Cracking the Elliptic Curve Cryptosystem

Chun Min Tan (CID: 02016068)
cmt122@ic.ac.uk

Introduction to Elliptic Curve

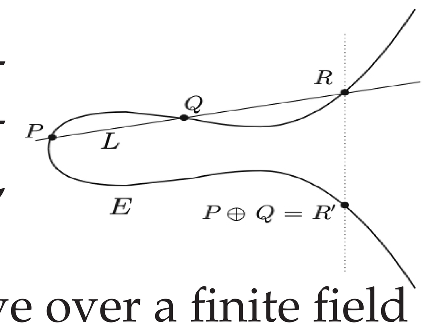
Definition 1. *(Elliptic Curve) An elliptic curve E is the set of solutions to a Weierstrass equation*

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

with point $\mathcal{O}$ and a, b satisfying $4a^3 + 27b^2 \neq 0$

Elliptic Curve Addition Algorithm

Let E be an elliptic curve, and suppose P, Q are points on E. Define $P \oplus Q = R'$, as in the figure [1, p.281].



In cryptography we use elliptic curve over a finite field

$$E(\mathbb{F}_p) = \{(x, y) : x, y \in \mathbb{F}_p \wedge (x, y) \in E\} \cup \{\mathcal{O}\}$$

Elliptic Curve Cryptography

Some notable elliptic curve cryptosystems are Diffie-Hellman key exchange and ElGamal public key, etc. A summary of Diffie-Hellman key exhcange [1, p.297]

Public parameter creation	
A trusted party chooses and publishes a (large) prime p, an elliptic curve E over $\mathbb{F}_p$, and a point P in $E(\mathbb{F}_p)$.	
Private computations	
Alice	**Bob**
Chooses a secret integer n_A. Computes the point $Q_A = n_A P$.	Chooses a secret integer n_B. Computes the point $Q_B = n_B P$.
Public exchange of values	
Alice sends Q_A to Bob $\longrightarrow$	Q_A
Q_B $\longleftarrow$	Bob sends Q_B to Alice
Further private computations	
Alice	**Bob**
Computes the point $n_A Q_B$.	Computes the point $n_B Q_A$.
The shared secret value is $n_A Q_B = n_A(n_B P) = n_B(n_A P) = n_B Q_A$.	

Note that if one can solve $Q_A = n_A P$ or $Q_B = n_B P$, then one is able to crack the cipher, i.e. the elliptic curve discrete logarithm problem (ECDLP).

Double-and-Add Algorithm

1. We write n in binary form as
$$n = n_0 + n_1 \cdot 2 + n_2 \cdot 2^2 + \cdots n_r \cdot 2^r \qquad n_i \in \{0, 1\}$$
2. Compute $Q_i = 2Q_{i-1} = 2^i P$ for $i \geq 0$. Then
$$nP = n_0 Q_0 + n_1 Q_1 + \cdots + n_r Q_r \qquad n_i \in \{0, 1\}$$

$r = \lfloor \log_2 n \rfloor \leq \log_2 n, \therefore$ at most $\boxed{2 \log_2 n}$ steps and on average it takes $\boxed{3/2 \log_2 n}$.

Improvements: We allow coefficients $n_i \in \{-1, 0, 1\}$ and an extra digit in the expansion.

Proposition 1. *For all $n \in \mathbb{N}$, there exists a ternary expansion where at most half of the coefficients are nonzero.*

Proof. We look for the first two or more consecutive nonzero u_i in binary expansion. Suppose we have

$$u_s = u_{s+1} = \cdots = u_{s+t-1} = 1 \quad \text{and} \quad u_{s+t} = 0$$

where $t \geq 2$. Then we have

$$2^s + 2^{s+1} + \cdots + 2^{s+t-1} + 0 \cdot 2^{s+t} = -2^s + 2^{s+t}$$

□

There are at most $\lfloor \log_2 n \rfloor + 1$ doublings and at most $\lfloor (\lfloor \log_2 n \rfloor + 1)/2 \rfloor + 1$ additions, added together gives $\boxed{3/2 \log_2 n + 5/2}$ and on average it takes $\boxed{4/3 \log_2 n + 7/3}$.

