15:853: Algorithms in the Real World
Fall 2019

Lecture 18

November 7 2019

Lecturer: Francisco Maturana *Scribe: Tianjun Ma*

18.1 Recap From Last Class

We covered the basic concept of cryptography and introduced several examples of private-key algorithms. We also started reviewing group theory, which supports the fundamental theory behind public-key algorithms, which we will be focusing on for this lecture.

18.2 Overview

In this lecture, we will cover the following topics:

- Group Theory
- Finite Groups for Public Key Cryptosystems
  - Definitions
  - Examples
  - The Euler Phi Function
  - Computational Complexity
- Public Key Cryptosystems
  - Diffie-Hellman Key Exchange
  - ElGamal
  - RSA

18.3 Group Theory Review

Details available in section **3.5** of the scribe note of Lecture 3.

18.4 Finite Groups for Public Key Cryptosystems

For the purpose of this lecture, we will only be focusing on finite groups (groups with finite number of elements). This section introduces some definitions, examples, and theories that serve as the basis for Public Key Cryptosystems.

18.4.1 Definitions

Given group $(G, *, I)$, denote

$$\underbrace{a * a * a \cdots a}_{j \text{ times}}$$

where $a \in G$ as a^j.

Order of Group

The order of $g \in G$ is the smallest positive integer m such that $g^m = I$.

Cyclic Groups

A group G is cyclic if there is a $g \in G$ such that the order of g is $|G|$, the size of the group.

Generator

An element $g \in G$ of order $|G|$ is called a generator or primitive element of the group G.

The Euler Phi Function

$$\phi(n) = |Z_n^*| = n \prod_{p|n} (1 - \frac{1}{p})$$

$p|n$ means we are considering prime numbers p such that p divides n. Therefore, if $n = pq$ and both p and q are primes, we will have

$$\phi(n) = pq(1 - \frac{1}{p})(1 - \frac{1}{q}) = (p-1)(q-1)$$

Fermat's Little Theorem

If p is a prime number, then for any integer a that is not divisible by p, $a^{p-1} \equiv 1 \mod p$

Fermat-Euler Theorem

$$a^{\phi(n)} = 1 \mod n \text{ for } a \in Z_n^*$$

So if $n = pq$, we will have

$$a^{(p-1)(q-1)} = 1 \mod n \text{ for } a \in Z_{pq}^*$$

We will later use this property to show that RSA works as expected.

18.4.2 Examples

We will introduce some finite groups based on modular arithmetic.

groups of positive integers modulo prime p

$$Z_p^* \equiv \{1, 2, 3, \cdots, p-1\}$$

where $\overset{*}{}_p$ means multiplication modulo p. We will denote this group as $(Z_p^*, \overset{*}{}_p)$. And we can show that this group satisfies the required properties for a group. What's more, since $|Z_p^*| = (p-1)$, by Fermat's Little Theorem, we have $a^{p-1} = 1 \mod p$, for all $a \in Z_p^*$. It was also proven that for all prime p, Z_p^* is cyclic.

multiplicative group modulo n

$$Z_n^* \equiv \{m : 1 \leq m < n, gcd(n, m) = 1\}$$

where $\overset{*}{}_p$ means multiplication modulo n. We will denote this group as $(Z_n^*, \overset{*}{}_p)$. And we can show that this group satisfies the required properties for a group.

18.4.3 Computional Complexity

Common Operations

- Multiplication: $a \cdot b \mod n$ can be done in $O(\log^2(n))$ bit operations or better
- Power: $a^k \mod n$ can be done in $O(\log^3 n)$ bit operations
- Inverse: $a^{-1} \mod n$ can be done in $O(\log^3 n)$ bit operations

Discrete Logarithms

If g is a generator of Z_n^*, then for all y there is a unique $x \mod \phi(n)$ such that $y = g^x \mod n$. We call x the discrete logarithm of y and we use represent it using $x = \log_g(y)$. The main assumption we make about discrete logarithms for them to be useful for cryptosystems is that finding x is as hard as prime factorization.

18.5 Public-Key Cryptosystems

18.5.1 Diffie-Hellman Key Exchange

The essense is for two parties A and B to agree on a secret through a public channel.

