Elliptic Curve Cryptography

분산네트워크 연구실 32731 김희주
트러스트 메타버스 실현을 위한 블록체인 융합기술 과제

Public-key cryptography

- ECC (Elliptic Curve Cryptography)

기본적인 타원곡선 암호

- ECDH (Elliptic Curve Diffie-Hellman)

기존에 있던 Diffie-Hellman 방법에 타원곡선 적용

공개키가 영구적이면 ECDH, 반영구적이면 ECDHE

- ECDSA (Elliptic Curve Digital Signature Algorihm)

전자 서명 방식에 타원곡선 적용

대칭 키 공유를 위해 ECDH 사용

ECC 사용 범위

- TLS

표준 인터넷 프로토콜

- PGP(Pretty Good Privacy)

컴퓨터 파일 암복호화 프로그램, 이메일 보안 표준

- SSH(Secure Shell)

원격 호스트에 접속하기 위해 사용하는 보안 프로토콜

- Bitcoin

ECDSA 알고리즘 응용한 Secp256k1 암호 사용

Elliptic Curves

- The set of Points described by the equation

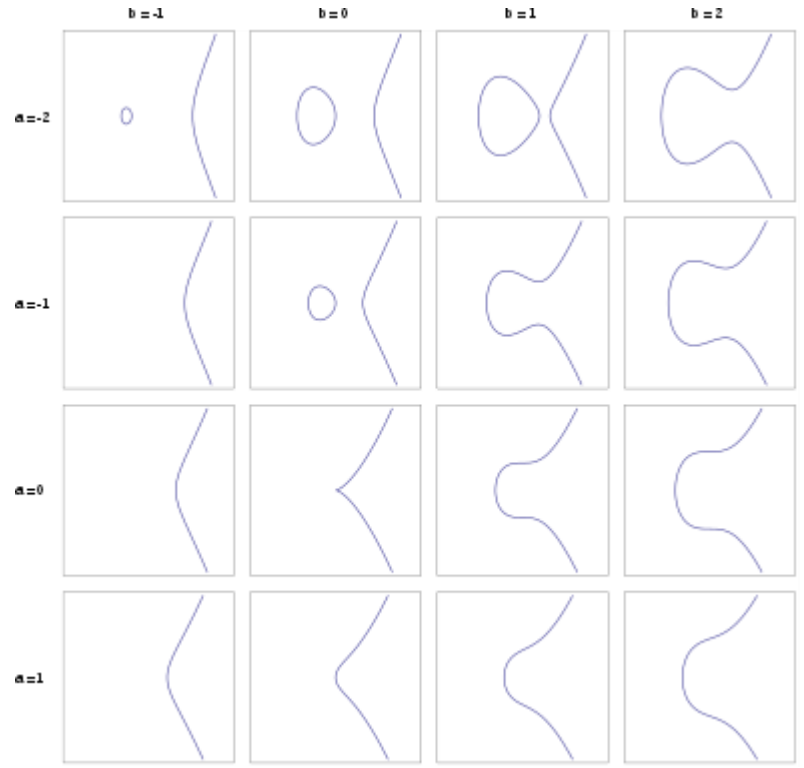
$$y^2 = x^3 + ax + b$$

a, b는 $-16(4a^3 + 27b^2) \neq 0$ 을 만족하는 상수

- Depending on the value a and b,

elliptic curves may assume different shapes

Symmetric about the x-axis



$$\{(x, y) \in \mathbb{R}^2 \mid y^2 = x^3 + ax + b,\ 4a^3 + 27b^2 \neq 0\} \cup \{0\}$$

Groups

- Group in mathematics is a set for “addition” (+)
- We must define addition in order to set group with four properties

1. **closure:** if a and b are members of $\mathbb{G}$, then $a + b$ is a member of $\mathbb{G}$;
2. **associativity:** $(a + b) + c = a + (b + c)$;
3. there exists an **identity element** 0 such that $a + 0 = 0 + a = a$;
4. every element has an **inverse**, that is: for every a there exists b such that $a + b = 0$.

- If we add a fifth requirement, the the group is called abelian group.

