Cryptography

~~Encryption~~ Based on Discrete Logarithm

Discrete logarithm

$\mathbb{Z}_p^\times$

base

$b^{ord(b)} = 1$

$y = b^{k}$

$dlog_b(y)$

Let G be a finite group, $b \in G$, and $k \in \overbrace{\{0, \ldots, ord(b) - 1\}}$. For $y = b^k$, the number k is called the discrete logarithm of y w.r.t. base b. For proper choices of G and b, it is believed to be difficult to compute the value k given the knowledge of G, b and b^k.

When focusing on discrete logarithms of base $b \in G$, we will be dealing with the values

$$\langle b \rangle = \{1, b, b^2, \ldots, b^{ord(b)-1}\},$$

$1 = b^0$

$b^i \cdot b^j = b^{i+j} = b^{i+j \pmod{ord(b)}}$

$b^i \cdot b^{ord(b)-i} = b^{ord(b)} = 1$

(where $ord(b)$ is the order of b in G).

The tuple $(\langle b \rangle, \cdot)$ is itself a group: a subgroup of ~~$\mathbb{Z}_N^*$~~ G. We call it a subgroup generated by b.

$\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$ $\quad \langle 2 \rangle = \{2, 4, 1\}$ $\quad \langle 3 \rangle = \{3, 2, 6, 4, 5, 1\}$

$\phi(6) = \{1, 5\}$

The difficulty of discrete logarithm of base b in G is determined by the size and the algebraic structure of $(\langle b \rangle, \cdot)$.

$y = b^k \quad b, b^2, b^3, \ldots, b^k$

$\mathbb{Z} \quad b \quad y = b^k$

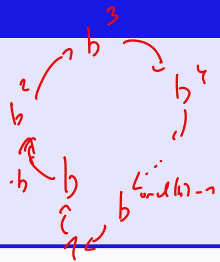
$k = 18 \quad b, b^2, \ldots, b^{16}, b^{32}$

Definition 1: Cyclic group

A group $(G, \cdot)$ is cyclic if there exists an element $b \in G$ s.t.

$$G = \{b^n \mid n \in \mathbb{N}\}.$$

The element b is called a generator of the group $(G, \cdot)$.



Cyclic groups

Definition 1: Cyclic group

A group $(G, \cdot)$ is cyclic if there exists an element $b \in G$ s.t.

$$G = \{b^n \mid n \in \mathbb{N}\}.$$

The element b is called a generator of the group $(G, \cdot)$.

Fact 1

$\forall p$ prime $\mathbb{Z}_p^\times$ is a cyclic group

If the group we want to work with is not cyclic (e.g. $\mathbb{Z}_N^\times$ for most non-prime choices of N, or certain elliptic curve groups), we use some cyclic sub-group of it that is given via it's generator. In the following, we assume that G is directly the cyclic group we work it.

Computing generators of cyclic groups

$\mathbb{Z}_p^\times$

Euler's totient function

Fact 2

Let $G = \boxed{\{1, b, b^2, \ldots, b^{m-1}\}}$ be a cyclic group of order m. Then G has $\phi(m)$ generators. I.e., the fraction of group elements that are its generators is $\frac{\phi(m)}{m}$. $\frac{\phi(m)}{m}$

$|\mathbb{Z}_m^\times|$

$b \quad b^k \quad \gcd(m, k) = 1$

$x, x^2, x^3, \ldots$

Computing generators of cyclic groups

Fact 2

Let $G = \{1, b, b^2, b^{m-1}\}$ be a cyclic group of order m. Then G has $\phi(m)$ generators. I.e., the fraction of group elements that are its generators is $\frac{\phi(n)}{n}$.

Fact 3

Let G be a cyclic group of order m. Given a prime factorization of m, one can efficiently compute a generator of G. (See Handbook of Applied Cryptography, algorithm 4.80)

Discrete logarithm in general cyclic groups

Definition 2: General discrete logarithm $\{1, b, b^2, b^3, \dots, b^{|G|-1}\}$ $b^k = y$

Let $(G, \cdot)$ be a finite cyclic group, $b \in G$ some generator of the group, and $k \in \{0, \dots, ord(b) - 1\}$. For $y = b^k$ the value k is called the discrete logarithm of y w.r.t. base b in G, written $k = dlog_b^G(y)$. For a proper choice of G and b it is believed to be difficult to compute, the value $dlog_b^G(y)$ given the knowledge of G, $|G|$, b, and $y = b^k$.

