Algorithmic Number Theory and Public-key Cryptography

Course 5

Jean-Sébastien Coron

University of Luxembourg

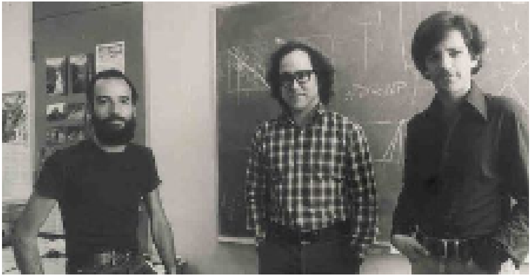
April 10, 2014

Summary

- Algorithmic number theory.
  - Application of Euler function and Fermat's little theorem: the RSA algorithm.
  - Probabilistic primality testing and prime number generation.

The RSA algorithm

- The RSA algorithm is the most widely-used public-key encryption algorithm
  - Invented in 1977 by Rivest, Shamir and Adleman.
  - Used for encryption and signature.
  - Widely used in electronic commerce protocols (SSL).



Public-key encryption

- Public-key encryption: two keys.
  - One key is made public and used to encrypt.
  - The other key is kept private and enables to decrypt.
- Alice wants to send a message to Bob:
  - She encrypts it using Bob's public-key.
  - Only Bob can decrypt it using his own private-key.
  - Alice and Bob do not need to meet to establish a secure communication.
- Security:
  - It must be difficult to recover the private-key from the public-key
  - but not enough in practice.

RSA

- Key generation:
  - Generate two large distinct primes p and q of same bit-size.
  - Compute $n = p \cdot q$ and $\phi = (p-1)(q-1)$.
  - Select a random integer e, $1 < e < \phi$ such that $\gcd(e, \phi) = 1$
  - Compute the unique integer d such that

    $$e \cdot d \equiv 1 \pmod{\phi}$$

    using the extended Euclidean algorithm.
  - The public key is (n, e). The private key is d.

RSA encryption

- Encryption
  - Given a message $m \in [0, n-1]$ and the recipent's public-key (n, e), compute the ciphertext:

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

- Decryption
  - Given a ciphertext c, to recover m, compute:

$$m = c^d \mod n$$

- Message encoding
  - The message m is viewed as an integer between 0 and $n-1$
  - One can always interpret a bit-string of length less than $\lfloor \log_2 n \rfloor$ as such a number.
  - One must be careful: plain RSA encryption is insecure.

Proof that decryption works

- We must show that $m^{ed} = m \bmod n$.
- Since $e \cdot d \equiv 1 \mod \phi$, there is an integer k such that $e \cdot d = 1 + k \cdot \phi = 1 + k \cdot (p-1) \cdot (q-1)$. Therefore we must show that:

$$m^{1+k\cdot(p-1)\cdot(q-1)} \equiv m \pmod{n}$$

- If $m \neq 0 \mod p$, then by Fermat's little theorem $m^{p-1} \equiv 1 \pmod{p}$, which gives :

$$m^{1+k\cdot(p-1)\cdot(q-1)} \equiv m \pmod{p}$$

  - This equality is also true if $m \equiv 0 \pmod{p}$.
  - This gives $m^{ed} \equiv m \pmod{p}$ for all m.
  - Similarly, $m^{ed} \equiv m \pmod{q}$ for all m.
  - By the Chinese Remainder Theorem, if $p \neq q$, then

$$m^{ed} \equiv m \pmod{n}$$

Decrypting with CRT

- Given the factors p and q of $n = p \cdot q$, instead of computing $m = c^d \bmod n$, compute:
  - $m_p = c^{d_p} \bmod p$, where $d_p = d \bmod (p-1)$
  - $m_q = c^{d_q} \bmod q$, where $d_q = d \bmod (q-1)$
  - Using CRT, find m such that $m \equiv m_p \pmod{p}$ and $m \equiv m_q \pmod{q}$:

$$m = \left(m_p \cdot (q^{-1} \bmod p) \cdot q + m_q \cdot (p^{-1} \bmod q) \cdot p\right) \bmod n$$

- Since exponentiation is cubic, this is roughly 4 times faster.

Security of RSA

- The security of RSA is based on the hardness of factoring.
  - Given $n = p \cdot q$, it should be difficult to recover p and q.
  - No efficient algorithm is known to do that. Best algorithms have sub-exponential complexity.
  - Factoring record: a 768-bit RSA modulus n.
  - In practice, one uses at least 1024-bit RSA moduli.
- However, there are many other lines of attacks.
  - Attacks against plain RSA encryption
  - Low private / public exponent attacks
  - Implementation attacks: timing attacks, power attacks and fault attacks

Primality Testing

- Motivation for prime generation:
  - Generate the primes p and q in RSA.
  - p and q must be large: at least 512 bits.
- Goal of primality testing:
  - Given an integer n, determine whether n is prime or composite.
- Simplest algorithm: trial division.
  - Test if n is divisible by 2, 3, 4, 5,... We can stop at $\sqrt{n}$.
  - Algorithm determines if n is prime or composite, and outputs the factors of n if n is composite.
  - Very inefficient algorithm
    - Requires around $\sqrt{n}$ arithmetic operations.
    - If n has 256 bits, then 2^{128} arithmetic operations. If 2^{30} operations/s, this takes 10^{22} years !