5. **commutativity:** $a + b = b + a$,

> 해당 정의에 따르면, 정수는 그룹이 될 수 있지만 자연수는 4번 조건을 만족하지 않기 때문에 그룹이 될 수 없음

The group law for elliptic curves

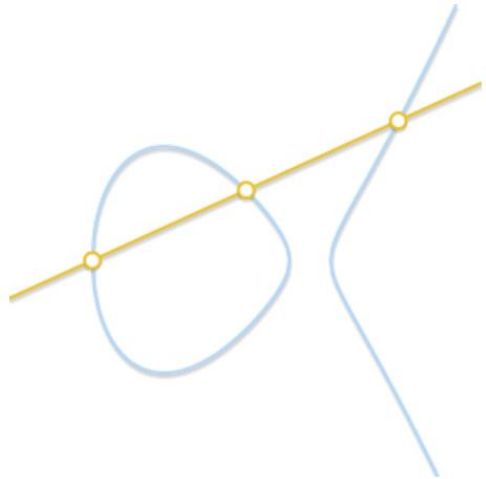
- We can define a group over elliptic curves
  - The element of the group are the points of an elliptic curve
  - The **identity element** is the point at infinity 0
  - The **inverse** of a point $\boldsymbol{P}$ is the one symmetric about the x-axis
  - **Addition** is given by the following rule: given three aligned, non-zero points P, Q and R, their sum is $\mathbf{P} + \mathbf{Q} + \mathbf{R} = \mathbf{0}$

- In this process, P, Q and R are aligned, then

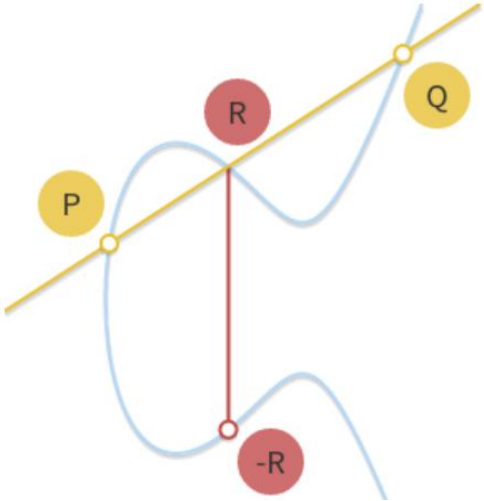
$$\mathbf{P} + (\mathbf{Q} + \mathbf{R}) = \mathbf{Q} + (\mathbf{P} + \mathbf{R}) = \ \ldots$$

  - ➢ Both Assosative and Commutative (결합법칙, 분배법칙)
    - ➢ Abelian group!

P, Q 가 타원 곡선 위 임의의 두 점일 경우, 두 점을 지나는 직선은 P, Q 어느쪽에서 그리던 상관 없기 때문에 교환법칙 성립



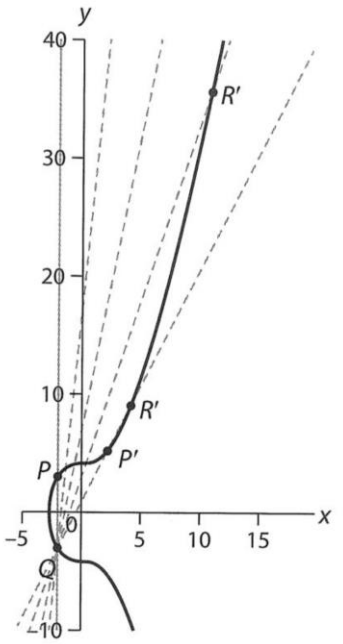
Geometric addition



- $\mathrm{P} + \mathrm{Q} + \mathrm{R} = 0$ means $P + Q = -\mathrm{R}$
- If we draw a line passing through P and Q, this line will intersect a third point on the curve, R
- If we take the inverse of this point, $-\mathrm{R}$, we have found the result of $\mathrm{P} + \mathrm{Q}$

- What if $P = 0$ or $Q = 0$?
  - We can't draw any line (0 is not on the xy-plane). But Given that we have defined 0 as the identity element, $P + 0 = P$ and $0 + Q = Q$, for any P and any Q