Discrete logarithm in general cyclic groups

Definition 2: General discrete logarithm

Let $(G, \cdot)$ be a finite cyclic group, $b \in G$ some generator of the group, and $k \in \{0, \ldots, ord(b) - 1\}$. For $y = b^k$ the value k is called the discrete logarithm of y w.r.t. base b in G, written $k = dlog_b^G(y)$. For a proper choice of G and b it is believed to be difficult to compute, the value $dlog_b^G(y)$ given the knowledge of $G, |G|, b$, and $y = b^k$.

The "proper choice" criteria include:

- $|G|$ should be sufficiently large, to prevent DL computation by bruteforcing or, e.g., the baby-step giant-step algorithm $\rightarrow O(\sqrt{|G|})$
- the operation of G should be efficiently computable
- $|G|$ should not be smooth, this is to defend against DL algorithms whose runtime is dominated by the term exponential in the bitsize of the largest prime factor of $|G|$ (Pohlig-Hellman algorithm, Pollard's ρ algorithm for discrete logarithm)

Discrete logarithm in general cyclic groups

Definition 2: General discrete logarithm

Let $(G, \cdot)$ be a finite cyclic group, $b \in G$ some generator of the group, and $k \in \{0, \ldots, ord(b) - 1\}$. For $y = b^k$ the value k is called the discrete logarithm of y w.r.t. base b in G, written $k = dlog_b^G(y)$. For a proper choice of G and b it is believed to be difficult to compute, the value $dlog_b^G(y)$ given the knowledge of $G, |G|, b$, and $y = b^k$.

The "proper choice" criteria include:

- $|G|$ should be sufficiently large, to prevent DL computation by bruteforcing or, e.g., the baby-step giant-step algorithm
- the operation of G should be efficiently computable
- $|G|$ should not be smooth, this is to defend against DL algorithms whose runtime is dominated by the term exponential in the bitsize of the largest prime factor of $|G|$ (Pohlig-Hellman algorithm, Pollard's ρ algorithm for discrete logarithm)

How to pick the right group?: The typical choices for G are $\mathbb{Z}_p^\times$ for a large prime p s.t. $p - 1$ is not smooth, or groups generated by elliptic curves (next lecture).

Diffie-Hellman key exchange

Suppose that Alice and Bob want to securely establish a shared symmetric key. The oldest public-key method for achieving this is the Diffie-Hellman key exchange scheme.

Diffie-Hellman key exchange

Suppose that Alice and Bob want to securely establish a shared symmetric key. The oldest public-key method for achieving this is the Diffie-Hellman key exchange scheme.

Setup: Alice and Bob agree in advance on a cyclic group G of order $|G|$, and on its generator b. Also, they fix a method for translating the elements of G into symmetric keys. This can be negotiated over an insecure channel.

Alice : α, b^α, b^β, b

Bob : β, b^β, b^α, b

Ad.: b^α, b^β

$b^\alpha b^\beta = b^{\alpha+\beta}$

Exchange:

- Alice randomly samples a number $\alpha \in \{1, \ldots, |G| - 1\}$ and sends $m_{Alice} = b^\alpha$ to Bob (keeping α secret).
- Bob randomly samples a number $\beta \in \{1, \ldots, |G| - 1\}$ and sends $m_{Bob} = b^\beta$ to Alice (keeping β secret).

Key derivation:

- Alice uses her knowledge of α to compute $k_{Alice} = m_{Bob}^\alpha$. $= (b^\beta)^\alpha = b^{\beta \cdot \alpha}$
- Bob uses his knowledge of β to compute $k_{Bob} = m_{Alice}^\beta$. $= (b^\alpha)^\beta = b^{\alpha \cdot \beta}$

Diffie-Hellman key exchange

Suppose that Alice and Bob want to securely establish a shared symmetric key. The oldest public-key method for achieving this is the Diffie-Hellman key exchange scheme.

Setup: Alice and Bob agree in advance on a cyclic group G of order $|G|$, and on its generator b. Also, they fix a method for translating the elements of G into symmetric keys. This can be negotiated over an insecure channel.

