Contents

Public Key Cryptography

Limitation of symmetric key

- Key distribution problem
  - How can keys be exchanged secretly?
- Too many symmetric keys
  - For n users, each user should keep n-1 keys and in total n(n+1)/2 keys are required.
- Alice and Bob may cheat each other.
  - Can be used for non-repudiation

Public Key Cryptography

- Two keys
  - Each user generates two keys: **public key** and **private key**
  - Each user lets others know its own **public key.**
  - At key generation time, two keys are computed.

K^{+}
(public key)

K^{-}
(private key)

Uses of Public Key Crypto

- Encryption
  - Suppose we **encrypt** M with Bob's public key
  - Bob's private key can **decrypt** to recover M
- Digital Signature
  - **Sign** by encrypting with your private key
  - Anyone can **verify** signature by decrypting with sender's public key
  - Like a handwritten signature, but way better…
- Key exchange
  - We will talk about it later.

How to build public key crypto

- Based on "trap door one-way function"
  - "One-way" means easy to compute in one direction, but hard to compute in other direction
  - One-way function f(x)
    - Computing $y=f(x)$ is computationally easy.
    - Computing $x=f^{-1}(y)$ is computationally infeasible.
  - "Trap door" used to create key pairs

3 kinds of public key crypto

- There are 3 kinds of mathematically hard one-way functions on which the public key crypto are based.
  - Factoring integers
    - RSA
  - Discrete Logarithm
    - Diffie-Hellman
  - Elliptic curve: generalized discrete log
    - ECDH, ECDSA

RSA

RSA

- Diffie and Hellman published the idea of the public key crypto in 1976.
- The RSA crypto was published by **R**ivest, **S**hamir, and **A**dleman (MIT) in 1977, and Clifford Cocks (GCHQ), independently,
- So far, RSA is the most widely used the public key cypto although ECC is gaining attention recently.

Factoring integers

- Let p and q be two large prime numbers
- Compute N = pq
- but, to find p and q from N such that N=pq for large enough p and q is computationally very hard problem.

Encryption and decryption

- **Public key** $K^{+}=(N,e)$
- **Private key** $K^{-}= d$
- Encryption $y=E_{K+}(x) = x^{e} \bmod N$
- Decryption $x=D_{K-}(y) = y^{d} \bmod N$

keys generation algorithm

At the setup time, the public and private keys are computed as follows:

1. Choose two large prime numbers
2. Compute $N=p\cdot q$
3. Compute $\varphi(n)=(p-1)(q-1)$
4. Choose $e \in\{1,2,3,\ldots, \varphi(n)-1\}$ such that

   $\gcd(e, \varphi(n)) = 1$
5. Compute d such that

   $d\cdot e=1 \bmod \varphi(n)$
6. Return $K^{+}=(N,e)$ and $K^{-}= d$

RSA

- Message M is treated as a number
- To encrypt M we compute

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

- To decrypt ciphertext C compute

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

- Recall that e and N are public
- If Trudy can factor N=pq, she can use e to easily find d since ed = 1 mod (p–1)(q–1)
- **Factoring the modulus breaks RSA**
  - Is the factoring the only way to break RSA?

Does RSA Really Work?

- Given $C = M^e \bmod N$ we must show
  $M = C^d \bmod N = M^{ed} \bmod N$
- We'll use **Euler's Theorem:**
  If x is relatively prime to n then $x^{\varphi(n)} = 1 \bmod n$
- Facts:
  1) $ed = 1 \bmod (p-1)(q-1)$
  2) By definition of "mod", $ed = k(p-1)(q-1) + 1$
  3) $\varphi(N) = (p-1)(q-1)$
- Then $ed - 1 = k(p-1)(q-1) = k\varphi(N)$
- Finally, $M^{ed} = M^{(ed-1)+1} = M \cdot M^{ed-1} = M \cdot M^{k\varphi(N)} = M \cdot (M^{\varphi(N)})^k \bmod N = M \cdot 1^k \bmod N =$ **$M \bmod N$**

Simple RSA Example

Alice

Message $x=8$

Bob

1. Select large primes $p=11$, $q=3$
2. $N=pq=33$
3. $\varphi(n)=(p-1)(q-1) = 20$
4. Choose $e=3$ (relatively prime to 20)
5. Find $d=7$ such that $ed=1 \bmod 20$

$K^{+} = (33,3)$ ←

$K^{-}=7$

$Y = x^e \bmod 33 = 8^3 = 512 = 17 \bmod 33$

$Y=17$ →

$x = y^d \bmod N = 17^7 = 410{,}338{,}673$
$= 12{,}434{,}505 * 33 + 8 = 8 \bmod 33$