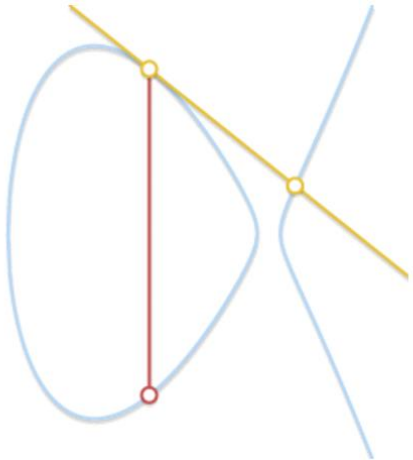
Geometric addition



- What if $P = -Q$?
  - The line going through the two points is vertical and does not intersect any third But P is the inverse of Q, then we have $P + Q = P + (-P) = 0$ from the definition inverse
  - P에 접근하는 P'가 존재하여 P'와 Q를 지나는 직선을 그린다고 생각해보면, P'가 P에 가까워질수록 제 3의 교점 R'가 점점 원점에서 멀어지는 것을 확인 가능
    - 해당 경우를 무한 원점이라고 칭하고, ∞ 로 표기
    - 반대로 두 점 P와 ∞를 지나는 직선은 점 Q에서 타원 곡선과 만나고, Q의 대칭 점은 P이므로 결론적으로 $P+\infty= P$
    - 이처럼 ∞는 덧셈에 대한 항등원 역할(0과 같은 역할)을 수행, 반면 P의 덧셈에 대한 역원은 y축과 평행한 직선과 타원 곡선의 교점인 Q

Geometric addition

- What if $P = Q$?
  - $P + P = -R$, where R is the point of intersection between the curve and the line tangent to the curve in P
  - What if $P \neq Q$, but there is no third point R?
    - We have to find the line passing through P and Q is tangent to the curve
    - If P is the tangency point, then $P + Q = -Q$



If line intersects just two points,
it means that it's tangent to the curve

Scalar multiplication

- Other than addition, we can define another operation
  - Scalar multiplication

$$nP = \underbrace{P + P + \cdots + P}_{n \text{ times}}$$

  - n is a natural number
  - It may seem that computing nP requires n additions (Bad algorithm)

$$\begin{aligned} 151 &= 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 2^7 + 2^4 + 2^2 + 2^1 + 2^0 \end{aligned}$$

  - But there exist faster algorithms

Scalar multiplication – Double and add

- Double and add

$$151 \cdot P = 2^7P + 2^4P + 2^2P + 2^1P + 2^0P$$

What the double and add algorithm tells us to do is:

- Take P.
- *Double* it, so that we get $2P$.
- *Add* $2P$ to P (in order to get the result of $2^1P + 2^0P$).
- *Double* $2P$, so that we get 2^2P.
- *Add* it to our result (so that we get $2^2P + 2^1P + 2^0P$).
- *Double* 2^2P to get 2^3P.
- Don't perform any addition involving 2^3P.
- *Double* 2^3P to get 2^4P.
- *Add* it to our result (so that we get $2^4P + 2^2P + 2^1P + 2^0P$).
- ...

- In the end, we can compute $151 \cdot P$ performing just seven doublings and four additions

Logarithm(이산대수 문제)

- Given n and P, we have algorithm for computing $Q = nP$
- What if we know Q and P and need to find out n?
  - Logarithm problem!
- Playing with multiplication it's easy to see some patterns.
  - If n is odd, $n\mathrm{P}$ is on the curve on the left semiplane
  - If n is even, $n\mathrm{P}$ is on the curve on the right semiplane
  - If we experimented more, we could find more patterns that eventually could lead us to write an algorithm for computing the logarithm on that curve efficiently

출처: https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/

The field of integers modulo p

- Set of integers modulo p, where p is a prime number $\mathbb{F}_p$
- In field, we have two binary operation
  - Addition (+) and Multiplication (*)
  - Multiplication is distributive over the addition
    - $x \cdot (y + z) = x \cdot y + x \cdot z$
- Set of integers modulo p consists of all the integers from 0 to $p - 1$

