법 n에서 이차잉여 a가 주어졌을 때, 법 n에서 a의 제곱근을 찾는 문제를 말한다.

문제의 정의에서도 알 수 있듯이 주어진 법 n이 합성수일 경우에만 이 문제는 어려운 문제이다. 즉, 소수 p가 법일 경우에는 그것의 제곱근은 효율적으로 계산할 수 있다. 예를 들어 p가 $4k+3$형태이면 이 법에서 이차잉여 a의 제곱근은 $a^{(p+1)/4}$이다. 그 이유는 다음 사실로부터 유추할 수 있다. p가 소수이고 a가 법 p에서 이차잉여이면 $a^{(p-1)/2} \equiv 1 \pmod{p}$이다. 이것이 성립하는 이유는 b가 a의 제곱근이면 다음이 성립하기 때문이다.

$$a^{(p-1)/2} = (b^2)^{(p-1)/2} = b^{p-1} \equiv 1 \pmod{p}$$

따라서 다음이 성립하므로 법 p에서 이차잉여 a의 제곱근은 $a^{(p+1)/4}$이다.

$$a^{(p+1)/2} = a^{(p-1)/2}a \equiv a \pmod{p}$$

이차잉여와 마찬가지로 n이 두 개의 소수의 곱일 경우로 한정하여 이 문제를 생각하여 보자. 서로 다른 소수 p와 q의 곱인 양의 정수 n과 법 n에서 이차잉여 a가 주어졌을 때, n을 인수분해할 수 있으면 다음과 같은 방법을 통해 이 법에서 a의 제곱근을 계산할 수 있다.

- **단계 1.** 법 p에서 a의 제곱근 r과 $-r$을 구하고, 법 q에서 a의 제곱근 s과 $-s$을 구한다.
- **단계 2.** 확장 유크리드 알고리즘을 이용하여 $cp+dq=1$인 c와 d를 구한다.
- **단계 3.** $x = rdq+scp \bmod n$과 $y = rdq-scp \bmod n$은 법 n에서 a의 제곱근이 된다.

여기서 단계 2와 단계 3은 중국인 나머지 정리에서 다음 연립합동식을 구하기 위한 절차이다.

$$x \equiv r \pmod{p}$$
$$x \equiv s \pmod{q}$$

이 처럼 인수분해 문제를 해결할 수 있으면 법 n에서 제곱근 문제도 해결할 수 있다. 그런데 제곱근 문제는 RSA나 이차잉여 문제와 달리 그 역도 성립한다. 즉, 법 n에서 제곱근 문제를 해결할 수 있으면 인수분해 문제를 해결할 수 있다. 이것은 random squaring 인수분해 알고리즘을 적용할 수 있는 두 개의 이차잉여를 찾을 수 있기 때문이다. 좀 더 구체적으로 설명하면 법 n에서 제곱근 문제를 해결할 수 있는 A라고 하는 효율적인 알고리즘이 있으면 다음과 같은 방법을 통해 n을 인수분해할 수 있다.

- **단계 1.** $\gcd(x,n)=1$인 정수 x를 랜덤하게 선택한다.
- **단계 2.** $a=x^2 \bmod n$을 계산한다.
- **단계 3.** n과 a를 이용하여 알고리즘 A를 수행하여 a의 제곱근 y를 얻는다.
- **단계 4.** $y \equiv \pm x \pmod n$이면 단계 3을 다시 수행한다.
- **단계 5.** $y \not\equiv \pm x \pmod n$이면 $\gcd(x-y,n)$은 n의 자명하지 않은 인수가 된다.

8.5 이산대수 문제

이산대수 문제(DLP, Discrete Logarithm Problem)는 인수분해 문제와 더불어 암호기술에서 가장 널리 사용하는 문제이다. 먼저 이산대수의 정의는 다음과 같다.

정의 8.4 (이산대수) 유한순환군 $G=\langle g \rangle$에서 기저 g에 대한 $y \in G$의 이산대수란 $y=g^x$인 x, $1 \le x < |G|$를 말하며, $\log_g y$로 표기한다.