Naive Collision Algorithm

Theorem 1. *(Collision Theorem) An urn contains N balls, n are red, $N - n$ are blue. Bob chooses m balls with replacement, then if X is the number of red ball observe, we have $P(X \geq 1) \geq 1 - e^{-mn/N}$*

Proof. Note that $P(X \geq 1) = 1 - P(X = 0)$, note that $P(X = 0) = ((N - n)/m)^m = (1 - n/N)^m \leq e^{-mn/N}$ □

1. If $N = \text{ord}(P)$, choose $r \approx 3\sqrt{N}$. We randomly choose $1 \leq y_1, \ldots, y_r \leq N$ and compute
$$y_1 P, y_2 P, \ldots, y_r P \in \langle P \rangle \subseteq E(\mathbb{F}_p) \qquad \text{in at most } 2r \log_2 N \text{ steps}$$
2. We randomly select $1 \leq z_1, \ldots, z_r \leq N$ and compute
$$z_1 P + Q, z_2 P + Q, \ldots, z_r P + Q \in \langle P \rangle \subseteq E(\mathbb{F}_p) \qquad \text{in at most } 2r \log_2 N + r \text{ steps}$$
3. Search for a collision, $y_\alpha P = z_\beta P + Q \implies Q = (y_\alpha - z_\beta)P$, hence the solution is $y_\alpha - z_\beta \pmod N$. Merge sort plus binary search combined requires on average $2r \log_2 r$ steps.

How likely is a collision? Treat $\langle P \rangle$ as the urn, y_i as the red balls. Pick r balls $(z_i P + Q)$, then by Collision theorem

$$P(\text{at least one collision}) = 1 - (1 - r/N)^r \geq 1 - e^{-r^2/N} \approx 1 - e^{-9} \approx 99.98\%$$

with total number of steps $2r \log_2 N + 2r \log_2 N + r + 2r \log_2 r = 3\sqrt{N} \log_2(N^4 \cdot 9N) + 3\sqrt{N} = \boxed{O(\sqrt{N} \log N)}$

Key Idea: Generate 2 lists $y_i P$ and $z_i P + Q$ by randomly selecting y_i and z_i from $[1, N]$ and search for a collision.

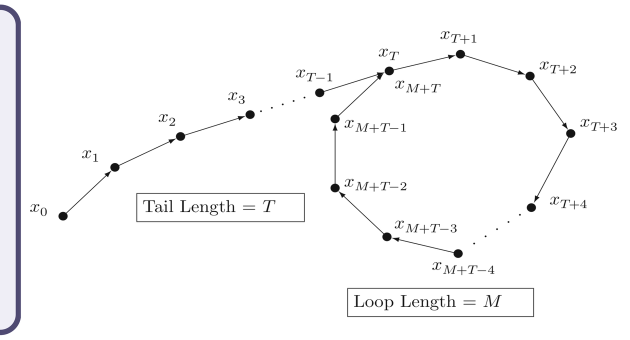
Pollard's ρ method for $ord(P) = p$ prime

Theorem 2. *Set $|S| = p$, $f : S \to S$ is sufficiently random, if (x_i) is generated by applying f, then $\mathrm{E}(T + M) = \sqrt{\pi p/2}$.*

Proof. One can show that for large p and T, M as in the figure provided [1, p.235]

$$P\left({x_0, x_1, \ldots, x_{k-1} \atop \text{are all different}}\right) = \prod_{i=1}^{k-1} P\left({x_i \neq x_j \text{ for} \atop \text{all } 0 \leq j < i} \middle| {x_0, x_1, \ldots, x_{i-1} \atop \text{are all different}}\right) = \prod_{i=1}^{k-1} (1 - i/p) \approx e^{-k^2/2p}$$

then $E(T + M) = \sum_{k=1}^{\infty} kP(x_k \text{ is first match}) \approx \sum_{k=1}^{\infty} (k^2/p) e^{-k^2/2p} = \sqrt{\pi p/2}$ □



1. Let $\{S_1, \ldots, S_L\}$ be a partition of $\langle P \rangle$. Now define $H(X) = j$ if $X \in S_j$ and let $a_j, b_j \in_R [0, p-1]$ for each $1 \leq j \leq L$. Then define
$$f : \langle P \rangle \to \langle P \rangle$$
$$X \mapsto X + a_j P + b_j Q \qquad \text{where } j = H(X)$$
2. Since $\langle P \rangle$ is finite, by ***Floyd's cycle-detection algorithm***, compute (X_i, X_{2i}) until $X_i = X_{2i}$, where $X_i = f(X_{i-1})$.

How fast is this algorithm? Since $\mathrm{E}(T + M) = \sqrt{p\pi/2}$, and in each evaluation of the function f, it only requires 2 modular addition, hence it takes approximately $2 \cdot \sqrt{p\pi/2}$ steps, therefore $\boxed{O(\sqrt{p\pi/2})}$ or $\boxed{O(\sqrt{p})}$.

Key Idea: Construct f to be sufficiently random, then compute (X_i, X_{2i}) until $X_i = X_{2i}$ where $X_i = f(X_{i-1})$.

Pohlig-Hellman Algorithm

Let $N = \text{ord}(P) = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ be given. To solve $\tilde{n} : Q = \tilde{n}P$, consider $\tilde{n} \equiv n_i \pmod{p_i^{e_i}}$ for each $1 \leq i \leq r$.

