Elliptic Curve Cryptography

Avi Bagchi

Mentor: Andrew Kwon

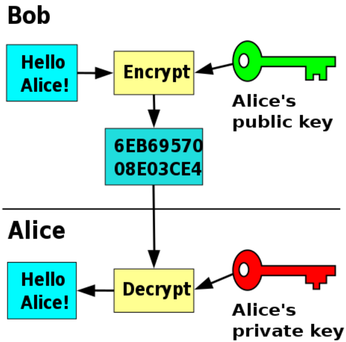
How do we send a secure message?

- Goal: Encrypt plaintext P into C
- Desired Properties
  - $\exists$ encryption function E
  - $\exists$ decryption function D
  - $P = D(E(P))$

Public-Key Cryptography

- E is public
  - E should not imply D
  - Authentication: $E(D(M)) = M$

Public-Key Cryptography

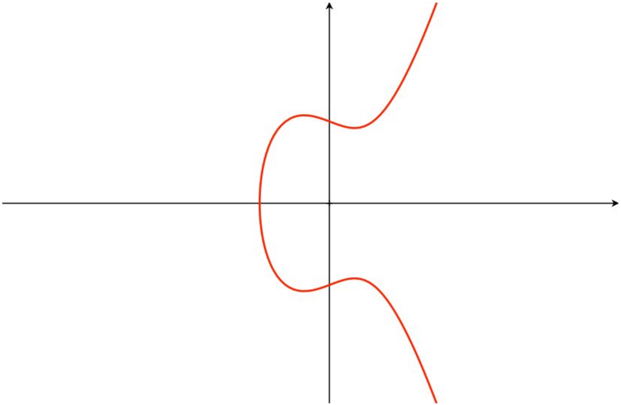


Elliptic Curve Cryptography

- Efficient alternative to RSA.
- Bitcoin

Elliptic Curves

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



Group Structure

Elliptic curves naturally form group structure

- Identity element
- Associative operation
- Every element has inverse

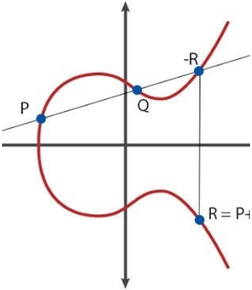
Identity Element

Point at infinity $\mathbf{0}$

- $P \oplus \mathbf{0} = P$

Operation

- Given P_1, P_2 find $P_3 = P_1 \oplus P_2$
- $\oplus$:
  - $*$: Draw line through P, Q and find third point $-R$
  - Apply $-R * \mathbf{0}$ to find R
    - "Reflecting over x-axis"



Discrete Log Problem

- Given points on elliptic curve P_1, P_2
- To find P_2 from P_1, how many times do we apply $\oplus$?
- Finding k such that $kP_1 = P_2$ is hard

Discrete Log Problem

- Base point P_1
- Public Key: $P_2 = kP_1$
- Private Key: Some $k \in Z$

Attacks

Can the discrete log problem be solved efficiently?

Pollard's Rho Algorithm

- Idea: Starting with two points, find two distinct paths that yield the same third point
- Formally, find $c^{'}P + d^{'}Q = c^{''}P + d^{''}Q$ such that $c^{'} \neq c^{''}, d^{'} \neq d^{''}$
- If we find $c^{'}, c^{''}, d^{'}, d^{''}$, then:
  - $(c^{'} - c^{''})P = (d^{''} - d^{'})Q = (d^{''} - d^{'})kP$
  - $(c^{'} - c^{''}) = (d^{''} - d^{'})k$
  - $k = (c^{'} - c^{''})(d^{''} - d^{'})^{-1}$

Pollard's Rho Algorithm

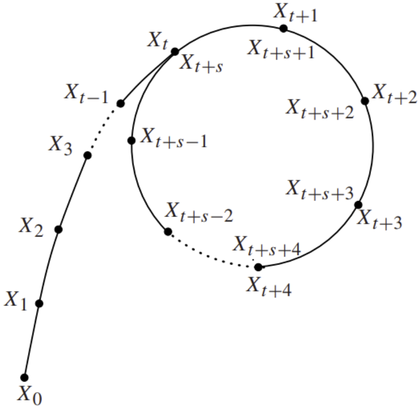
How do we find $c^{'}, c^{''}, d^{'}, d^{''}$?

- Naiive: Random generation, storing all past operations
- Pollard's: Pseudo-random, space efficient

Pollard's Rho Algorithm

- Define f as the doubling operation
- If $X = cP + dQ$, we can get the next point $X^{'} = F(X)$ with new coefficients $c^{'}, d^{'}$
- $f(X) = X + a_j P + b_j Q$ where $c^{'} = c + a_j$ and $d^{'} = d + b_j$
- Divide curve into subsets $S_i ... S_L$ where some subset S_j has associated coefficients a_i, a_j
- Find sequence $X_i = f(X_{i-1})$
- Eventually, there will be a cycle. Collision point found by Floyd's Cycle Finding algorithm.
- $\exists$ two distinct paths to the same point, so we can extract $c^{'}, c^{''}, d^{'}, d^{''}$

Pollard's Rho Algorithm



Proof of Correctness

- Define group G, continuously updating $c^{'}, c^{''}, d^{'}, d^{''}$
- Lemma: A cycle must exist
  - G is finite, but the sequence is infinite
  - By Pigeonhole, a cycle must exist
- Say the cycle is detected at $X_t = X_{t+s}$ for some j
- $X_t \in S_x$ for some x with corresponding a_x, b_x
- $X_{t+s} \in S_y$ with corresponding a_y, b_y
- Without loss of generality consider $c^{'}$.
- On iteration i, assume we can determine $c^{'} = c$
- On iteration $i+1$, $c^{'} = a_x + c$
- Thus, we can extract $c^{'}, c^{''}, d^{'}, d^{''}$, and find $k = (c^{'} - c^{''})(d^{''} - d^{'})^{-1}$, solving the discrete log problem
- Runtime: Collision expected after $\sqrt{\frac{\pi n}{2}}$.

Post-Quantum Cryptography

- Fourier Transforms can also find these ”cycles”
- Quantum computers compute Fourier Transforms extremely efficiently
- In a quantum world, Shor’s Algorithm breaks elliptic curve cryptography