어떤 원소의 이산대수를 다른 말로 그 원소의 색인(index)이라 한다. 일반 로그 관련 성질은 이산대수에서도 성립한다. 예를 들어 $G_n=\langle g \rangle$이고, $a,b,c \in G_n$이면 다음이 성립한다.

- 모든 $x \in \mathbb{Z}$에 대해 $\log_g a^x \equiv x \log_g a \pmod n$이다.
- $\log_g(ab) \equiv \log_g a + \log_g b \pmod n$이다.

이산대수 문제란 유한순환군의 원소가 주어졌을 때, 그것의 이산대수를 찾는 문제를 말하며, 그것의 정의는 다음과 같다.

정의 8.5 (이산대수 문제) 유한순환군 $G=\langle g \rangle$와 어떤 $y \in G$가 주어졌을 때, $y=g^x$인 x, $1 \le x < |G|$를 찾는 것을 이산대수 문제라 한다.

정의에서 알 수 있듯이 이산대수 문제는 g^1, g^2과 같이 차례대로 구하여 결국에는 찾을 수 있다. 하지만 군의 위수가 매우 크면 이와 같은 방법으로 찾는 것은 계산적으로 어렵다. 이와 같이 가능한 모든 경우를 검사하여 찾는 방법보다 효율적인 방법들이 제안되어 있지만 군의 위수가 클 경우에는 이들 방법을 사용하여도 계산적으로 어렵다. 지금까지 제안된 유한순환군에서 DLP의 해를 찾는 알고리즘은 다음과 같이 크게 두 종류로 분류된다.

- (범용 알고리즘) Pollard의 rho 알고리즘, Baby-step/Giant-step 알고리즘과 같이 임의의 군에 적용할 수 있는 알고리즘
- (전용 알고리즘) Index Calculus 알고리즘처럼 특정 군에만 적용 가능한 알고리즘

범용 알고리즘들의 경우에는 군의 위수가 n이면 복잡성이 $O(\sqrt{n})$이다. 따라서 군의 크기가 160비트 정도만 되어도 현재 컴퓨팅 수준으로는 그 해를 찾는 것은 계산적으로 어렵다. Index calculus 알고리즘은 p가 소수일 때 $\mathbb{Z}_p^*$ 군에 효율적으로 적용할 수 있는 알고리즘이

지만 이 알고리즘의 복잡성은 $O(e^{((\ln p)^{1/3}(\ln(\ln p))^{2/3})})$이다. 이와 같은 복잡성은 직관적으로 이해하기 어렵지만 p가 512비트 정도이면 $O(2^{70})$ 정도이다. 즉, p가 512비트보다 큰 소수이면 $\mathbb{Z}_p^*$에서 이산대수를 구하는 것 역시 계산적으로 어렵다.

유한순환군에서 DLP의 해를 찾는 알고리즘 중에 baby-step/giant-step 알고리즘을 자세히 살펴봄으로써 어떤 방법으로 DLP 문제를 해결할 수 있는지 알아보자. 이 알고리즘은 다음 원리를 이용한다.

- $G_n = \langle g \rangle$의 원소 y의 이산대수가 x일 때 $x = im + j$로 표현할 수 있다. 여기서 $m = \sqrt{n}$이다. 그러면 $g^x = g^{im} g^j$이며, $y(g^{-m})^i = g^j$이다. 따라서 가능한 g^j를 모두 구하고, $y(g^{-m})^i$를 차례로 구하면서 이미 구한 g^j와 일치하는 값을 찾으면 x를 계산할 수 있다.

예8.1) $\mathbb{Z}_{113}^* = \langle 3 \rangle$에서 $\log_3 44$는?