1. To find $n_i \pmod{p_i^{e_i}}$, we write it in its p_i-adic expansion modulo $p_i^{e_i}$. This will give
$$n_i \equiv z_0 + z_1 p_i + \cdots + z_{e_i - 1} p_i^{e_i - 1} \pmod{p_i^{e_i}} \qquad \text{where } z_i \in [0, p_i - 1]$$
2. Define $P_0 = (N/p_i)P$ and $Q_0 = (N/p_i)Q$, since P_0 has order p_i, we have
$$Q_0 = (N/p_i)Q = (N/p_i)\tilde{n}P = \tilde{n}(N/p_i \cdot P) = \tilde{n}P_0 = [\tilde{n}]_{p_i} P_0 = z_0 P_0$$
therefore $z_0 = \log_{P_0} Q_0$ which can be solved using Pollard's ρ method.
3. If $z_0, \ldots, z_{t-1}$ have been computed, then $z_t = \log_{P_0} Q_t$ can be computed using Pollard's ρ method as well where
$$Q_t = \frac{N}{p_i^{t+1}}(Q - z_0 P - z_1 p_i P - \cdots - z_{t-1} p_i^{t-1})$$
4. This allow us to compute $z_0, \ldots, z_{e_i - 1}$. Repeat for all n_j, then $\tilde{n}$ can be found using ***Chinese Remainder Theorem***.

How fast is this algorithm? Given $N = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$, then Pohlig-Hellman Algorithm takes $\boxed{O\left(\sum_{i=1}^{r} e_i (\log n + \sqrt{p_i})\right)}$

Key Idea: Solve for $\tilde{n} \equiv n \pmod{p_i^{e_i}}$ by solving $Q_t = z_t P$ and combine each n_i using Chinese Remainder Theorem.

General Attack on ECDLP

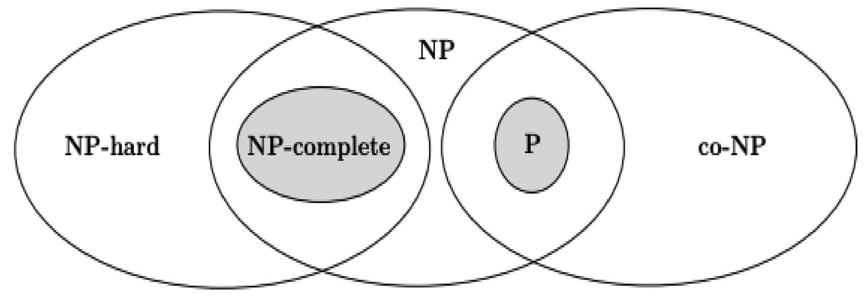
The best general attack is a combination of Pollard's ρ algorithm for factorisation and the Pohlig-Hellman Algorithm. The expected running time of this is

$$\boxed{O(\sqrt{p})}$$

where p is the largest prime divisor of $N = \text{ord}(P)$.

☞ To resist this attack one should pick an elliptic curve with point P of order N where all the prime divisors of N are large[2, §4.1].

Million Dollar Question(?)



Complexity Classes Venn Diagram [3]

Definition 2. *(Decision version ECDLP) Given $(E(\mathbb{F}_p), N, d, Q, P)$ where $E(\mathbb{F}_p)$ is an elliptic curve over $\mathbb{F}_p$, $P \in E(\mathbb{F}_p)$ with order N and $d \leq N$ an integer. Then the decision problem is:*

Is there an integer $k \leq d : Q = kP$?

The decision version of ECDLP is known to be in NP ∩ co-NP. It is not known to be in P as currently there is no deterministic polynomial time algorithm that solves it. **Therefore if one can show that there does not exist a deterministic polynomial time algorithm that solves ECDLP, this would imply**

$$\text{P} \neq \text{NP}$$

thus settling one of the most important question in computer science. Further if one can show that it is NP-complete, then this would imply that

$$\text{NP} = \text{co-NP}$$

which also solves another unsolved question in computer science[2, §4.1].

Conclusions

The underlying working principle of ECDLP is based on the assumption that P ≠ NP. The algorithms covered here involve some probabilistic elements and the fastest known algorithm that can solve ECDLP in polynomial time can only be done on a non-deterministc Turing machine. But all of this may change when the age of quantum computing truly begins.

References

[1] Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman, and Joseph H Silverman. *An introduction to mathematical cryptography*, volume 1. Springer, 2008.

[2] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[3] Anton Pierre De Villiers. *Edge criticality in secure graph domination*. PhD thesis, Stellenbosch: Stellenbosch University, 2014.