The field of integers modulo p

- Addition and multiplication work as in modular arithmetic

- Addition: $(18 + 9) \bmod 23 = 4$
- Subtraction: $(7 - 14) \bmod 23 = 16$
- Multiplication: $4 \cdot 7 \bmod 23 = 5$
- Multiplicative inverse: $9^{-1} \bmod 23 = 18$

  Indeed: $9 \cdot 9^{-1} \bmod 23 = 9 \cdot 18 \bmod 23 = 1$

- Additive inverse: $-5 \bmod 23 = 18$

  Indeed: $(5 + (-5)) \bmod 23 = (5 + 18) \bmod 23 = 0$

- p must be prime number
  - If p is not a prime number, there has no multiplicative inverse
    - i.e. the set of integers modulo 4 is not a field
      - 2 has no multiplicative inverse
      - $2 \cdot x \; mod \; 4 = 1$ has no solutions

Elliptic curves in $\mathbb{F}_p$

- Previous version

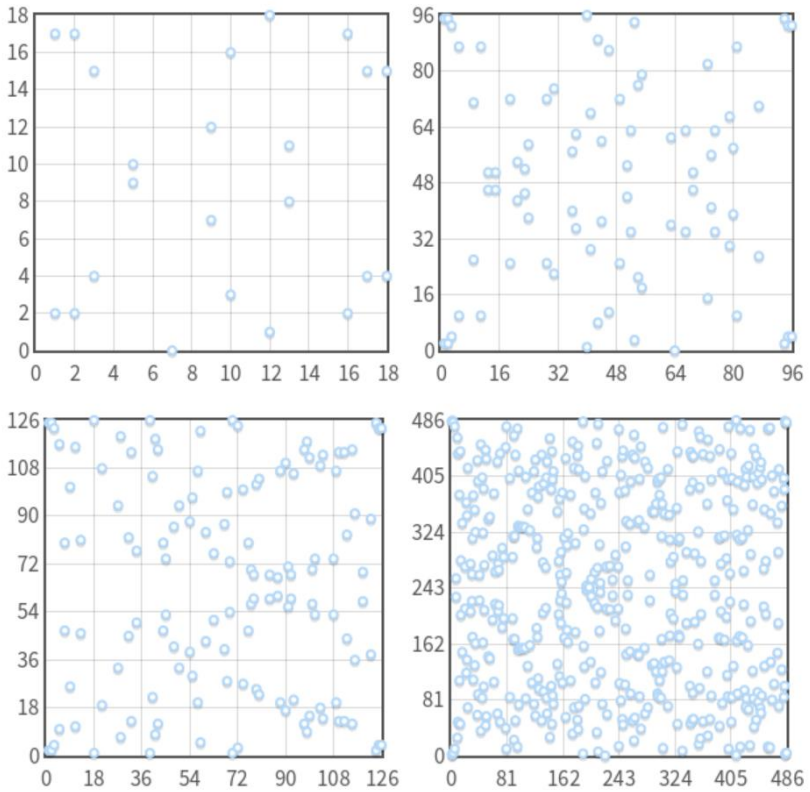
$$\{(x, y) \in \mathbb{R}^2 \quad | \quad y^2 = x^3 + ax + b, \quad 4a^3 + 27b^2 \neq 0\} \cup \{0\}$$

- $\mathbb{F}_p$ version

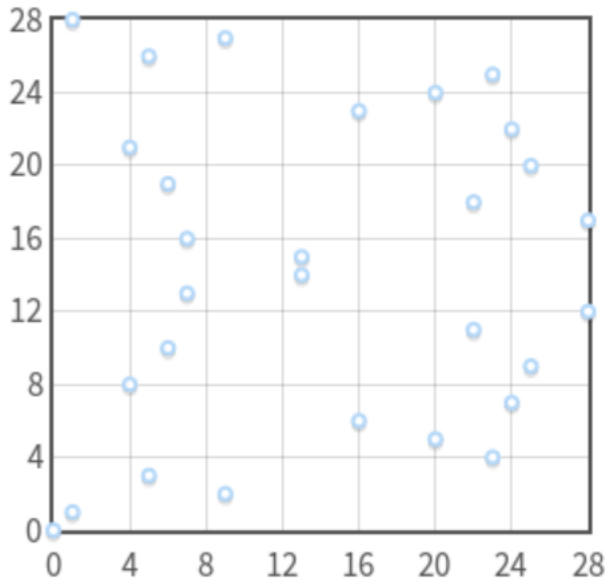
$$\{(x, y) \in (\mathbb{F}_p)^2 \quad | \quad y^2 \equiv x^3 + ax + b \pmod{p}, \quad 4a^3 + 27b^2 \not\equiv 0 \pmod{p}\} \cup \{0\}$$