- $m = \lceil \sqrt{112} \rceil = 11$

j	0	1	2	3	4	5	6	7	8	9	10
$3^j \bmod 113$	1	3	9	27	81	17	51	40	7	21	63

검색하기 쉽도록 정렬한다.

j	0	1	8	2	5	9	3	7	6	10	4
$3^j \bmod 113$	1	3	7	9	17	21	27	40	51	63	81

- $3^{-11} \equiv 3^{101} \equiv 58 \pmod{113}$

i	0	1	2	3	4	5	6	7	8	
$44 \cdot 58^i \bmod 113$	44	66	99	92	25	94	28	42	63	

- $x = 8 \cdot 11 + 10 = 98$

암호기술에서는 많이 사용하는 유한순환군은 $\mathbb{Z}_p^*$이다. 앞서 언급한 바와 같이 이 군에서 DLP 문제를 해결할 수 있는 가장 효율적인 알고리즘은 index calculus 알고리즘이다. 유한순환군의 모든 부분군은 역시 유한순환군이다. 특히 이들 부분군 중에 위수가 소수인 부분군들이 존재하며, 항등원을 제외한 이 군의 모든 원소는 그 군의 생성자가 된다. 그러면 부분군에서 이산대수를 해결하는 비용은 원 군에서 이산대수를 해결하는 비용과 어떤 관계가 있을까? 군의 크기가 줄어들기 때문에 단순하게 생각하면 쉬워질 것으로 생각할 수 있다. 그런데 $\mathbb{Z}_p^*$에서 DLP를 해결할 수 있는 가장 효율적인 알고리즘은 index calculus 알고리즘이다. 반면에 $\mathbb{Z}_p^*$의 위수가 소수 q인 부분군 G_q에서 DLP는 $\mathbb{Z}_p^*$에 대한 index calculus를 적용하여 구할 수 있고, 범용 알고리즘을 직접 G_q에 적용하여 구할 수 있다. 그런데 p가 512비트 정도이고 q가 140비트 정도이면 index calculus를 이용하나 범용 알고리즘을 이용하나 G_q에서 DLP를 해결하기 위한 비용은 비슷하다. 즉, $\mathbb{Z}_p^*$을 사용하나 그것의 부분군 G_q를 사용하나 안전성에는 차이가 없다. 따라서 안전성에는 차이가 없으므로 $\mathbb{Z}_p^*$ 대신에 그것의 부분군인 G_q을 사용하면 군의 크기가 축소되었으므로 연산의 효율성을 높일 수 있다.

8.6 Diffie-Hellman 문제

이산대수 문제와 매우 밀접한 또 다른 문제는 Diffie-Hellman 문제이다. Diffie와 Hellman은 공개키 방식을 처음으로 제안하였으며, 이들이 제안한 유명한 암호프로토콜이 기반으로 하고 있는 문제를 Diffie-Hellman 문제라 한다. Diffie-Hellman 문제는 크게 **Diffie-Hellman 계산 문제**(computational problem)와 **Diffie-Hellman 결정 문제**(decision problem)로 나누어지며, 그것의 정의는 다음과 같다.

정의 8.6 (**Diffie-Hellman 계산 문제**) 유한순환군 $G=\langle g\rangle$와 G의 임의의 두 원소 g^a과 g^b가 주어졌을 때 g^{ab}를 계산하는 문제를 Diffie-Hellman 계산 문제라 한다.

정의 8.7 (**Diffie-Hellman 결정 문제**) 유한순환군 $G=\langle g\rangle$와 G의 임의의 세 원소 g^a, g^b, g^c가 주어졌을 때 g^{ab}와 g^c가 같은지 결정하는 문제를 Diffie-Hellman 결정 문제라 한다.

만약 이산대수 문제를 해결할 수 있으면 g^a로부터 a를 계산한 후에 $(g^b)^a$를 계산하여 g^{ab}를 계산할 수 있다. 즉, 이산대수 문제를 해결할 수 있으면 Diffie-Hellman 문제를 해결할 수 있다. 마찬가지로 Diffie-Hellman 문제를 해결할 수 있으면 g^a과 g^b로부터 g^{ab}를 계산하여 g^c와 비교하여 Diffie-Hellman 결정 문제를 해결할 수 있다. 그러나 이들 문제들이 등가임이 증명되어 있지 않다. 즉, Diffie-Hellman 문제를 해결하는 알고리즘을 이용하여 이산대수 문제를 해결할 수 있는지는 아직 증명되어 있지 않다.

