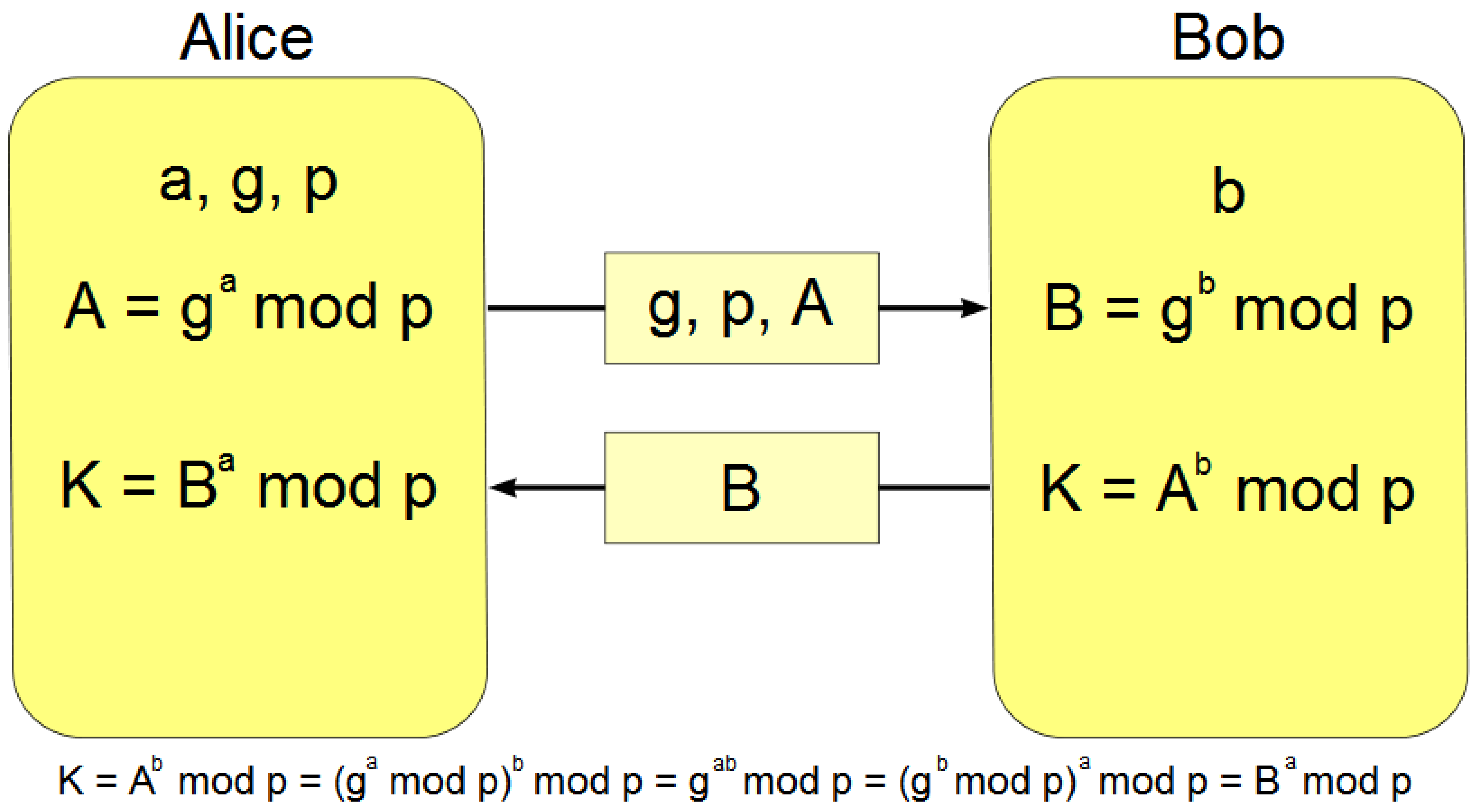
Public-key Algorithms

Z. Jerry Shi
Department of Computer Science and Engineering
University of Connecticut

Key establishment

- Symmetric-key cipher requires sharing of a secret
- Can it be done on a public channel?

Diffie-Hellman Protocols



Alice
Bob
a, g, p
$A = g^a \bmod p$
$K = B^a \bmod p$
g, p, A
B
b
$B = g^b \bmod p$
$K = A^b \bmod p$
$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$

g and p can be public. g can be small.

Difficult to do discrete logarithm

Diffie-Hellman key exchange

- Alice and Bob want to construct a private key over a public channel. Both agree on a public prime p and primitive root g modulo p.

1. Alice chooses a random value x
2. Alice sends g^x mod p to Bob
3. Bob chooses a random value y
4. Bob sends g^y mod p to Bob
5. Alice computes g^{xy} mod p as $(g^x$ mod $p)^y$ mod p
6. Bob computes g^{xy} mod p as $(g^y$ mod $p)^x$ mod p

- **Discrete logarithm problem:** Given g, p, and g^x mod p, find x.
- **Diffie-Hellman problem:** Given g, p, g^x mod p, and g^y mod p, find g^{xy} mod p.