  - Where 0 is still the point at infinity, and a and b are two integers in $\mathbb{F}_p$

Elliptic curves in $\mathbb{F}_p$



The curve $y^2 \equiv x^3 - 7x + 10 \pmod{p}$ with $p = 19, 97, 127, 487$. Note that, for every x, there are at most two points. Also note the symmetry about $y = p/2$.

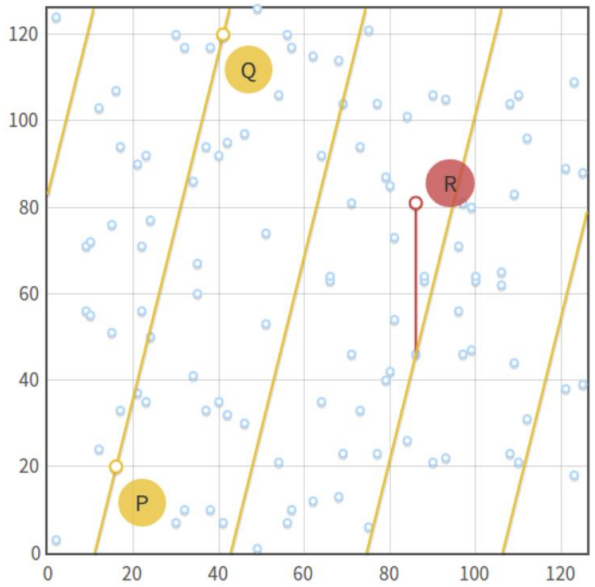


The curve $y^2 \equiv x^3 \pmod{29}$ is singular and has a triple point in $(0, 0)$. It is not a valid elliptic curve.

Point addition in $\mathbb{F}_p$

- Three points are aligned if there's a line that connects all of them ($P + Q + R = 0$)
  - Lines are different in R
  - line in $\mathbb{F}_p$ is the set of points (x, y) that satisfy the equation

$$ax + by + c \equiv 0 \pmod{p}$$



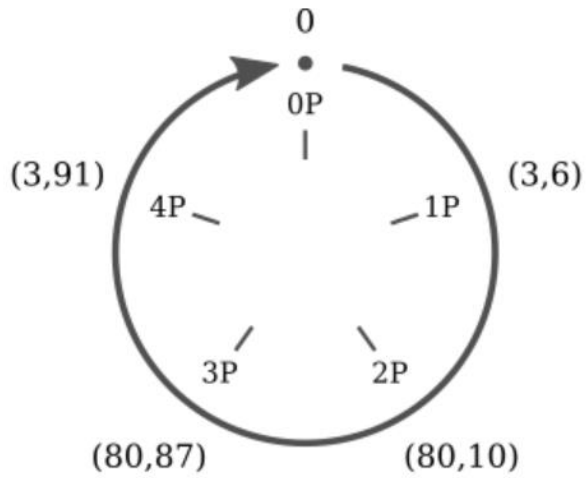
Point addition over the curve $y^2 \equiv x^3 - x + 3 \pmod{127}$, with $P = (16, 20)$ and $Q = (41, 120)$. Note how the line $y \equiv 4x + 83 \pmod{127}$ that connects the points "repeats" itself in the plane.

Given that we are in a group, point addition retains the properties we already know:

- $Q + 0 = 0 + Q = Q$ (from the definition of identity element).
- Given a non-zero point Q, the inverse $-Q$ is the point having the same abscissa but opposite ordinate. Or, if you prefer, $-Q = (x_Q, -y_Q \bmod p)$. For example, if a curve in $\mathbb{F}_{29}$ has a point $Q = (2, 5)$, the inverse is $-Q = (2, -5 \bmod 29) = (2, 24)$.
- Also, $P + (-P) = 0$ (from the definition of inverse element).