8.7 타원곡선

기존 공개키 암호알고리즘들은 인수분해 또는 이산대수 문제의 어려움에 기반하고 있다. 이들 문제는 해결이 불가능한 문제는 아니다. 다만 주어진 크기의 정수를 인수분해하는 것이 계산적으로 어렵거나 주어진 군에서 이산대수 문제를 계산하는 것이 계산적으로 어렵다는 것이다. 그러나 컴퓨팅 능력이 발달하고, 이들 문제를 해결하는 알고리즘의 성능이 향상되면 보다 더 큰 정수를 사용하거나 보다 더 큰 위수의 군을 사용해야 한다. 그런데 사용하는 정수의 크기가 커지거나 사용하는 군의 위수가 커지면 그만큼 각 알고리즘에서 사용하는 연산의 비용이 높아진다. 물론 컴퓨팅 능력의 발달로 연산 비용이 높아지는 것을 부분적으로 상쇄할 수는 있지만 지속적으로 이와 같은 현상을 반복할 수는 없다. 따라서 같은 안전성을 제공하면서 키 길이는 짧은 다른 대안을 찾게 되었으며, 그 결과 제안된 것이 **타원곡선**(elliptic curve)이다. 타원곡선은 타원과 큰 연관은 없으며, 3차 방정식(cubic equation)을 사용한다.

타원곡선에 기반을 두고 있는 암호알고리즘의 기본 원리는 타원곡선을 이용하여 유한순환군을 만들 수 있다는 것이다. 이 유한순환군은 기존에 매우 큰 소수를 이용하여 만든 유한순환군을 대체할 수 있다. 따라서 이산대수에 기반을 두고 있는 기존 암호알고리즘을 그대로 타원곡선 기반으로 쉽게 전환할 수 있다. 하지만 기존에 매우 큰 소수를 이용하여 만든 유한순환군은 곱셈군이지만 타원곡선을 이용하여 만든 유한순환군은 덧셈군이다.

타원곡선은 다음과 같은 형태의 3차 방정식을 이용한다.

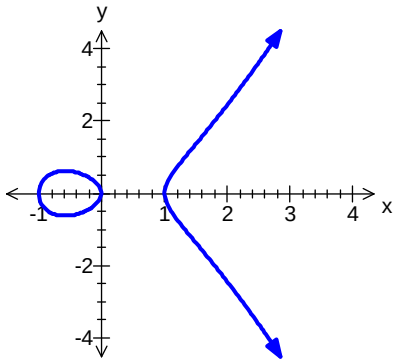
$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

이 방정식을 간단하게 $y^2 = x^3 + ax + b$ 형태로 축소하여 사용할 수 있다. 이와 같은 타원곡선을 이용하여 군을 형성하기 위해서는 $x^3 + ax + b$가 $y^2 = (x+2)(x-1)^2$처럼 중근을 가지면 안 된다. 즉, 다음 조건을 만족해야 한다.

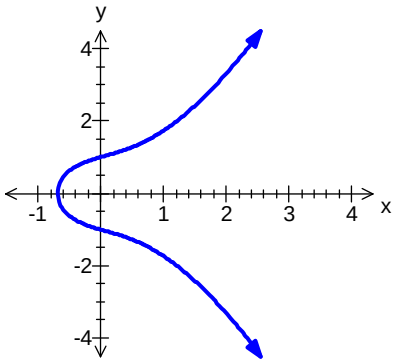
$$4a^3 + 27b^2 \neq 0$$

사용하는 3차 방정식에 의해 그려지는 곡선 위의 점들이 타원곡선을 이용한 군의 원소들이 된다. 이 때 무한대에 있는 가상의 점(point at infinity)이 추가로 이 군의 원소가 되며, 이 점이 타원곡선을 이용하여 만든 덧셈군의 항등원이 된다. 이 항등원은 $O(x, \infty)$로 표기된다.