DHP and DLP

- One way to solve DHP is to solve DLP

  Given g^x mod p and g^y mod p, if one can solve DLP, s/he can recover x and y, and easily compute g^{xy} mod p.

  - Other ways? Not very likely (so far)

- If DHP is hard, the key agreement protocol is secure

- DHP is not harder than DLP

- Many groups:
  - DHP is equivalent to DLP
    - Given an oracle to solve DHP, DLP can be solved

DLP

- Discrete log in GF(*p*)
  - Hard if *p* is ‘safe prime’: (*p* – 1)/2 is also prime
  - RFC defines groups with moduli of from 1536 to 8192 bits
    - The size of exponents are from 180 to 620

- Discrete in other groups
  - Probably even harder (because of less algebraic structure)
  - For example, points on elliptic curve over a finite field

Diffie-Hellman Protocol for 3 Parties

	Alice	Bob	Charlie
Generate secret	x	y	z
Exponentiation	g^x	g^y	g^z
Communication	$g^x \rightarrow$ B	$g^y \rightarrow$ C	$g^z \rightarrow$ A
Exponentiation	g^{xz}	g^{xy}	g^{yz}
Communication	$g^{xz} \rightarrow$ B	$g^{xy} \rightarrow$ C	$g^{yz} \rightarrow$ A
Exponentiation	g^{xyz}	g^{xyz}	g^{xyz}

The number of exponentiations can be reduced, e.g., by coordinating through a binary tree

Properties of public-key algorithms

- A pair of keys: public and private keys
  - Encrypt with private key, decrypt with public key
  - Encrypt with public key, decrypt with private key

- The public key can be known by everyone and the private key should be kept secret
  - It is hard to derive private key from public key

- RSA is the best known public-key algorithm
  - Rivest, Shamir & Adleman of MIT in 1977
  - Patent expired on 9/21/2000 (only in US)
  - Clifford Cocks (UK) had the idea in 1973

RSA set up

Find two distinct prime numbers p and q

which should be large and kept secret

$n = p \bullet q$ n is the modulus. n's length is the key length.

$\varphi(n) = (p - 1)(q - 1)$ $\varphi(n)$ should be kept secret.

Alternaivly use $\lambda(n) = \mathrm{lcm}(p - 1, q - 1)$

Choose e such that $1 < e < \varphi(n)$ and e is coprime to $\varphi(n)$.

e can be released as the public exponent

Compute the private exponent d such that

$$e \bullet d = 1 \bmod (p - 1)(q - 1)$$

RSA encryption and decryption

Encryption:

Given a message m, compute

$$c = m^e \bmod n$$

Note e and n are public.

Decryption:

Given a ciphertext c, compute

$$c^d = (m^e)^d = m^{ed} = m^{k(p-1)(q-1)+1} = m \bmod n$$

It works even if m is not coprime to n.

RSA Proof (correctness)

- gcd $(m, n) = 1$

  $c^d = (m^e)^d = m^{ed} = m^{k(p-1)(q-1)+1} = m \bmod n$

- gcd $(m, n) > 1$, $p \mid m$ or $q \mid m$. (m is evenly divided by p or q)

  Use CRT. If two numbers are congruent both mod p and mod q, they also congruent mod pq.

  WLOG, let us assume $p \mid m$.

  $m = 0 \bmod p$. $m^{ed} = 0 \bmod p$. Therefore, $m = m^{ed} \bmod p$.

  $m^{ed} = m^{k(p-1)(q-1)+1} = (m^{k(p-1)})^{(q-1)} \bullet m = m \bmod q$. (FLT)

  By CRT, $m = m^{ed} \bmod n$, where $n = pq$.

RSA examples

$p = 19$, $q = 31$, $n = 589$

$\varphi(n) = 18 * 30 = 540$

$e = 13$ (factors in 540: 2, 3, 5).

$d = e^{-1} = 457$ mod 540 (not mod 589)

$m = 387$ $c = m^d = 387^{457} = 178$ $m^{ed} = 178^{13} = 387$ mod 589

$m = 323$ $c = m^d = 323^{457} = 456$ $m^{ed} = 456^{13} = 323$ mod 589

Note: 323 = 19 * 17 has a common factor with $n = 589$

RSA Implementation

- Use CRT to accelerate the exponentiation if p and q are known
  - Speedup 2.5 times or more
- Miller-Rabin used as p and q for primality test
- Strong primes are mandated in ANSI X9.31
- $p - q$ should be large

- Use smaller public exponent
  - Security concerns

- RSA is much slower than block ciphers
  - For example, on 200M Hz Pentium Pro, the throughput of AES is about 70M bits/s while that of RSA decryption is about 30K bits/s

Square root of 1 mod *n*

Fact 1: if n is prime, there are no non-trivial square roots of 1 mod n

Trivial square roots of 1 mod p : 1 and –1 (or p – 1)

If n is not prime, there are non-trivial square roots of 1.

Suppose $n = p \bullet q$.

According to CRT: A number x mod n can be mapped to

two numbers (x mod p, x mod q).

square roots are: (1, 1), (1, – 1), (– 1, 1), (– 1, – 1).