Explanation

The public channel is established using the following steps:

1. Making $(G, *)$ and a generator g public
2. Alice picks a and sends g^a to Bob

3. Bob picks b and sends g^b to Alice
4. Both Alice and Bob can then use their shared key g^{ab} to encrypt and decrypt messages between them

$g,\ g^a,\ g^b$ are visible to the public. Since finding the discrete logarithm is conjectured to be computationally infeasible, it's hard for Eve to know a, b or g^{ab}.

Question: What could go wrong with this protocol?

Person-in-the-middle attack could be conducted as Malloy could modify g^a and g^b so that Alice and Bob will not longer able to have the same share key.

18.5.2 ElGamal

This differs from Diffie-Hellman in that intead of A and B sharing a same key used for decoding, each of the parties have their own private keys to decode the message encrypted using their individual public keys, respectively.

Explanation

The encryption and decryption are carried out in the following steps:

1. The person who wants to receive messages chooses a group $(G, *)$ so that the discrete logarithm is hard to compute.
2. Find $\alpha \in G$, α being the generator for G.
3. Find $a \in Z_{|G|}$
4. Find $\beta \in \alpha^a$
5. Make (α, β) and some description of G public, keeping a as the private key
6. Then, anyone who wants to send the publisher message m can pick arbitrary $r \in Z_{|G|}$ and send $(y_1, y_2) = (\alpha^r, m \cdot \beta^r)$
7. The publisher then decode the message by computing $y_2 \cdot (y_1^a)^- 1 = (m \cdot \beta^r) \cdot (\alpha^{ra})^{-1} = m \cdot \beta^r \cdot (\beta^r)^{-1} = m$

And it's hard to get m without a.

18.5.3 RSA

Similar to ElGamal, in the RSA Public-Key Cryptosystem, each party have their own private keys and publish their public keys.

Explanation

If someone wants to receive encrypted messages that only he/she could decrypt, here are the steps to be followed:

1. find large primes, p and q that are of roughly the same size
2. compute $n = pq$ and $\phi(n) = (p-1)(q-1)$
3. choose $e \in Z^*_{\phi(n)}$
4. compute private key $d = e^{-1} \mod \phi(n)$
5. publish public key (e, n)

Then, after the public keys (e, n) are available, other parties can send messages $m \in Z_n$ to the owner of the public key by computing $c = m^e \mod n$.For the publisher to decode, he/she simply needs to compute $c^d \mod n$ to get m. We argue that $m = c^d \mod n$ since

$$\begin{aligned} D(c) &= c^d \mod n \\ &= m^{ed} \mod n \end{aligned}$$

According to Fermat-Euler Theorem, we have $a^{\phi(n)} \mod n = 1$. Therefore, $m^{ed} = m^{ed \mod \phi(n)}$.

$$\begin{aligned} D(c) &= m^{ed \mod \phi(n)} \mod n \\ &= m^{ee^{-1} \mod \phi(n)} \mod n \\ &= m \mod n \end{aligned}$$

Also notice that this mechanism works for all $m \in Z_n$.

Efficiency of RSA

We can estimate the efficiency by looking at the inidvidual steps we take during encryption and decrytion that involves computation:

1. we need to run primality test while finding primes p and q, but this is a one-time thing so it does not affect encryption and decryption computation
2. $e^{-1} \mod (p-1)(q-1)$ will take $\log^2(n)$ using Euclid's algorithm.
3. for encoding, m^e can be computed using the power function which takes $\log(e)\log^2(n)$ time.
4. for decoding, c^d can be computed using the power function which takes $\log(d)\log^2(n)$ time.

Therefore, selecting a small e at the beginning will make the encryption faster, and this approach is used in practice.

Security of RSA

To make sure our keys are secure enough, we need to attend to the following details:

- Need to make sure that $p-1$ and $q-1$ have large prime factors.
- p, q need to be large enough so that it's hard for attackers to factor them out and obtain d from (n, e)
- e cannot be too small
- There are specific attacks against plain RSA so padding should be incorporated all the time
- Person-in-the-Middle attack is also possible since under current scheme, we cannot guarantee the authenticity of published public keys. In practice, this is resolved via Certificates or a Web-of-Trust.