More Efficient RSA (1)

- Modular exponentiation example
  - $5^{20} = 95367431640625 = 25 \bmod 35$
- A better way: **repeated squaring**
  - $20 = 10100$ base 2
  - $(1, 10, 101, 1010, 10100) = (1, 2, 5, 10, 20)$
  - Note that $2 = 1 \cdot 2$, $5 = 2 \cdot 2 + 1$, $10 = 2 \cdot 5$, $20 = 2 \cdot 10$
  - $5^1 = 5 \bmod 35$
  - $5^2 = (5^1)^2 = 5^2 = 25 \bmod 35$
  - $5^5 = (5^2)^2 \cdot 5^1 = 25^2 \cdot 5 = 3125 = 10 \bmod 35$
  - $5^{10} = (5^5)^2 = 10^2 = 100 = 30 \bmod 35$
  - $5^{20} = (5^{10})^2 = 30^2 = 900 = 25 \bmod 35$
- No huge numbers and it's efficient!

More Efficient RSA (2)

- Use **$e = 3$** for all users (but not same N or d)
  - + Public key operations only require 2 multiplies
  - o Private key operations remain expensive
  - \- If $M < N^{1/3}$ then $C = M^e = M^3$ and **cube root attack**
  - \- For any M, if C_1, C_2, C_3 sent to 3 users, cube root attack works (uses Chinese Remainder Theorem)
- Can prevent cube root attack by padding message with random bits
- Note: $e = 2^{16} + 1$ also used ("better" than $e = 3$)

RSA in retrospect

- Currently RSA is the most widely used public crypto.
- Main uses are digital signature and key exchange.
- Currently 1024bits cannot be factored, but 2048 to 3076 bits are highly recommended for long-term security.
- Ingenuous implementation exposes several attacks. Meticulous implementation is required.

Encrypting Large File with RSA?

- Duration of 1024-bit RSA encryption
  - ~1 ms on 1 GHz Pentium
- Duration of 1024-bit RSA decryption
  - ~10 ms on 1 GHz Pentium
- Duration to encrypt 1 Mbyte file?
  - Encrypt 1024 bits / RSA operation = 128 bytes
  - 1 Mbyte = 2^{20} bytes
  - Time: $2^{20} / 2^{7} * 1ms = 2^{13}$ ms = 8 seconds!
  - Compare with the time by the symmetric key?

Symmetric-key vs. public-key

- Symmetric crypto
  - Need shared secret key
  - 80 bit key for high security (year 2010)
  - ~1,000,000 ops/s on 1GHz processor
  - 10x speedup in HW
- Public-key crypto
  - Need authentic public key
  - 2048 bit key (RSA) for high security (year 2010)
  - ~100 signatures/s
    ~1000 verify/s (RSA) on 1GHz processor
  - Limited speedup in HW

Discrete Logarithmic problem
and
Diffie-Hellman key exchange

Cyclic Group

Suppose a cyclic group Z^*_{11}={1,2,3,….,10).

What happens if we compute 2^x mod 11.

Observation:
“2” generates all members of Z^*_{11}
at every 11th computation.

So, a=2 is called a generator of Z^*_{11}.

2^1 mod 11=2
2^2mod 11=4
2^3mod 11=8
2^4mod 11=5
2^5mod 11=10
2^6mod 11=9
2^7mod 11=7
2^8mod 11=3
2^9mod 11=6
2^{10}mod 11=1
2^{11}mod 11=2
2^{12}mod 11=4

Discrete Logarithm Prob(DLP)

Given the finite cyclic group Z_p* of order p-1 and a primitive element $g \in Z_p$* and another element $\beta \in Z_p$*.

The DLP is the problem of determining the integer x such that

$1 \le x \le p-1$

$g^x = \beta \bmod p$, i.e.,

$x = \log_g \beta \bmod p$

In the previous example, $2^x = 3 \bmod 11$, what is x?

$5^x = 41 \bmod 47$, what is x?

D-H key exchange

Alice	p, g : public	Bob
Select a $\in\{2,3,\ldots,p-2\}$ (private to Alice)		Select b $\in\{2,3,\ldots,p-2\}$ (private to Bob)
Compute $A= g^{a} \bmod p$		Compute $B= g^{b} \bmod p$
	$\xrightarrow{A}$	
	$\xleftarrow{B}$	
$K_{AB}=B^{a} \bmod p = g^{ab} \bmod p$		$K_{AB} = A^{b} \bmod p = g^{ab} \bmod p$
Message x Encrypt: $Y=E_{K_{AB}}(x)$	$\xrightarrow{y}$	Decrypt: $x=D_{K_{AB}}(y)$

Security of D-H

- Suppose an attacker can only listen the channel(passive attack).
  - What can he know? g, p, A, B
  - What does he want to know? $K_{AB}=g^{ab} \bmod p$
- One way of solving the problem is:
  - Compute $a = \log_g A \bmod p$ or $b = \log_g B \bmod p$
- This computation is a very hard problem if p is large enough.

