Groups

Let G be a set. A *binary operation* on G is a function $\star$ from $G \times G$ into G. If g, h are elements of G, then we will write $g \star h$ for the result of the binary operation applied to g and g. $g \star h$ is an element of G.

Definition. A group is a set G with a binary operation $\star$ satisfying these properties:

Closure: $\forall g, h \in G,\ g \star h \in G$.

Identity: $\exists e \in G$ so that $e \star g = g \star e = g$ for all $g \in G$.

Inverse: $\forall g \in G\ \exists h \in G$ so that $g \star h = e = h \star g$.

Associative: $\forall f, g, h \in G\ (f \star g) \star h = f \star (g \star h)$.

When G is a finite set, we say G is a finite group and let $|G|$ denote the *order* of G, the number of elements in G.

A group is *abelian* or commutative if it also satisfies this property:

Commutative: $\forall g, h \in G,\ g \star h = h \star g$.

All of our groups will be abelian.

A group is called *additive* if the operation is $+$ rather than $\star$. In this case, the identity is written 0 and the inverse of g is written $-g$.

A group is called *multiplicative* if the operation is $\cdot$ rather than $\star$. In this case, the identity is written 1 and the inverse of g is written g^{-1}.

Examples:

The integers with addition ($+$). The identity is 0. The inverse of n is $-n$.

The set of positive real numbers with multiplication ($\cdot$). The identity is 1. The inverse of x is $1/x = x^{-1}$.

More examples of groups:

When $N \geq 2$ is an integer, the set $Z_N = \{0, 1, \ldots, N-1\}$ with addition modulo N, that is, $a + b := [(a + b) \bmod N]$. The identity is 0. The inverse of a is 0 if $a = 0$ and $N - a$ if $1 \leq a \leq N - 1$. Note: $|Z_N| = N$.

When $N \geq 2$ is an integer, the set $Z_N^\star = \{a | 1 \leq a < N, \gcd(a, N) = 1\}$ with multiplication modulo N, that is, $a \cdot b := [(a \cdot b) \bmod N]$. The identity is 1. The inverse of a is $a^{-1} \bmod N$ computed by the extended Euclidean algorithm, for example. Note: $|Z_N^\star| = \phi(N)$.

Group Exponentiation

Let g be an element of a group G. Let m be a positive integer.

If G is an additive group with operation $+$, define

$$mg = m \cdot g = \overbrace{g + \cdots + g}^{m \text{ times}}.$$

If G is a multiplicative group with operation $\cdot$, define

$$g^m = \overbrace{g \cdots g}^{m \text{ times}}.$$

Theorem. If G is a finite group with order $m = |G|$, then for any element $g \in G$ we have

(a) $mg = 0$ if G is additive and

(b) $g^m = 1$ if G is multiplicative.

Example: For $G = Z_N^\star$, this theorem is Euler's theorem.

Corollary. If G is a finite group with order $m = |G| > 1$, then for any element $g \in G$ and any integer i we have

(a) $ig = [i \bmod m]g$ if G is additive and

(b) $g^i = g^{[i \bmod m]}$ if G is multiplicative.

Example:

$$2^{1234567893} \bmod 11 =$$

$= 2^{1234567893 \bmod 10} \bmod 11 = 2^3 \bmod 11 = 8,$

since $|Z_{11}^\star| = \phi(11) = 10$.

Corollary. If G is a (multiplicative) finite group with order $m = |G| > 1$, and e is a positive integer, define $f_e : G \to G$ by $f_e(g) := g^e$. If $\gcd(e, m) = 1$ then f_e is a permutation (one-to-one and onto). If $d = [e^{-1} \bmod m]$ then f_d is the inverse function to f_e.

Example: This corollary with $G = Z_{pq}^\star$ explains why RSA works.

Definition If m is an integer > 1, then a *primitive root* modulo m is an integer g with $\gcd(g, m) = 1$ and so that the smallest positive integer e for which $g^e \equiv 1 \pmod{m}$ is $e = p - 1$.

A primitive root modulo m is a *generator* of $Z_m^\star$

Fact: A primitive root exists modulo m if and only if $m = 2$, $m = 4$, $m = p^k$ (a power of an odd prime) or $m = 2p^k$ (twice a power of an odd prime).