Scalar multiplication and cyclic subgroups

Multiplication over points for elliptic curves in $\mathbb{F}_p$ has an interesting property. Take the curve $y^2 \equiv x^3 + 2x + 3 \pmod{97}$ and the point $P = (3, 6)$. Now calculate all the multiples of P:



The multiples of $P = (3, 6)$ are just five distinct points (0, P, $2P$, $3P$, $4P$) and they are repeating cyclically. It's easy to spot the similarity between scalar multiplication on elliptic curves and addition in modular arithmetic.

- $0P = 0$
- $1P = (3, 6)$
- $2P = (80, 10)$
- $3P = (80, 87)$
- $4P = (3, 91)$
- $5P = 0$
- $6P = (3, 6)$
- $7P = (80, 10)$
- $8P = (80, 87)$
- $9P = (3, 91)$
- ...

Scalar multiplication and cyclic subgroups

Here we can immediately spot two things: firstly, the multiples of P are just five: the other points of the elliptic curve never appear. Secondly, they are **repeating cyclically**. We can write:

- $5kP = 0$
- $(5k+1)P = P$
- $(5k+2)P = 2P$
- $(5k+3)P = 3P$
- $(5k+4)P = 4P$

for every integer k. Note that these five equations can be "compressed" into a single one, thanks to the modulo operator: $kP = (k \bmod 5)P$.

Not only that, but we can immediately verify that **these five points are closed under addition**. Which means: however I add 0, P, $2P$, $3P$ or $4P$, the result is always one of these five points. Again, the other points of the elliptic curve never appear in the results.

The same holds for every point, not just for $P = (3, 6)$. In fact, if we take a generic P:

$$nP + mP = \underbrace{P + \cdots + P}_{n \text{ times}} + \underbrace{P + \cdots + P}_{m \text{ times}} = (n+m)P$$

Scalar multiplication and cyclic subgroups

$$nP + mP = \underbrace{P + \cdots + P}_{n \text{ times}} + \underbrace{P + \cdots + P}_{m \text{ times}} = (n + m)P$$

Which means: **if we add two multiples of P, we obtain a multiple of P** (i.e. multiples of P are closed under addition). This is enough to prove that **the set of the multiples of P is a cyclic subgroup** of the group formed by the elliptic curve.

A “subgroup” is a group which is a subset of another group. A “cyclic subgroup” is a subgroup which elements are repeating cyclically, like we have shown in the previous example. The point P is called **generator** or **base point** of the cyclic subgroup.

Subgroup order

- What the order of a subgroup generated by a point P
  - Cyclic subgroup we can give a new definition: the order of P is the smallest positive integer n such that $\mathrm{n}P = 0$
  - The order of P is linked to the order of the elliptic curve by Lagrange's theorem, which states that the order of a subgroup is a divisior of the order of the parent group
- These two information together give us a way to find out the order of a subgroup with base point P

1. Calculate the elliptic curve's order N using Schoof's algorithm.
2. Find out all the divisors of N.
3. For every divisor n of N, compute nP.
4. The smallest n such that $nP = 0$ is the order of the subgroup.

Domain Parameters

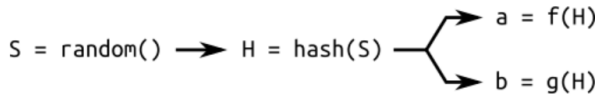
- Elliptic curve algorithms will work in a cyclic subgroup of an elliptic curve over a finite field
- Algorithm will need the parameter
  - The prime p that specifies the size of the finite field
  - The coefficients a and b of the elliptic curve equation
  - The base point G that generates our subgroup
  - The order n of the subgroup
  - The cofactor h of the subgroup
- Domain parameters for our algorithms are the sextuple (p, a, b, G, n, h)

Random curves

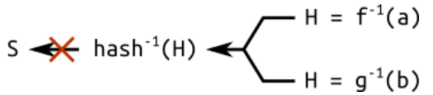
- Some classes of elliptic curves that are particularly weak
  - For example, All the curves that have $\boldsymbol{p = hn}$ (the order of the finite field is equal to the order of the elliptic curve) that are vulnerable to Smart's attack, which can be used to solve discrete logarithms in polynomial time on a classical computer
- How can we assure that the curve is safe?
  - Use additional domain parameter: the seed S
  - S is a random number used to generate the coefficients a and b, or the base point G, or both

Random curves

- a, b, G are generated by computing the hash of the seed S
- Hashes are “easy” to compute, but “hard” to reverse



A simple sketch of how a random curve is generated from a seed: the hash of a random number is used to calculate different parameters of the curve.



