SJSU SAN JOSÉ STATE UNIVERSITY

Lesson 6 – Public Key Crypto 2

Yan Chen

CS166 Fall 2024

- Public key crypto: different keys to encrypt and decrypt
  - ➢ Use "trapdoor one-way functions" – NO key exchange needed!
- Public key crypto for encryption
  - ➢ $\{M\}_{Alice}$: encrypt M with Alice's public key
  - ➢ Alice decrypts $\{M\}_{Alice}$ to get M with her private key
- Public key crypto for key exchange
- Public key crypto for signature
  - ➢ $[M]_{Bob}$: encrypt M with Bob's private key ($[M]_{Bob}$ = "signature")
  - ➢ Others verify the signature by decrypting $[M]_{Bob}$ with Bob's public key to check if M is correct (i.e., $\{[M]_{Bob}\}_{Bob}$= M)

- Prime: an integer (> 1) only is divided by 1 and itself.
- Greatest Common Divisor (GCD) of x and y: the largest integer d such as d is a divisor of both x and y.
  - Can be calculated by Euclidean algorithm
- Two integers x and y are relatively prime if $\gcd(x, y) = 1$.
  - x and y does NOT have to be primes
- Totient function $\varphi(n)$: the number of numbers less than n that are relatively prime to n.
  - $\varphi(p) = p - 1$ if p is prime
  - $\varphi(pq) = \varphi(p) * \varphi(q) = (p - 1)(q - 1)$ if p and q are primes

- $x \bmod N$ (modulo): remainder of x divided by n.
  - Result "circled" from 0 to N – 1 ("Clock" arithmetic)
- Congruence: $a \equiv b \pmod N$ means $a \bmod N = b \bmod N$
  - n is the divisor of a – b : $a - b = kN$ for an integer k
- Modular inverses
  - Additive: $-x \bmod N = y$ if $x + y \equiv 0 \pmod N$
  - Multiplicative: $x^{-1} \bmod N = y$ if $xy \equiv 1 \pmod N$
  - $x^{-1} \bmod N$ exists only when x and N are relatively prime, and it can be calculated using Extended Euclidean algorithm ($x^{-1} \bmod N = a$ in $ax + bN = 1$ when x and N are coprime)

- To generate keys:
  - Let p and q be two large prime numbers, and N = pq
  - Choose e relatively prime to $(p-1)(q-1) = \varphi(N)$
  - Find $d = e^{-1} \bmod \varphi(N)$ (i.e., $ed \equiv 1 \bmod \varphi(N)$)
  - Public key is (N, e)
  - Private key is d (p and q are also secrets!)
- Message (plaintext) M is treated as a number
  - To encrypt M, compute $C \equiv M^e \bmod N$
  - To decrypt ciphertext C, compute $M \equiv C^d \bmod N$

- The security of RSA is based on it's hard to factorize N
  - Once Trudy knows p & q that used to get N, easy to get private key d using extended Euclidean algorithm
  - N needs to be big, new standard asks for 2048 bits at least
- Size of e doesn't matter as much as the size of N...
  - But choosing 3 as e may result in cube root attack
  - Since it is possible that $M^3 < N$ so $C = M^3 \bmod N = M^3$

- Diffie-Hellman (DH): key exchanging algorithm
  - Invented by Williamson, Diffie and Hellman
  - Used to exchange symmetric key
  - NOT for encrypting or signing!
- DH also uses a pair of 2 keys
  - Public key: a prime p and a “generator” g
  - Private key: each party has one private key
- And DH is based on a “one-way function”
  - Easy: given (g, p, x), get $g^x \bmod p$
  - Hard: given (g, p, $g^x \bmod p$), get x

- Diffie-Hellman algorithm:

g^a mod p →

← g^b mod p

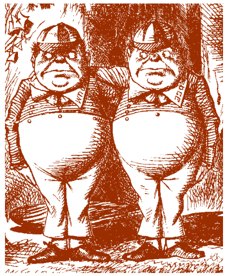
Alice, a — Bob, b

  - a and b are private, but $g^a \bmod p$ and $g^b \bmod p$ are public
  - After exchange, compute $K = g^{ab} \bmod p$ as symmetric key
- Only Alice and Bob can get $g^{ab} \bmod p$
  - Alice computes $(g^b \bmod p)^a \equiv g^{ba} \equiv g^{ab} \bmod p$
  - Bob computes $(g^a \bmod p)^b \equiv g^{ab} \bmod p$
- Trudy can't since $(g^a \bmod p) * (g^b \bmod p) \neq g^{ab} \bmod p$

- DH is subject to man-in-the-middle (MiM) attack

$g^a \bmod p$ →

← $g^t \bmod p$



$g^t \bmod p$ →

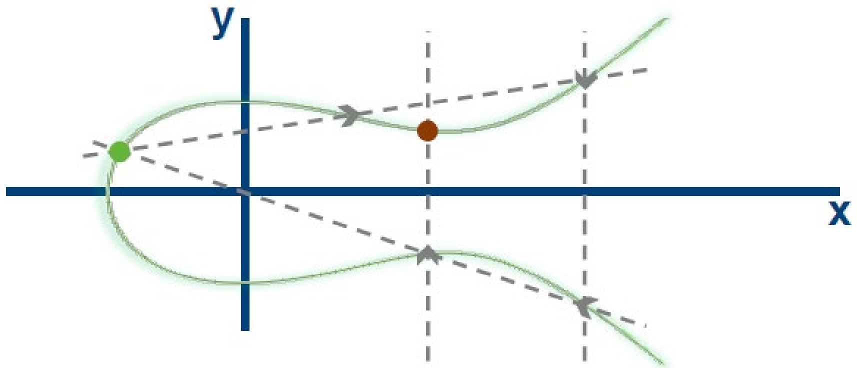
← $g^b \bmod p$

Alice, a | Trudy, t | Bob, b

  - Trudy shares secret $g^{at} \bmod p$ with Alice
  - Trudy shares secret $g^{bt} \bmod p$ with Bob
  - Alice and Bob doesn't know each other's private key ...

    i.e., they didn't know what they would get from each other...
  - Therefore, Alice and Bob don't know Trudy is in the middle!
- Will come back DH later when discussing protocols...

- Elliptic Curve Crypto (ECC): a different way to do the math in public key system using curve $y^2 = x^3 + ax + b$
  - "Elliptic curve" is not a cryptosystem
  - We can have ECC version of DH and RSA, etc.
- A high-level view of ECC



  - Public key: the curve, start point and end point
  - Private key: number of "steps" to get from start to end

- Pros: smaller keys, more efficient
  - Recall: we need large key for RSA (at least 2048 bits)
  - For ECC, ~224 bits key will provide **same level of security**
  - Roughly speaking, to achieve **same level of security**, the key length for RSA is 10 times the key length for ECC
- Cons: math too complicated
  - No formal proof of security yet
  - Not many people can fully understand it...
  - Like, how to pick a secure curve, etc.

- Digital Signature
- PKI

- Diffie-Hellman
  - MiM attack
- Elliptic curve crypto

- Suppose Alice and Bob try to exchange a symmetric key using Diffie-Hellman. Given p & g, if Bob gets N from Alice, and Bob's private key is b, what should be the shared symmetric key?
- About ECC
  - ➢ What is its advantage?
  - ➢ What is its disadvantage and why?

References

- Stamp, Mark, “Information Security, Principles and Practice, 2nd ed.,” Wiley, New Jersey, USA, 2011