Exchange:

- Alice randomly samples a number $\alpha \in \{1, \dots, |G| - 1\}$ and sends $m_{Alice} = b^\alpha$ to Bob (keeping α secret).
- Bob randomly samples a number $\beta \in \{1, \dots, |G| - 1\}$ and sends $m_{Bob} = b^\beta$ to Alice (keeping β secret).

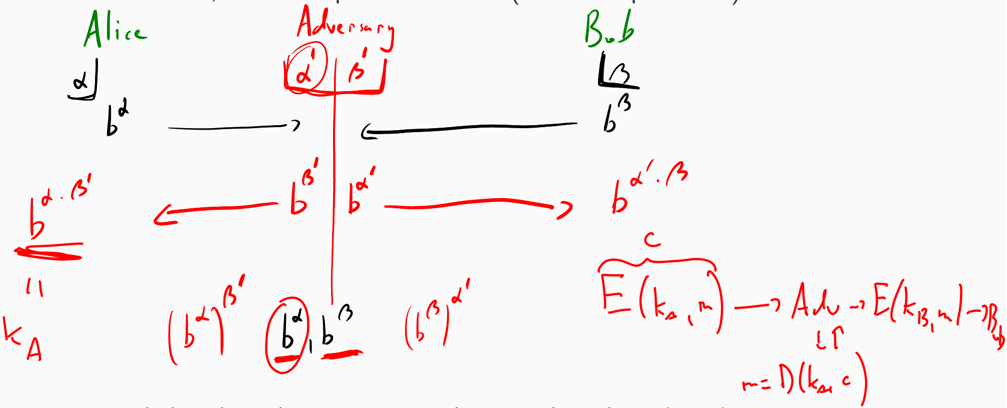
Then $k_{Alice} = (b^\alpha)^\beta = (b^\beta)^\alpha = k_{Bob}$!

Key derivation:

- Alice uses her knowledge of α to compute $k_{Alice} = m_{Bob}^\alpha$.
- Bob uses his knowledge of β to compute $k_{Bob} = m_{Alice}^\beta$.

Man-in-the-middle attack against Diffie-Hellman

DH key exchange is considered to be secure against passive adversaries. However, since it lacks any authentication mechanism, it is susceptible to active ("chosen ciphertext") attacks.



In practice, DH is extended with authentication mechanisms based on digital signatures to achieve a secure key establishment. (E.g. the STS - station-to-station - protocol).

Further remarks on discrete logarithm

- DL-based techniques are also heavily used to design digital signature methods (e.g. DSA, ECDSA – coming in a couple of weeks).

Further remarks on discrete logarithm

- DL-based techniques are also heavily used to design digital signature methods (e.g. DSA, ECDSA – coming in a couple of weeks).
- In practice, Diffie-Hellman and other DL techniques are used either with a $\mathbb{Z}_p^\times$ group for a suitable prime p or with a group generated by an elliptic curve. To achieve roughly the same level of security, much larger key sizes are required for the $\mathbb{Z}_p^\times$-based methods compared to the elliptic curve methods. Hence, there is a general trend in public-key crypto to switch to the elliptic curve techniques.

Further remarks on discrete logarithm

$y = b^k$ k b^k

SAT

$\boxed{\varphi}$

- DL-based techniques are also heavily used to design digital signature methods (e.g. DSA, ECDSA – coming in a couple of weeks).
- In practice, Diffie-Hellman and other DL techniques are used either with a $\mathbb{Z}_p^\times$ group for a suitable prime p or with a group generated by an elliptic curve. To achieve roughly the same level of security, much larger key sizes are required for the $\mathbb{Z}_p^\times$-based methods compared to the elliptic curve methods. Hence, there is a general trend in public-key crypto to switch to the elliptic curve techniques.
- A nice theoretical property of the discrete logarithm problem is its random self-reducibility: Suppose, that for a given group G and its generator b there exists an algorithm which efficiently computes $dlog_b^G(x)$ for a non-negligible fraction of possible inputs x. Then there exists an efficient algorithm for computing $dlog_b^G(x)$ for all $x \in G$. That is, an average instance of the DL problem has $\pm$ the same difficulty as the worst-case instance. Similar properties are unlikely to hold for, e.g. NP-complete problems.

$y \in G$ $dlog_b(y)$