AI2615 算法设计与分析 2020-2021 春季学期

Lecture 2 – Algorithms with numbers

2021 年 3 月 5 日

Lecturer: 张驰豪 *Scribe:* 张宇昊

1 What Is Cryptography?

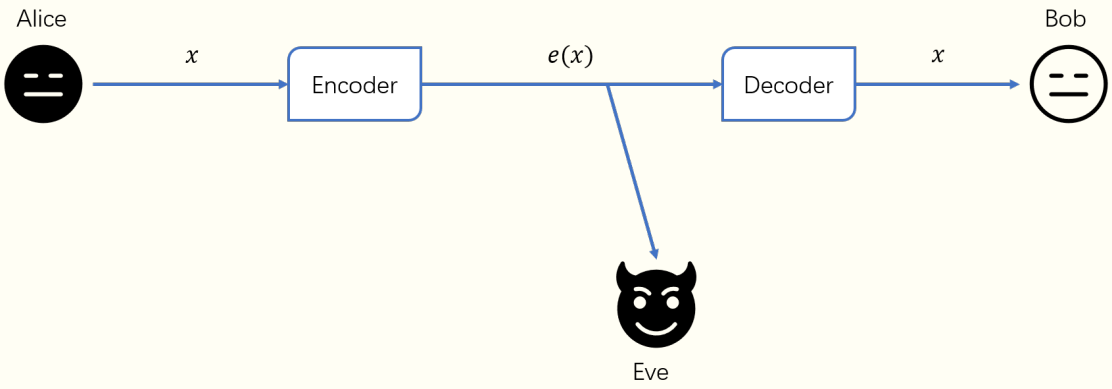

Figure 1: Alice: How to send message x to Bob, such that Eve can only get $e(x)$ but Bob can receive x?

Method 1: Private Key Scheme

Intuition Alice and Bob share the same private key, then they encode and decode the message via this private key.

Algorithm 1 One Time Pad

Alice: Choose a private key r and send it to Bob. (secure channel)

Alice: Encode message x: $e(x) = x \oplus r$.

Alice: Send $e(x)$ to Bob. (insecure channel).

Bod: Decode $e(x)$: $y = d(e(x)) = e(x) \oplus r = x$.

Remark The one time pad scheme is efficient because we can realize $\oplus$ efficiently, and it is secure if 1) we have a secure channel to send the private key, 2) and we use a new key before each communication. (very hard to realize)

Method 2: Public Key Scheme

Intuition Bob has a private key and a public key, and the public key is known by everyone. Everyone can encode message via Bob's public key, but only Bob can decode it by his private key. We needn't generate many keys and we also do not need a secure channel.

Algorithm 2 RSA (Rivest–Shamir–Adleman)

(Set-up) Bob: Choose two big (n-bit) prime p, q, and set $N = p \cdot q$.

(Set-up) Bob: Choose a number e which is relatively prime to $(p-1)(q-1)$.

(Set-up) Bob: Get d from $ed \equiv 1 \mod (p-1)(q-1)$.

(Set-up) Bob: Set d to be the private key. Set e to be the public key and send it to public.

Alice: Encode message x: $e(x) = x^e \mod N$.

Alice: Send $e(x)$ to Bob. (insecure channel).

Bod: Decode $e(x)$: $y = d(e(x)) = e(x)^d \mod N$.

Questions

1. Correctness: Does y equal to x?
2. Security: Is it hard to break? – Conjecture: we can not factor a big number efficiently.
3. Efficiency: Can we realize it efficiently?

Lemma 1 (Correctness). $y = d(e(x)) = x$

Proof. Because $ed \equiv 1 \mod (p-1)(q-1)$, there exists an integer k such that $ed = k(p-1)(q-1)+1$. We want to show that

$$d((e(x))) - x = x^{ed} - x = x^{k(p-1)(q-1)+1} - x \equiv 0 \mod N.$$

By Fermat's little theorem,

$$x^{p-1} \equiv 1 \mod p, \quad x^{q-1} \equiv 1 \mod q.$$

Therefore,

$$x^{k(p-1)(q-1)} \equiv 1 \mod p, \quad x^{k(p-1)(q-1)} \equiv 1 \mod q.$$

That implies

$$x^{k(p-1)(q-1)+1} - x \equiv 0 \mod p, \quad x^{k(p-1)(q-1)+1} - x \equiv 0 \mod p.$$

Therefore, we conclude

$$x^{k(p-1)(q-1)+1} - x \equiv 0 \mod pq.$$

□

2 How to Realize These Schemes Efficiently? Basic Arithmetic Operations

Basic Arithmetic to n-bit Numbers

- Addition/Subtractions: $O(n)$
- Multiplication: $O(n^2)$ ($O(n \log n)$ using FFT)
- Modular Division
- Primal Testing

2.1 Modular Division

How to solve x in $ax \equiv b \mod N$?