- Attacks against the DLP
  - Goal: solve $x = \log_g \beta \mod p$
    - $g, \beta \in Z_p^*$, n=the number of elements of Z_p^*(cardinality of Z_p^*)
  - Brute force attack requires O(n) steps.
  - If this is the only possible attack, $n \geq 2^{80}$.
  - But the Square-Root method can compute β $\sqrt{n}$ steps.
  - So, choose $n=2^{160}$.
  - In practice, $p \geq 2^{1024}$

Encryption with DLP

- Use the classic D-H key exchange algorithm.

Alice	p, g : public	Bob
Select $a \in \{2,3,\ldots,p-2\}$ (private to Alice)		Select $b \in \{2,3,\ldots,p-2\}$ (private to Bob)
Compute $A = g^a \bmod p$		Compute $B = g^b \bmod p$
	$\xrightarrow{A}$	
	$\xleftarrow{B}$	
$K_{AB} = B^a \bmod p = g^{ab} \bmod p$		$K_{AB} = A^b \bmod p = g^{ab} \bmod p$
Message x		
Encrypt: $Y = x \cdot K_{AB} \bmod p$	$\xrightarrow{y}$	Decrypt: $x = y \cdot K_{AB}^{-1} \bmod p$

Elgamal Encryption algorithm

- Was published around 1985
- Very similar to D-H, but the steps are reordered.

Alice | **Bob**

Bob:
Select p, $g \in \{2,3,\ldots,p-2\}$
$K- = d \in \{2,3,\ldots,p-2\}$
$K+ = \beta = g^d \bmod p$

Bob → Alice: $(K+=\beta, g, p)$

Alice:
Select $i \in \{2,3,\ldots,p-2\}$
$K_E = g^i \bmod p$ (ephemeral key)
$K_M = \beta^i \bmod p$ (masking key)

Message x
Encrypt: $Y = x \cdot K_M \bmod p$

Alice → Bob: (y, K_E)

Bob:
$K_M = K_E^d \bmod p$
Decrypt: $x = y \cdot K_M^{-1} \bmod p$

Proof

Bob computes:

$$
\begin{aligned}
y\, K_M^{-1} &= y(K_E^{d})^{-1} \\
&= x\, K_M\, K_E^{-d} \\
&= x\, \beta^{i}\, (g^{i})^{-d} \\
&= x\, (g^{d})^{i}\, (g^{i})^{-d} \\
&= x
\end{aligned}
$$

In Elgamal encryption, the public key($K+=\beta$) is fixed, but i is Chosen for each message. So, K_E must be different for every plaintext.

Elliptic Curve Cryptography (ECC)

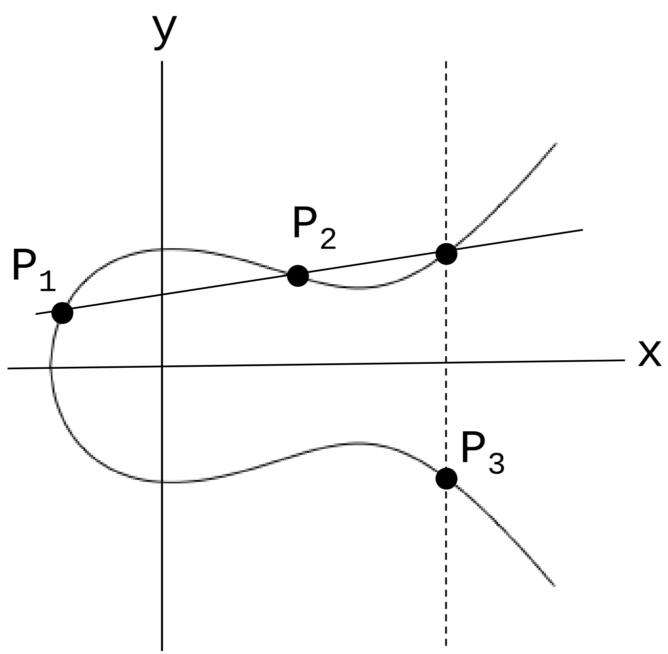
What is an Elliptic Curve?