If we wanted to cheat and try to construct a seed from the domain parameters, we would have to solve a "hard" problem: hash inversion.

- A curve generated through a seed is said to be verifiably random
  - Known as “nothing up my sleeve”
- The curve has not been specially crafted to expose vulnerabilities known to the author

Elliptic Curve Cryptography

1. The private key is a random integer d chosen from $\{1, \ldots, n-1\}$ (where n is the order of the subgroup)
2. The public key is the point $H = dG$ (where G is the base point of the subgroup)

- If we know d and G(along with the other domain parameters), finding H is "easy"
- But if we know H and G, finding the private key d is "hard"
  - Discrete logarithm problem

Encryption with ECDH

- ECDH is a variant of the Diffie-Hellman algorithm for elliptic curve
- Key agreement protocol, more than an encryption algorithm
- ECDH defines how keys should be generated and exchanged between parties
- The problem it solves is the following: two parties want to exchange information securely, so that a third party (the Man In the Middle) may intercept them but may not decode them.

Encryption with ECDH

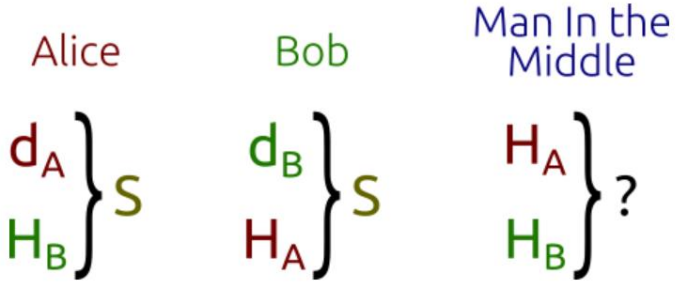
1. Alice and Bob generate their own private and public keys
   - Alice: d_A for private key, $H_A = d_A G$ for public key
   - Bob: d_B for private key, $H_B = d_B G$ for public key
   - Both are using the same domain parameters: the same base point G on the same elliptic curve on the same finite field
2. Alice and Bob exchange their public keys H_A and H_B over an insecure channel
   - Man In the Middle would intercept H_A and H_B, but won't be able to find out nither d_A nor d_B without solving the discrete logarithm problem

Encryption with ECDH

3. Alice and Bob calculate the shared secret S
   - Alice calculates $S = d_A H_B$(using her own private key and Bob's public key)
   - Bob calculates $S = d_B H_A$(using his own private key and Alice's public key)

$$S = d_A H_B = d_A(d_B G) = d_B(d_A G) = d_B H_A$$

- This is known as the Diffie-Hellman problem, which can be started as follow
  - Given three points P, aP and bP, what is the result of abP
  - Given three integers k, k^x and k^y, what is the result of k^{xy}?



The Diffie-Hellman key exchange: Alice and Bob can "easily" calculate the shared secret, the Man in the Middle has to solve a "hard" problem

Ephemeral ECDH

- ECDHE
  - “E” in ECDHE stands for “Ephemeral(일시적인)” and refers to the fact that the keys exchanged are temporary
  - Used in TLS, where both the client and the server generate their public-private key pair on the fly
  - Keys are signed with the TLS certificate (for authentication) and exchanged between the parties

ECDSA

- Alice : select Private key d_A, calculate $H_A = d_A G$
  1. Calculate $e = HASH(m)$
  2. z는 e의 왼쪽 L_n 비트
  3. Take cryptographically secure random integer k from {1, …, n-1}
  4. Calculate $(x_1, y_1) = \mathrm{k}G$
  5. $r = x_1 \, mod \, n$, If $r = 0$, go back to step 3
  6. $s = k^{-1}(z + rd_A) \, mod \, n$, if $s = 0$, go back to step 3
  7. Signature : (r, s)