Example, $p = 5$, $q = 7$, and $n = 35$. Square roots of 1 are:

Trivial roots: 1 □ (1, 1) 34 = –1 □ (– 1, – 1) = (4, 6)

Non-trivial roots:

6 □ (1, – 1) = (6 mod 5, 6 mod 7)

29 □ (– 1, 1) = (4 mod 5, 1 mod 7)

Square root of 1 mod *n* (2)

Let $x > 1$ be a non-trivial square root of 1 mod n. So $x^2 = 1 \bmod n$.

$$x^2 - 1 = 0 \bmod n$$

$$(x + 1) \bullet (x - 1) = 0 \bmod n$$

$$(x + 1) \bullet (x - 1) = k\, n$$

If x is not 1 or –1, $(x + 1)$ and $(x - 1)$ have common factors with n.

gcd(x+1, n) and gcd(x – 1, n) can be computed with extended Euclidean algorithm.

Factoring *n* vs exposing *d*

Theorem: Factoring N is equivalent to exposing d

If one can factor N, d can be computed from e

If d is known, one can factor N

$e \cdot d = 1 \bmod \varphi(n)$

$e \cdot d - 1 = k\,\varphi(n)$

$a^{e \cdot d - 1} = 1 \bmod n$ for all a that is coprime to n

Let $(e \cdot d - 1) = 2^s \cdot t$ and t is odd.

Select a that is coprime to n, compute

$r_1 = \mathrm{a}^{(e \cdot d - 1)/2}$, $r_2 = a^{(e \cdot d - 1)/4}$, $r_3 = a^{(e \cdot d - 1)/8}$,… ,$a^t \bmod n$

With 50% chance, one of the value $\neq \pm1$. Note r_2 is the square root of r_1.

Suppose the first is z. $\gcd(z - 1, n)$ is a non-trivial factor of n

Example: factoring *n* if *d* is known

$p = 19$, $q = 31$, $n = 589$

$\varphi(n) = 18 * 30 = 540$

$e = 13$ (13 is coprime to 540. Factors in 540 are 2, 3, and 5)

$d = e^{-1} = 457$ mod 540 (not mod 589)

$(d \cdot e - 1) = 5940 = 2^2 * 1485$ ($s = 2$, $t = 1485$)

The exponents to be used in test are 1485 and 2970.

Suppose randomly pick 90. 90 is coprime to 589.

90 $90^{2970} = 1$ $90^{1485} = 94$ mod 589

94 is a non-trivial square-root of 1.

93 and 95 have common factors with 589.

gcd(93, 589) = 31 gcd(95, 589) = 19

Possible attacks on RSA (1)

- Factor *n*
  - Factor *n*, then *d* can be computed easily from *e*, because *φ* (*n*) is known
- Guessing *d*
  - The factors of *n* can be found if *d* is known
  - It is slow because there are so many *d*s
- Cycle attack
  - Repeat encryption until the ciphertext repeats
- Common modulus
  - For example, use *n* to generate (*e*0, *d*0), (*e*1, *d*1), etc.
  - Do not use the same *n* to encrypt messages to different parties
    - If one knows *d*, he can factor *n*

Possible attacks on RSA (2)

- Faulty encryption
  - Similar to common modulus

- Low exponent (either public or private)
  - If e is small, and m is also same for e parties, m can be obtained with CRT
  - If d is small (length is less ¼ of n), it can be computed from (n, e)

- Small message

Common modulus attack

Suppose Alice and Bob generate their keys using the same n.

Alice's keys are (n, e_a). Bob's keys are (n, e_b). $\gcd(e_a, e_b) = 1$

One send the same message m to both Alic and Bob.

Eve obtained the ciphertext: and .

She can find out m.

First, she finds out s_a and s_b such that, .

Then she computes

Factoring large numbers

- General number field sieve
  - $(c + o(1))\, n^{1/3} \log^{2/3} n)$ for some $c < 2$

- The largest number factored in 2010 was 768 bits long
  - RSA-768

- In $p - 1$ is a product of factors less than B, then N can be factored in time less than B^3
  - Such p can be rejected

ElGamal Encryption

- **Key generation:**
  - Parameters: (safe) prime p and generator α of $GF(p)$
  - **Private key**: x $(1 < x < p - 1)$
  - **Public key**: $y = \alpha^{x} \bmod p$
- **Encryption:**
  - Generate random k $(1 < k < p - 1)$ with $\gcd(k, p - 1) = 1$
  - $r = \alpha^{k} \bmod p$ (k and r are ephemeral key pair)
  - $s = y^{k} \cdot m \bmod p$ $(0 \leq m \leq p - 1)$
  - **Ciphertext** $c = (r, s)$
- **Decryption:**
  - $m = s \cdot r^{-x} \bmod p$

    $r^{-x} = \alpha^{-kx} = y^{-k} \bmod p$

About ElGamal

- Security relies on the discrete log problem and not on factoring
- Ciphertext twice as long as the plaintext
- Secure random number generator required for *k*
  - Non-deterministic encryption: the same plaintext will always result in different ciphertexts

Some theroms

- It is difficult to find the e^{th} root of c mod $n = pq$ when the factors of n is unknown