- An elliptic curve E is the graph of an equation of the form

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

- Also includes a "(imaginary) point at infinity"
- What do elliptic curves look like?

Elliptic Curve Picture



- Consider elliptic curve

  $E: y^2 = x^3 - x + 1$

- If P_1 and P_2 are on E, we can define

  $$P_3 = P_1 + P_2$$

  where + is a point addition operator(not a vector operator).
- Point addition operator is all we need

Analytical expression for operator +

Given a EC, $y^2 = x^3+ax+b$, $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P_3=(x_3, y_3)$=?

Assume that the equation of a line passing through P_1 and P_2,

$$y = mx+c$$

Then, $(mx+c)^2 = x^3+ax+b \rightarrow$ 3 solutions: P_1 , P_2, and $P_3=(x_3, y_3)$

$$x_3 = m^2-x_1 -x_2 \bmod p,$$

$$y_3 = m(x_1 -x_3)-y_1 \bmod p$$

where $m = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \bmod p & \text{; if } P \neq Q \text{ (point addition)} \\ \dfrac{3x_1^2 + a}{2y_1} \bmod p & \text{; if } P = Q \text{ (point doubling)} \end{cases}$

the (imaginary) point of infinity: ∞

We define a "point of infinity", ∞ as

$$P + \infty = P \text{ for all } P \text{ on EC}$$

What is the graphic interpretation of ∞?

$$P + (-P) = \infty \text{ for all p}$$

That is, -P of P(x, y) is by definition (x,-y).

Cyclic Group

Suppose the following EC: $y^2 = x^3+2x+2 \bmod 17$, and a primitive point(generator) P=(5,1)

2P = P + P = (5,1) + (5,1) = (6,3)
3p = 2p + P = (10,6)
4p = (3,1)
5P = (9,16)
6P = (16,13)
7P = (0,6)
8P = (13,7)
9P = (7,6)
10P = (7,11)

11P = (13,10)
12p = (0,11)
13p = (16,4)
14P = (9,1)
15P = (3,16)
16P = (10,11)
17P = (6,14)
18P = (5,16)
19P = ∞
20P =

These points on EC has the cyclic group of the order |E|=19.

(source: Understanding Cryptography)

Number of points on an EC

- How many points can be on an arbitray EC?
- Hasse's Theorem"
  - Given an elliptic curve module p, the number of points on the curve is bounded by

    $$p+1-2\sqrt{p} \le \#E \le p+1+2\sqrt{p}$$

  So, the number of point is close to p.

  To generate a curve with about 2160 points, a prime of a length of about 160 bits is required.

(source: Understanding Cryptography)

EC Discrete Logarithm Problem

- Given an EC, we consider a primitive element p and another point T on the curve. The DL problem is to find the integer d, where $1 \leq d \leq |E|$, such that

$$\underbrace{p + p + p + \ldots + p}_{\text{d times}} = d \cdot p = T$$

EC DH Key Exchange and encryption

Alice	$E: y^2 = x^3+ax+b$, $P= (x_p, y_p)$	Bob
Select $a \in \{2,3,\ldots,\lvert E\rvert\}$ (private to Alice) Compute $A= aP =(x_A, y_A)$		Select $b \in \{2,3,\ldots,\lvert E\rvert\}$ (private to Bob) Compute $B= bP =(x_B, y_B)$
	$\xrightarrow{A}$	
	$\xleftarrow{B}$	
$K_{AB}= aB = abP = (x_{AB}, y_{AB})$		$K_{AB} = bA=abp =(x_{AB}, y_{AB})$
Message x Encrypt: $Y=E_{K_{AB}}(x)$	$\xrightarrow{y}$	Decrypt: $x=D_{K_{AB}}(y)$

ECC Security

- Practical parameter size for ECC
  - p with 160 bits (roughly 160 points on the curve) provides 2^{80} steps that are required by an attacker.
- Why smaller for ECC (160-256bits) than for RSA(1024-3072bits)?
  - Attacks on ECC are weaker than those on the integer factoring or integer DL.
- For this reason, ECC slowly becomes popular on many applications, especially on embedded platforms such as mobile devices.

Comparison of Security level

Algorithm family	cryptosystem	Security level(bits)			
		80	128	192	256
Integer factoring	RSA	1024	3072	7680	15360
Discrete logarithm	DH, DSA, Elgamal	1024	3072	7680	15360
Elliptic curve	ECDH, ECDSA	160	256	384	512
Symmetric key	AES, 3DES	80	128	192	256

(source: Understanding Cryptography)