Algorithmic Number Theory and Public-key Cryptography

Course 6

Jean-Sébastien Coron

Université du Luxembourg

April 11, 2014

Summary

- Algorithmic number theory.
  - Generators of $\mathbb{Z}_p$
  - Discrete logarithm and applications.

Groups

- Definitions
  - A group G is *finite* if $|G|$ is finite. The number of elements in a finite group is called its *order*.
  - A group G is *cyclic* if there is an element $g \in G$ such that for each $h \in G$ there is an integer i such that $h = g^i$. Such an element g is called a generator of G.
  - Let G be a finite group and $a \in G$. The *order* of a is definded to be the least positive integer t such that $a^t = 1$.

- Facts
  - Let G be finite group and $a \in G$. The order of a divides the order of G.
  - Let G be a cyclic group of order n and $d|n$, then G has exactly $\phi(d)$ elements of order d. In particular, G has $\phi(n)$ generators.

Properties of $\mathbb{Z}_n^*$

- Definition of $\mathbb{Z}_n^*$
  - The set $\mathbb{Z}_n^*$ is the set of integers modulo n which are invertible modulo n.
  - The set $\mathbb{Z}_n^*$ is a group of order $\phi(n)$ for the operation of multiplication modulo n.
- Properties
  - $\mathbb{Z}_p^*$ for prime p is a cyclic group of order $p-1$.
  - There exists a generator $g \in \mathbb{Z}_p^*$ such that for all $\alpha \in \mathbb{Z}_p^*$, α can be written uniquely as $\alpha = g^x \mod p$ for $0 \leq x < p-1$.
  - The integer x is called the *discrete logarithm* of α to the base g, and denoted $\log_g \alpha$.

Finding a generator of $\mathbb{Z}_p^*$

- Finding a generator of $\mathbb{Z}_p^*$ for prime p.
  - The factorization of $p-1$ is needed. Otherwise, no efficient algorithm is known.
  - Factoring is hard, but it is possible to generate p such that the factorization of $p-1$ is known.
- Generator of $\mathbb{Z}_p^*$
  - $g \in \mathbb{Z}_p^*$ is a generator of $\mathbb{Z}_p^*$ if and only if $g^{(p-1)/q} \neq 1 \mod p$ for each prime factor q of $p-1$.
  - There are $\phi(p-1)$ generators of $\mathbb{Z}_p^*$

Finding a generator

- Let $q_1, \ldots q_r$ be the prime factors of $p-1$
  - 1) Generate a random $g \in \mathbb{Z}_p^*$
  - 2) For $i = 1$ to r do
    - Compute $\alpha_i = g^{(p-1)/q_i} \mod p$
    - If $\alpha_i = 1 \mod p$, go back to step 1.
  - 3) Output g as a generator of $\mathbb{Z}_p^*$
- Complexity:
  - There are $\phi(p-1)$ generators of $\mathbb{Z}_p^*$.
  - A random $g \in \mathbb{Z}_p^*$ is a generator with probability $\phi(p-1)/(p-1)$.
  - If $p - 1 = 2 \cdot q$ for prime q, then $\phi(p-1) = q - 1$ and this probability is $\simeq 1/2$.

Generating p and q

- Goal: generate p such that $p-1=2\cdot q$ for prime q.
  - Generate a random prime p.
  - Test if $q=(p-1)/2$ is prime. Otherwise, generate another p.

Discrete logarithm

- Let g be a generator of $\mathbb{Z}_p^*$
  - For all $a \in \mathbb{Z}_p^*$, a can be written uniquely as $a = g^x \bmod p$ for $0 \leq x < p - 1$.
  - The integer x is called the *discrete logarithm* of a to the base g, and denoted $\log_g a$.
- Computing discrete logarithms in $\mathbb{Z}_p^*$
  - Hard problem: no efficient algorithm is known for large p.
  - Brute force: enumerate all possible x. Complexity $\mathcal{O}(p)$.
  - Baby step/giant step method: complexity $\mathcal{O}(\sqrt{p})$.

Baby step/giant step method

- Given $a = g^x \bmod p$ where $0 \leq x < p-1$, we wish to compute x.
- Let $m = \lfloor \sqrt{p} \rfloor$. Build a table:

$$L = \left\{ (g^i \bmod p, i) \,|\, 0 \leq i < m \right\}$$

  and sort L according to the first component $g^i \bmod p$.
  - Size: $\mathcal{O}(\sqrt{p} \log p)$. Time: $\mathcal{O}(\sqrt{p} \log^2 p)$.
- Compute the sequence of values $a \cdot g^{-j \cdot m} \bmod p$, until a collision with g^i is found in the table L, which gives:

$$a \cdot g^{-j \cdot m} = g^i \bmod p \Rightarrow a = g^{j \cdot m + i} \bmod p \Rightarrow x = j \cdot m + i$$

- Time: $\mathcal{O}(\sqrt{p} \log^2 p)$. Memory: $\mathcal{O}(\sqrt{p} \log p)$

Discrete Logarithms in groups of order q^e

- Let p be a prime and g a generator of a subgroup of $\mathbb{Z}_p^*$ of order q^e for some q, where $e > 1$.
- Given $a = g^x \bmod p$ for $0 \leq x < q^e$, we wish to compute x.
- We write $x = u \cdot q + v$ where $0 \leq v < q$ and $0 \leq u < q^{e-1}$
  - $a^{q^{e-1}} = \left(g^{q^{e-1}}\right)^x = \left(g^{q^{e-1}}\right)^v \bmod p$
  - We compute v by using the previous method in the subgroup of order q generated by $g^{q^{e-1}}$
- $a \cdot g^{-v} = (g^q)^u$ so we compute u recursively, in the subgroup of order q^{e-1} generated by g^q.
- Time complexity $\mathcal{O}(e \cdot \sqrt{q} \cdot \log^2 p)$