Lemma 2. *There exists $x, y \in Z$, such that $ax + Ny = b$ if and only if $gcd(a, N)|b$.*

Proof. =>: Assume $ax + Ny = b$, we have $\gcd(a, N)|ax$ and $\gcd(a, N)|Ny$. It implies that $\gcd(a, N)|b$.
<=: Assume $\gcd(a, N)|b$ and define $g = \gcd(a, N)$. To prove that $ax + Ny = b$ has an integral solution, we only need to prove that $ax + Ny = g$ has an integral solution. Assume s to be the minimum positive integer such that $ax + Ny = s$ has an integral solution, we have

$$a = q_a s + r_a, \quad N = q_N s + r_N.$$

(q_a and q_n are the quotient, r_a and r_N are the reminder.) Then, we have

$$r_a = a - q_a(ax + Ny) = (1 - q_a x) \cdot a - (q_a y) \cdot N < s.$$

Because we have already defined s to be the minimum positive integer such that $ax + Ny = s$ has an integral solution, r_a can only be 0. Do the same thing to r_N,

$$r_N = N - q_N(ax + Ny) = (1 - q_N x) \cdot N - (q_N x) \cdot a < s.$$

We can also prove $r_N = 0$. Therefore, s is a common divisor of N and a, combining with $gcd(a, N)|s$, we have that $s = gcd(a, N)$.

□

How to Compute gcd(a,b)? The Euclid Algorithm.

Algorithm 3 Euclid

```
Function Euclid(a, b)
    if b = 0 return a
    return Euclid(b, a mod b)
EndFunction
```

Time Complexity $O(n^3)$.

Lemma 3. *if* $a \geq b \geq a \bmod b > 0$, *then* $gcd(a,b) = gcd(b, a \bmod b)$.

Proof.
1. Let $d = \gcd(b, a \bmod b)$, and $r = a \bmod b > 0$, $a = qb + r$. Because d is a divisor of b and r, it's easy to see that d is also a divisor of a, so d is a common divisor of a and b, and $d \leq \gcd(a,b)$.
2. Let $d' = \gcd(a,b)$, $r = a - qb > 0$, because d' is a divisor of a and b, it's easy to show that d' is also a divisor of r, so d' is a common divisor of b and r, and $d' \leq \gcd(b, a \bmod b) = d$.
3. Therefore, we conclude $d' = \gcd(a,b) = \gcd(b, a \bmod b) = d$.

□

Corollary 4. *Let* $Euclid(a,b) = gcd(a,b)$.

How to find x and y? The Extended-Euclid Algorithm.

Algorithm 4 Extended-Euclid

```
Function Ext-Euclid(a, b)
    if b = 0 return (1, 0, a)
    (x', y', d) = Ext-Euclid(b, a mod b)
    return (y', x' − ⌊a/b⌋ · y', d)
EndFunction
```

Lemma 5. *Let* $(x,y,d) = Ext\text{-}Euclid(a,b)$, $ax + by = d$ *and* $d = gcd(a,b)$.

Proof. Base case: when $a \bmod b = 0$, we have $x = y' = 0,\ y = 1 - 0 = 1, d = b$, which satisfies $ax + by = b = d$ and $d = \gcd(a,b)$.
Induction: Assume $x' \cdot b + y' \cdot (a \bmod b) = \gcd(b, a \bmod b)$, we have

$$x'b + y'(a - \lfloor a/b \rfloor \cdot b) = \gcd(b, a \bmod b).$$

Applying Lemma 3, we conclude

$$y' \cdot a + (x' - \lfloor a/b \rfloor) \cdot b = \gcd(a,b).$$

$(x = y',\ y = x' - \lfloor a/b \rfloor,\ d = \gcd(a,b).)$ □

2.2 Primal Testing

Let N be a big integer and $2 \leq a \leq N-1$, we have that:

Lemma 6. *If* N *is not Carmichael and co-prime, then more than half of* a *satisfy* $a^{N-1} \not\equiv 1 \mod N$.

Proof. We prove it in the following steps.

1. Carmichael number: $\forall b$ which are relatively prime to N, $b^{N-1} \equiv 1 \mod N$.
2. If N is not Carmichael, we can find a, which is relatively prime to N, such that $a^{N-1} \not\equiv 1 \mod N$.
3. For any b such that $b^{N-1} \equiv 1 \mod N$, $(ab)^{N-1} \not\equiv 1 \mod N$.
4. Because a is relatively prime to N, $(ab \mod N)$ are different numbers with different b, which concludes the lemma.

□

Remark There are not too many Carmichael numbers in a reasonable range.

A Ranomized Testing To test a big integer N, continue to randomly choose integer $2 \leq a \leq N-1$, calculate $d = a^{N-1} \mod N$. Return "no" when $d \neq 1$ in one test, and return "yes" if it passes all the tests.