In the true converse of Fermat's Little Theorem:

Theorem. Let $p > 3$ be odd. If for every prime $q|p-1$ there exists an a such that $a^{p-1} \equiv 1 \pmod{p}$, but $a^{(p-1)/q} \not\equiv 1 \pmod{p}$, then p is prime.

if a single integer a works for all primes $q|p-1$, then a is a primitive root modulo p.

Discrete logarithms

Let g be a primitive root modulo a prime p. If $y \equiv g^x \pmod{p}$, then we say that x is the *discrete logarithm* of y with respect to g modulo p.

(Sometime we say this even when g is not a primitive root modulo p, and also sometimes when p is just a positive integer, not necessarily prime.)

When the modulus p is understood, we sometimes we just say x is the discrete logarithm of y with respect to g to mean $y \equiv g^x \pmod{p}$.

Write $x = \log_g y$ to mean x is the discrete logarithm of y with respect to g, that is, $y \equiv g^x \pmod{p}$.

Write $x = \log_g y$ to mean x is the discrete logarithm of y with respect to g, that is, $y \equiv g^x \pmod{p}$.

Discrete logarithms enjoy many properties similar to those of logarithms:

Theorem: For integers x, y, g and p, we have

1. $x \equiv g^{\log_g x} \pmod{p}$,

2. $x \equiv y \pmod{p}$ if and only if $\log_g x = \log_g y$,

3. $\log_g(xy) \equiv \log_g x + \log_g y \pmod{\phi(p)}$,

4. $\log_g(g^x) \equiv x \pmod{\phi(p)}$, and

5. $\log_g(y^x) \equiv x(\log_g y) \pmod{\phi(p)}$.

The Discrete Logarithm Problem

Given a prime p with primitive root g, and an integer y relatively prime to p, there is an integer x with $y \equiv g^x \pmod{p}$. The Discrete Logarithm Problem DLP is to find x. For some large primes p and some g, the DLP is believed to be hard for most y.

One can define discrete logarithms for groups:

Definition: Let G be a finite group generated by g, that is,

$$G = \{g^0, g^1, \ldots, g^{m-1}\},$$

where $m = |G|$. The *discrete logarithm* of an element $h \in G$ is the number $x = \log_g h$ for which $h = g^x$. x is defined only modulo m.

The Discrete Logarithm Problem DLP in G is to compute $x = \log_g h$ for any given $h \in G$.

The El-Gamal public-key cipher

The **ElGamal public key cryptosystem** is defined as follows: Fix a large prime p which is public. Also public is a primitive root g modulo p in $1 < g < p$. Each user A who wishes to participate in this public-key cryptosystem chooses a secret a_A in $0 < a_A < p-1$ and publishes $b_A = g^{a_A} \bmod p$. When a user B wants to send a secret message M in $0 < M < p$ to A, she chooses a random k in $0 < k < p-1$ and sends to A the pair

$$C = (g^k \bmod p, (Mb_A^k) \mod p).$$

The plaintext M is enciphered by multiplying it by b_A^k in the second component of C. Note that $b_A^k \equiv (g^{a_A})^k \equiv g^{a_A k} \pmod{p}$. The first component of C provides a hint for deciphering M from the second component of C, but one which is useful only to A. Only A knows the secret key a_A, so only A can compute $(g^k)^{a_A} \equiv g^{a_A k} \pmod{p}$. If the multiplicative inverse of this number is multiplied times the second component, one recovers M:

$$\left(g^{a_A k}\right)^{-1} \left(M b_A^k\right) \equiv \left(g^{a_A k}\right)^{-1} \left(M g^{a_A k}\right) \equiv M \pmod{p}.$$

An eavesdropper who could solve the discrete logarithm problem modulo p could compute M from C and public data without knowing a_A as follows. The first component of C is $h = g^k \bmod p$. This number and $T = (Mb_A^k) \bmod p$ are observed by the eavesdropper. The eavesdropper knows p and g because these numbers are public. He can also obtain A's public key b_A from A's directory, just as B did. He would solve the discrete logarithm problem $g^k \equiv h \pmod{p}$ for k and then compute

$$T\left(b_A^k\right)^{-1} \equiv \left(Mb_A^k\right)\left(b_A^k\right)^{-1} \equiv M \pmod{p}.$$

Pohlig-Hellman cipher

This is NOT a public-key cipher.