Discrete Logarithms in $\mathbb{Z}_p^*$

- Let p be a prime and we know the factorization
$$p - 1 = \prod_{i=1}^{r} q_i^{e_i}$$
- Given $a = g^x \bmod p$ for $0 \le x < p - 1$ where g is a generator of $\mathbb{Z}_p^*$, we wish to compute x.
- For $1 \le i \le r$ we have:
$$a^{(p-1)/q_i^{e_i}} = \left(g^{(p-1)/q_i^{e_i}}\right)^x = \left(g^{(p-1)/q_i^{e_i}}\right)^{x \bmod q_i^{e_i}} \mod p$$
- We compute $x_i = x \bmod q_i^{e_i}$ for all $1 \le i \le r$ by using the previous method in the subgroup generated by $g^{(p-1)/q_i^{e_i}}$
- Using CRT we find x from the x_i's.
- Complexity $\mathcal{O}(\sqrt{q} \cdot \log^k p)$, where $q = \max q_i$
- The hardness of computing discrete logarithms in $\mathbb{Z}_p^*$ is determined by the size of the largest prime factor of $p - 1$.
  - In general we work in a subgroup of $\mathbb{Z}_p^*$ of prime order.

Diffie-Hellman protocol

- Enables Alice and Bob to establish a shared secret key that nobody else can compute, without having talked to each other before.
- Key generation
  - Let p a prime integer, and let g be a generator of $\mathbb{Z}_p^*$. p and g are public.
  - Alice generates a random x and publishes $X = g^x \bmod p$. She keeps x secret.
  - Bob generates a random y and publishes $Y = g^y \bmod p$. He keeps y secret.

Diffie-Hellman protocol

- Key establishment
  - Alice sends X to Bob. Bob sends Y to Alice.
  - Alice computes $K_a = Y^x \bmod p$
  - Bob computes $K_b = X^y \bmod p$

$$K_a = Y^x = (g^y)^x = g^{xy} = (g^x)^y = X^y = K_b$$

- Alice and Bob now share the same key $K = K_a = K_b$
  - Without knowing x or y, the adversary is unable to compute K.
  - Computing g^{xy} from g^x and g^y is called the *Diffie-Hellman problem*, for which no efficient algorithm is known.
  - The best known algorithm for solving the Diffie-Hellman problem is to compute the discrete logarithm of g^x or g^y.

El-Gamal encryption

- Key generation
  - Let G be a subgroup of $\mathbb{Z}_p^*$ of prime order q and g a generator of G.
  - Let $x \overset{R}{\leftarrow} \mathbb{Z}_q$. Let $h = g^x \bmod p$.
  - Public-key : (g, h). Private-key : x
- Encryption of $m \in G$:
  - Let $r \overset{R}{\leftarrow} \mathbb{Z}_q$
  - Output $c = (g^r, h^r \cdot m)$
- Decryption of $c = (c_1, c_2)$
  - Output $m = c_2/(c_1^x) \bmod p$

Security of El-Gamal

- To recover m from $(g^r, h^r \cdot m)$
  - One must find h^r from $(g, g^r, h = g^x)$
- Computational Diffie-Hellman problem (CDH) :
  - Given (g, g^a, g^b), find g^{ab}
  - No efficient algorithm is known.
  - Best algorithm is finding the discrete-log
- However, attacker may already have some information about the plaintext !

Chosen-ciphertext attack

- El-Gamal is not chosen-ciphertext secure
  - Given $c = (g^r, h^r \cdot m)$ where $pk = (g, h)$
  - Ask for the decryption of $c' = (g^{r+1}, h^{r+1} \cdot m)$ and recover m.
- The Cramer-Shoup encryption scheme (1998)
  - Can be seen as extension of El-Gamal.
  - Chosen-ciphertext secure (IND-CCA) without random oracle.

The Cramer-Shoup cryptosystem

- Key generation
  - Let G a group of prime order q
  - Generate random $g_1, g_2 \in G$ and randoms $x_1, x_2, y_1, y_2, z \in \mathbb{Z}_q$
  - Let $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, $h = g_1^z$
  - Let H be a hash function
  - $pk = (g_1, g_2, c, d, h, H)$ and $sk = (x_1, x_2, y_1, y_2, z)$
- Encryption of $m \in G$
  - Generate a random $r \in \mathbb{Z}_q$
  - $C = (g_1^r,\ g_2^r,\ h^r m,\ c^r d^{r\alpha})$
  - where $\alpha = H(g_1^r, g_2^r, h^r m)$

The Cramer-Shoup cryptosystem

- Decryption of $C = (u_1, u_2, e, v)$
  - Compute $\alpha = H(u_1, u_2, v)$ and test if :
  $$u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v$$
  - Output "reject" if the condition does not hold.
  - Otherwise, output :
  $$m = e/(u_1)^z$$
- INC-CCA security
  - Cramer-Shoup is secure secure against adaptive chosen ciphertext attack
  - under the decisional Diffie-Hellman assumption,
  - without the random oracle model.
- Decision Diffie-Hellman problem:
  - Given (g, g^x, g^y, z) where $z = g^{xy}$ if $b = 0$ and $z \leftarrow G$ if $b = 1$, where $b \leftarrow \{0, 1\}$, guess b.