그림 8.1과 8.2는 타원곡선의 예이다.

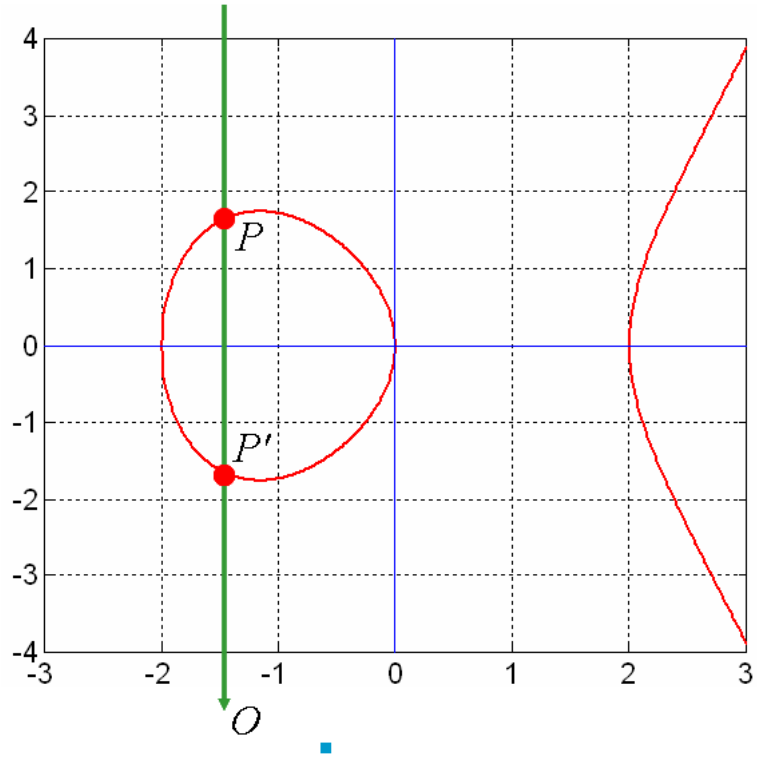


<그림 8.1> $y^2 = x^3 - x$



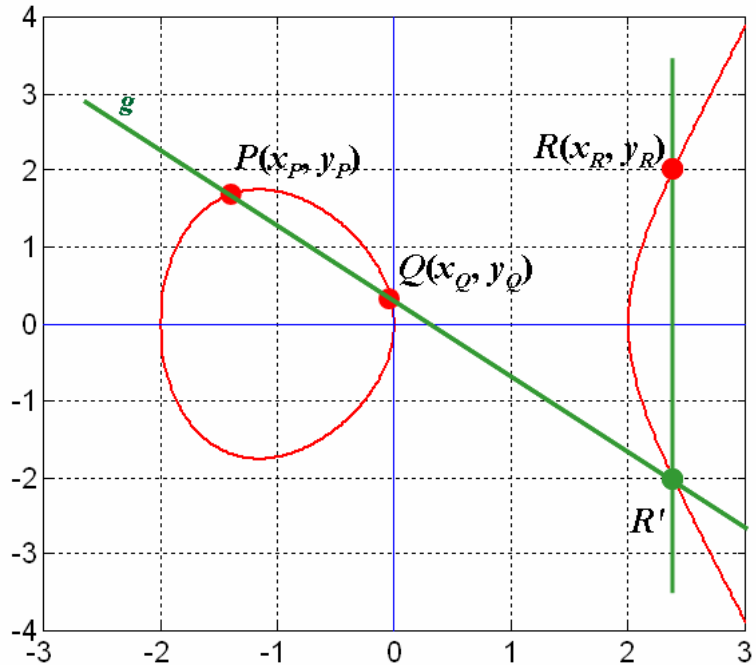
<그림 8.2> $y^2 = x^3 + x + 1$

타원곡선의 한 점 P의 역원은 그림 8.3과 같이 그 점을 지나는 y축과 평행인 선을 그렸을 때 교차하는 점이다. 즉, 점 P의 y 좌표의 부호만 바꾸면 점 P의 역원을 구할 수 있다.