Let $n = p =$ prime. Then $\phi(p) = p - 1$ and $ed \equiv 1 \pmod{p-1}$.

Method 1: Keep all of p, e, d secret. All three are the "key". There is just one user or one pair of users.

Encipher: $C = M^e \bmod p$.

Decipher: $M = C^d \bmod p$.

Method 2: Let p be public and keep e and d secret. The key is the pair (e, d). Each user has a secret pair to safeguard her personal secrets. Each pair of users who wish to communicate choose a key pair.

Since it may take a while to generate a large prime, Method 2 is more common than Method 1. Furthermore, Method 2 has interesting mathematical properties which foster its use in special ways discussed later (Massey-Omura, mental poker).

Cryptanalysis: For a known-plaintext attack on Method 2, one is given a prime p, C and M, and must find an exponent e so that $C \equiv M^e \pmod{p}$ or (equivalently) d so that $M \equiv C^d \pmod{p}$.

The Massey-Omura public-key cipher

One can change the Pohlig-Hellman private-key cipher slightly to make a public-key cipher. This was done by Massey and Omura. Their system is not used much because it is inefficient. (But the elliptic curve version is used.)

Consider a Pohlig-Hellman cipher with common prime p. This was called Method 2 earlier. Suppose users A and B have encryption algorithms E_A and E_B and decryption algorithms D_A and D_B. (So $E_A(M) = M^{e_A} \bmod p$, $D_A(C) = C^{d_A} \bmod p$, where $e_A d_A \equiv 1 \pmod{p-1}$, etc.) Since the encryption and decryption algorithms are all exponentiation modulo a fixed modulus, they all *commute*, that is, they may be done in any order and give the same result. For example, $E_A(D_B(x)) = D_B(E_A(x))$ for every x because both are just $x^{e_A d_B} \equiv x^{d_B e_A} \bmod p$.

How do A and B use this property as a public-key cipher? The "public key" is the common prime modulus p. The private keys are ALL of the exponents (unlike RSA). If Alice wants to send a message $0 < M < p$ to Bob, she first sends $E_A(M)$ to Bob. Bob replies by sending $E_B(E_A(M))$ to Alice. Then Alice sends

$$D_A(E_B(E_A(M))) = E_B(D_A(E_A(M))) = E_B(M)$$

to Bob. Bob deciphers the message by applying D_B to it. The security depends on the difficulty of the Discrete Logarithm Problem. The system is a protocol which requires a two-way exchange of three messages–not impossible, but still less convenient than RSA or El-Gamal in which just one message is sent.

The Diffie-Hellman key-exchange protocol

This protocol allows two users to choose a common secret key, for DES or AES, say, while communicating over an insecure channel (with eavesdroppers).

The two users agree on a common large prime p and a constant value a, which may be publicly known and available to everyone. It is best if the smallest exponent $e > 0$ for which $a^e \equiv 1 \pmod{p}$ is $e = p-1$, but the protocol will work if $e < p-1$ provided e is still large.

Alice secretly chooses a random x_A in $0 < x_A < p-1$ and computes $y_A = a^{x_A} \bmod p$. Bob secretly chooses a random x_B in $0 < x_B < p-1$ and computes $y_B = a^{x_B} \bmod p$.

Alice sends y_A to Bob. Bob sends y_B to Alice. An eavesdropper, knowing p and a, and seeing y_A and y_B, cannot compute x_A or x_B from this data unless he can solve the Discrete Logarithm Problem quickly.

Alice computes $K_A = y_B^{x_A} \bmod p$. Bob computes $K_B = y_A^{x_B} \bmod p$.

Then

$$K_A \equiv a^{x_A \cdot x_B} \equiv K_B \pmod{p}$$

and $0 < K_A, K_B < p$, so $K_A = K_B$.