ECDSA

- Bob : verification algorithm
  1. 다음 3개를 확인 $H_A \neq 0$, H_A on the curve, $n * H_A = 0$
  2. r, s 가 [1, n-1] 범위에 있으면 정상, 아니면 비정상
  3. Calculate $e = HASH(m)$
  4. z는 e의 왼쪽 L_n 비트
  5. Calculate $w = s^{-1} \, mod \, n$
  6. $u_1 = z \, w \, mod \, n, u_2 = r \, w \, mod \, n$
  7. $(x_1, y_1) = u_1 * G + u_2 * H_A$, (x_1, y_1) = 0 면 invalid
  8. The signature is valid only if $r = x_1 \, mod \, n$

ECDSA

- 다른 메시지를 사인한 두 명의 k가 같은 경우 위험성
  1. 사인은 각각 (r_1, s_1), (r_2, s_2), 두 메시지의 hash가 각각 (z_1, z_2)라면
  2. K가 같다면 $r_1 = r_2$
  3. $(s_1 - s_2) \bmod n = k^{-1}(z_1 - z_2) \bmod n$
  4. $k(s_1 - s_2) \bmod n = (z_1 - z_2) \bmod n$
  5. $k = (z_1 - z_2)(s_1 - s_2)^{-1} \bmod n$
  6. $s = k^{-1}(z + rd_A) \bmod n \Rightarrow d_A = r^{-1}(sk - z)$

Schnorr signature

- Schnorr signature
  - A cyclic group G of prime order p
  - A generator g of G
  - A hash function H
  - Private/public key pair $(x, X) \in \{0, p-1\} \times G \; where \; X = g^x$
- Signing message m
  - Select r in Z_p
  - Compute $R = g^r$, $c = H(X, R, m), s = r + cx$
  - Signature : (R, s)
  - Verify : $g^r = RX^c$ $(= g^r X^c = g^r g^{xc} = g^r g^{x(s-r)/x} = g^s)$

Schnorr signature

- Naive way Multi-sign : n signer with a message m
  - $L = \{X_1 = g^{x_1}, \dots, X_n = g^{x_n}\}$ set of public key
  - $R = \prod_{i=1}^{n} R_i \quad : R_i = g^{r_i}$
  - $c = H(\tilde{X}, R, m)$: $\tilde{X} = \prod_{i=1}^{n} X_i$
  - $s = \prod_{i=1}^{n} s_i \ mod\ p \quad : s_i = r_i + cx_i$
  - Signature : (R, s)
  - Verify : $g^s = R\tilde{X}^c$
- Rogue-key attack에 취약

EC-Schnorr signature

- Key generation $(sk, pk) \in \{1, n-1\}, where\ pk = sk * G$
- Signing message m
  - Select k in $\{1, n-1\}$
  - $Q = kG$
  - $r = H(Q||pk||m)\ mod\ n$
  - $s = k - r * sk\ mod\ n$
- Verify
  - $Q = sG + r * pk = (k - r * sk)G + r * sk * G$
  - $v = H(Q||pk||m)\ mod\ n$
  - If $v = \mathrm{r}$ return 1, else 0

EC-Schnorr multi-signature

- Key generation $(sk, pk) \in \{1, n-1\}, where\ pk = sk * G$
- Signing message m
  - Select k_i in $\{1, p-1\}$
  - $Q = k_i G$
  - $Pk = \prod_{i=1}^{n} pk_i$
  - $Q = \prod_{i=1}^{n} Q_i$
  - $r = H(Q||pk||m)\ mod\ p$
  - $s_i = k_i - r\ * sk_i\ mod\ p$
  - $S = \sum_{i=1}^{n} S_i$
  - Signature: (r, S)

EC-Schnorr multi-signature

- Verify
  - $Q = SG + r * Pk$
  - $v = H(Q||Pk||m)\ mod\ p$
  - If $v = \mathrm{r}$ return 1, else 0

Thank you :)