<그림 8.3> 타원곡선 위의 점 P의 역원과 항등원

타원곡선의 서로 다른 두 점 P와 Q의 덧셈은 그림 8.4과 같이 이루어진다. 즉, 두 점을 잇는 직선을 그리면 이 직선은 곡선과 또 다른 위치에서 만나게 되며, 이 교차점 R'에서 y 축과 평행이 직선을 그렸을 때 만나는 또 다른 교차점 R이 덧셈의 결과가 된다.



<그림 8.4> 타원곡선 위의 서로 다른 두 점의 덧셈

이 R를 계산하기 위해서는 먼저 두 점 P와 Q를 잇는 직선이 곡선과 만나는 또 다른 교차점 R'을 계산해야 한다. 이 직선을 $y = sx + y_0$라 할 때 기울기는 $s = \frac{y_Q - y_P}{x_Q - x_P}$와 같으며, y_0는 P 또는 Q를 이용하여 계산할 수 있다. 예를 들어 $y_0 = y_P - s \bullet x_P$이다. 그러면 R'은 다음 식의 해를 구하여 계산할 수 있다.

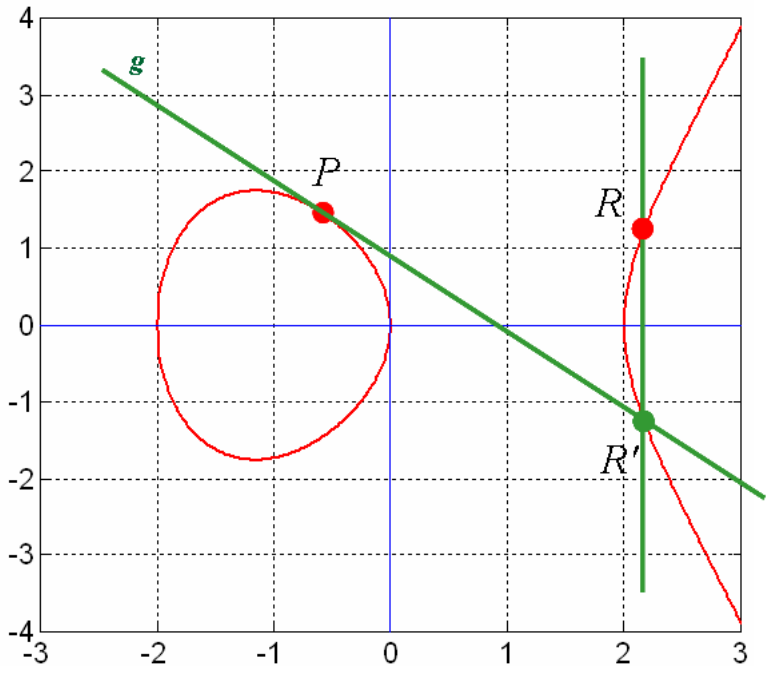
$$(s \bullet x + y_0)^2 = x^3 + ax + b$$

이 식은 $x^3 - (s \bullet x + y_0)^2 + ax + b = 0$로 표현할 수 있다. 모닉 다항식에서 근들의 합과 두 번째 최고차항의 계수를 더하면 0이 된다는 사실을 이용하면 R'의 x좌표를 보다 쉽게 구할 수 있다.

$$x_{R'} = s^2 - x_P - x_Q$$

그러면 y 좌표는 $y_{R'} = s \bullet x_{R'} + y_0$이다. 따라서 우리가 구하고자 하는 R은 $(x_{R'}, -y_{R'})$이다.

지금까지 설명한 것은 서로 다른 두 개의 점을 더하는 방법이고, 어떤 점을 자기 자신에 더하는 방법은 다음과 같다. 그림 8.5와 같이 그 점의 접선을 그려, 이 접선과 곡선이 만나는 또 다른 교차점을 찾는다. 서로 다른 두 점을 더할 때와 마찬가지로 이 점 대신에 그것의 역원이 덧셈의 결과가 된다.