Alice and Bob choose certain agreed-upon bits from K_A to use as their key for a single-key cipher like DES or AES.

Although this protocol provides secure communication between Alice and whomever is at the other end of the communication line, it does not prove that Bob is the other party. To guarantee that Bob is at the other end, they would have to use a signature system like RSA.

Discrete Logarithms

The Diffie-Hellman key exchange, the ElGamal public key cryptosystem, the Pohlig-Hellman private key cryptosystem and the Digital Signature Algorithm could all be broken if we could compute discrete logarithms quickly, that is, if we could solve the congruence $a^x \equiv b \bmod p$ easily.

Neglecting powers of $\log p$, the congruence may be solved in $O(p)$ time and $O(1)$ space by raising a to successive powers modulo p and comparing each with b. It may also be solved in $O(1)$ time and $O(p)$ space by looking up x in a precomputed table of pairs $(x, a^x \bmod p)$ sorted by the second coordinate.

Shanks' "giant step–baby step" algorithm is a meet-in-the-middle method which solves the congruence in $O(\sqrt{p})$ time and $O(\sqrt{p})$ space as follows. Let $m = \lceil \sqrt{p-1} \rceil$. Compute and sort the m ordered pairs $(j, a^{mj} \bmod p)$, for j from 0 to $m-1$, by the second coordinate. Compute and sort the m ordered pairs $(i, ba^{-i} \bmod p)$, for i from 0 to $m-1$, by the second coordinate. Find a pair (j, y) in the first list and a pair (i, y) in the second list. This search will succeed because every integer between 0 and $p-1$ can be written as a two-digit number ji in radix m. Finally, $x = mj + i \bmod p - 1$.

Discrete Log Assumption

Let Group be a polynomial-time algorithm that, on input 1^n, outputs (G, g, q), where G is a cyclic group generated by $g \in G$ and with order $|G| = q$, where q is a number of n bits.

Discrete Logarithm experiment $\mathsf{DLog}_{\mathcal{A},\mathsf{Group}}(n)$:

1. Run Group(1^n) to get (G, g, q).

2. Choose $h \leftarrow G$ (Say, $x' \leftarrow Z_q$; $h := g^{x'}$.)

3. $\mathcal{A}$ is given G, g, q, h, and outputs $x \in Z_q$.

4. The output of the experiment is 1 if $g^x = h$, and 0 otherwise.

Definition: We say that the discrete logarithm problem is hard with respect to Group if for all PPT $\mathcal{A}$ there exists a negligible function negl such that

$$Pr[\mathsf{DLog}_{\mathcal{A},\mathsf{Group}}(n) = 1] \leq \mathsf{negl(n)}.$$

The *discrete logarithm assumption* is that there is a Group with respect to which the discrete logarithm problem is hard.

The Computational Diffie-Hellman Problem

Fix a cyclic group G and a generator $g \in G$. Given two group elements h_1 and h_2, define $\mathsf{DH}_g(h_1, h_2) := g^{\log_g h_1 \cdot \log_g h_2}$. This means that if $h_1 = g^x$ and $h_2 = g^y$, then

$$\mathsf{DH}_g(h_1, h_2) = g^{xy} = h_1^y = h_2^x.$$

The CDH problem is to compute $\mathsf{DH}_g(h_1, h_2)$ given randomly chosen h_1 and h_2.

If the discrete log problem relative to some Group is easy, then the CDH problem is easy for Group, too.

It is not clear whether the CDH problem must be hard when the discrete log problem is hard.

The Decisional Diffie-Hellman Problem

The DDH problem is to distinguish $\mathrm{DH}_g(h_1, h_2)$ from a random group element for randomly chosen h_1 and h_2.

If CDH is easy, then so is DDH.

The converse is probably false, however, because there are groups for which DDH is easy and CDH appears to be hard.

Generally speaking, CDH is harder in groups of prime order than in other groups with composite order of nearly equal size. This is so because in many groups G of size $q = m \cdot n$, the CDH in G can be reduced to solving two CDHs, one in a group of size m and one in a group of size n.

This fact explains why one uses a subgroup of $Z_q^\star$ with prime order rather than the whole group, whose order is the composite number $\phi(q)$.