Probabilistic primality testing

- Goal: describe an efficient probabilistic primality test.
  - Can test primality for a 512-bit integer n in less than a second.
- Probabilistic primality testing.
  - The algorithm does not find the factors of n.
  - The algorithm may make a mistake (pretend that an integer n is prime whereas it is composite).
  - But the mistake can be made arbitrarily small (*e.g.* $< 2^{-100}$, so this makes no difference in practice.

Distribution of prime numbers

- Let $\pi(x)$ be the number of primes in the interval $[2, x]$.
- Theorem (Prime number theorem)
  - We have $\pi(x) \simeq x/\log x$.
- Fact (approximation of the n-th prime number)
  - Let p_n denote the n-th prime number. Then $p_n \simeq n \cdot \log n$. More explicitly,

$$n \log n < p_n < n(\log n + \log\log n) \quad \text{for } n \geq 6$$

The Fermat test

- Fermat's little theorem
  - If n is prime and a is an integer between 1 and $n-1$, then $a^{n-1} \equiv 1 \mod n$.
  - Therefore, if the primality of n is unknown, finding $a \in [1, n-1]$ such that $a^{n-1} \not\equiv 1 \mod n$ proves that n is composite.
- Fermat primality test with security parameter t.

```
For i = 1 to t do
    Choose a random a ∈ [2, n − 2]
    Compute r = a^(n−1) mod n
    If r ≠ 1 then return "composite"
Return "prime'
```

- Complexity: $\mathcal{O}(t \cdot \log^3 n)$

Analysis of Fermat's test

- Let $L_n = \{a \in [1, n-1] : a^{n-1} \equiv 1 \mod n\}$
- Theorem:
  - If n is prime, then $L_n = \mathbb{Z}_n^*$. If n is composite and $L_n \subsetneq \mathbb{Z}_n^*$, then $|L_n| \leq (n-1)/2$.
- Proof:
  - If n is prime, $L_n = \mathbb{Z}_n^*$ from Fermat.
  - If n is composite, since L_n is a sub-group of $\mathbb{Z}_n^*$ and the order of a subgroup divides the order of the group, $|\mathbb{Z}_n^*| = m \cdot |L_n|$ for some integer m.

$$|L_n| = \frac{1}{m}|\mathbb{Z}_n^*| \leq \frac{1}{2}|\mathbb{Z}_n^*| \leq \frac{n-1}{2}$$

Analysis of Fermat's test

- If n is composite and $L_n \subsetneq \mathbb{Z}_n^*$
  - then $a^{n-1} = 1 \mod n$ with probability at most $1/2$ for a random $a \in [2, n-2]$.
  - The algorithm outputs "prime" wih probability at most 2^{-t}.
- Unfortunately, there are odd composite numbers n such that $L_n = \mathbb{Z}_n^*$.
  - Such numbers are called Carmichael numbers. The smallest Carmichael number is 561.
  - Carmichael numbers are rare, but there are an infinite number of them, so we cannot ignore them.

The Miller-Rabin test

- The Miller-Rabin test is based on the following fact:
  - Let n be a prime > 2, let $n - 1 = 2^s \cdot r$ where r is odd. Let a be any integer such that $\gcd(a, n) = 1$. Then either $a^r \equiv 1 \mod n$ or $a^{2^j \cdot r} \equiv -1 \mod n$ for some j, $0 \leq j \leq s - 1$.
- Proof:
  - Since n is prime, $a^{n-1} \equiv 1 \mod n$, therefore $a^{r \cdot 2^s} \equiv 1 \bmod n$
  - Consider j_0 the minimum $0 \leq j \leq s - 1$ such that $a^{r \cdot 2^{j+1}} \equiv 1 \mod n$. Let $\beta := a^{r \cdot 2^{j_0}} \mod n$
  - Then $\beta^2 \equiv 1 \bmod n$. We must have $\beta = \pm 1$ because a polynomial of degree 2 has at most two roots over $\mathbb{Z}_n$ for n prime.
  - If $j_0 = 0$, then $a^r \equiv \pm 1 \bmod n$
  - If $j_0 > 0$, then we must have $a^{r 2^{j_0}} \equiv -1 \bmod n$ (instead j_0 would not be the minimum).

The Miller-Rabin test

Write $n - 1 = 2^s \cdot r$ for odd r.
For $i = 1$ to t do
  Generate a random $a \in [2, n-2]$. Let $\beta \leftarrow a^r \mod n$.
  If $\beta \neq 1$ and $\beta \neq -1$ do
    $j \leftarrow 1$.
    While $j \leq s - 1$ and $\beta \neq -1$ do
      Let $\beta \leftarrow \beta^2 \mod n$
      If $\beta = +1$ return "composite"
      $j \leftarrow j + 1$
    If $\beta \neq -1$ return "composite"
Return "prime"

The Miller-Rabin test

- Property
  - If n is prime, then the Miller-Rabin test always declares n as prime.
  - If $n \geq 3$ is composite, then the probability that the Miller-Rabin test outputs "prime" is less than $\left(\frac{1}{4}\right)^t$
- Most widely used test in practice.
  - With $t = 40$, error probabitility less than 2^{-80}. Much less than the probability of a hardware failure.
  - Can test the primality of a 512-bit integer in less than a second.
  - Complexity: $\mathcal{O}(\log^3 n)$

Prime number generation

- To generate a prime integer of size ℓ bits
  - Generate a random integer n of size ℓ bits
  - Test its primality with Miller-Rabin.
  - If n is declared prime, output n, otherwise generate another n again.