<그림 8.5> 타원곡선 위의 한 점을 자기 자신에 더하는 방법

$P + P = 2P = R$을 계산하는 구체적인 방법은 다음과 같다. 먼저 접선을 구해야 하는데, 접선이 $y = sx + y_0$라 할 때, 기울기는 다음과 같으며, $y_0 = y_P - s \bullet x_P$이다.

$$s = \frac{dy}{dx} = \frac{3x_P^2 + a}{2y_P}$$

이 접선과 곡선과의 교차점은 서로 다른 두 점을 계산할 때와 마찬가지로 다음과 같이 계산

할 수 있다.

$$x_R = s^2 - 2x_P$$

$$y_R = -(s \bullet x_{R'} + y_0)$$

한 점의 연속적인 덧셈은 이 과정을 반복하면 된다.

지금까지의 설명은 실수에서 타원곡선에 대한 설명이다. 하지만 암호기술에서는 실수를 이용할 수는 없다. 따라서 암호기술에서는 유한체 $\mathbb{Z}_p$에서 타원곡선 군을 정의하여 사용한다. 즉, 타원곡선 방정식에서 계수들이 모두 유한체 $\mathbb{Z}_p$의 원소가 되며, 각 연산이 유한체 $\mathbb{Z}_p$에서 이루어진다. 예를 들어 $p = 23$이고, 타원곡선 $y^2 = x^3 + x + 1$을 이용할 경우, 이 곡선을 통해 정의되는 모든 원소는 표 8.1과 같이 계산할 수 있다. 모든 원소를 구해보면 항등원을 포함하여 총 28개의 원소가 존재한다.

예8.2) $p = 23$, 타원곡선 $y^2 = x^3 + x + 1$의 두 점 (3,10)+(9,7)?

- $s = \frac{7-10}{9-3} = \frac{-3}{6} = 20 \bullet 4 = 80 \equiv 11 \pmod{23}$
- $x_R = 11^2 - 3 - 9 \equiv 17 \pmod{23}$
- $y_R = -10 + (11)(3-17) = -10 + 11 \bullet 9 \equiv 20 \pmod{23}$

여기서 $y_R = -(s \bullet x_R + y_0)$이지만 이 식을 이용하기 위해서는 y_0룰 먼저 계산해야 한다. 따라서 y_0를 계산하는 대신에 다음 식을 이용할 수도 있다.

$$y_R = -(s \bullet x_R + y_0) = -(s \bullet x_R + y_P - s \bullet x_P) = -(s(x_R - x_P) + y_P)$$

<표 8.1> $\mathbb{Z}_{23}$위의 타원곡선 $y^2 = x^3 + x + 1$의 모든 원소

x	$y^2 = x^3 + x + 1 \bmod 23$	이차잉여	y
0	1	yes	1, 22
1	3	yes	7, 16
2	11	no	
3	8	yes	10, 13
4	0	yes	0
5	16	yes	4, 19
6	16	yes	4, 19
⋮	⋮	⋮	⋮
22	22	no	

8.8 겹선형 쌍함수

겹선형 쌍함수(bilinear pairing)은 타원곡선 상의 이산대수 문제를 유한체상의 이산대수 문제로 축소시켜 그 어려움을 줄여 타원곡선 암호시스템을 공격하는 도구로 원래 제안되었다. 최근에는 공격 도구가 아닌 정보보호를 위한 암호학적 도구로 사용되고 있다. 겹선형 쌍함수는 다른 말로 겹선형 사상(bilinear map)이라 한다. 이 절에서는 다음과 같은 표기법을 사용하며, 이 map의 정의는 다음과 같다.

- q: 매우 큰 소수
- G_1: 위수가 q인 타원곡선 위의 덧셈군
- G_2: 위수가 q인 유한체 위의 곱셈군
- $P, Q, R \in_R G_1$
- $a, b, c \in_R \mathbb{Z}_q^*$

정의 8.8 (**사용가능 겹선형 사상(admissible bilinear map)**). 다음과 같은 특성을 만족하는 함수 $\hat{e}$: $G_1 \times G_1 \rightarrow G_2$를 **사용가능 겹선형 사상**이라 한다.

- **겹선형**(bilinear): 임의의 P, Q, R에 대하여 다음이 성립해야 한다.
  - $\hat{e}(P, Q+R) = \hat{e}(P,Q) \bullet \hat{e}(P,R)$
  - $\hat{e}(P+Q, R) = \hat{e}(P,R) \bullet \hat{e}(Q,R)$
- **비퇴화성**(non-degenerate): G_1의 모든 쌍 P, Q에 대하여, $\hat{e}(P,Q) \neq O$이어야 한다. 여기서 O는 G_1의 영원이다.
- **계산가능성**(computable): 임의의 P, Q에 대하여 $\hat{e}(P,Q)$를 계산할 수 있는 효율적인 알고리즘이 존재해야 한다.

겹선형 특성에 의해 다음이 성립한다.

$$\hat{e}(aP, bQ) = \hat{e}(P, bQ)^a = \hat{e}(aP, Q)^b = \hat{e}(P,Q)^{ab} = \hat{e}(abP, Q) = \hat{e}(P, abQ)$$

이 사상 때문에 타원곡선 상에서 Diffie-Hellman 결정 문제는 다음 식을 통해 쉽게 해결할 수 있다.

$$\hat{e}(aP, bP) = \hat{e}(cP, P) \Rightarrow ab = c$$

따라서 겹선형 사상을 암호학적 도구로 사용하는 많은 암호프로토콜에서는 다음 문제의 어려움에 기반하고 있다.

정의 8.9 (BDHP, Bilinear Diffie-Hellman Problem) G_1의 원소 P, aP, bP, cP가 주어졌을 때, $\hat{e}(P,P)^{abc}$를 계산하는 문제를 겹선형 Diffie-Hellman 계산 문제라 한다.

이 문제는 ECDLP 문제를 해결할 수 있으면 해결할 수 있다. 예를 들어 aP로부터 a를 계산할 수 있으면 $\hat{e}(bP, cP)^a$를 통해 $\hat{e}(P,P)^{abc}$를 계산할 수 있다. 뿐만 아니라 이 문제는 ECDHP 문제를 해결할 수 있으면 해결할 수 있다. 예를 들어 aP, bP로부터 abP를 계산할 수 있으면 $\hat{e}(abP, cP)$를 통해 $\hat{e}(P,P)^{abc}$를 계산할 수 있다. ECDLP와 ECDHP와 마찬가지로 이 문제는 현재 다항시간 내에 계산하는 것이 어렵다고 알려져 있다.

참고문헌

[1] 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.

[2] Whitfield Diffie and Martin E. Hellman, "New Directions in Cryptography," IEEE Trans. on Information Theory, Vol. 22, pp. 644-654, 1976.
[3] Adi Shamir, "Identity-based Cryptosystems and Signature Schemes," Advances in Cryptology, Crypto 1984, LNCS 196, pp. 47-53, 1985.
[4] 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. $\mathbb{Z}_5^*$의 이차잉여 1과 $\mathbb{Z}_7^*$의 이차잉여 2를 이용하여 $\mathbb{Z}_{35}^*$의 이차잉여를 구하시오.
2. $\mathbb{Z}_{11}^*$위의 타원곡선 $y^2 = x^3 + 2x + 1$의 모든 원소를 구하시오.
3. $\mathbb{Z}_{11}^*$위의 타원곡선 $y^2 = x^3 + 2x + 1$의 두 원소 (3,1)과 (5,2) 두 점의 합을 구하시오.
4. $\mathbb{Z}_{11}^*$위의 타원곡선 $y^2 = x^3 + 2x + 1$의 원소 (3,1)을 자기 자신과 합한 결과를 구하시오.