

cyclic groups are commutative

defined to be the identity element

Definition. A group G is cyclic if there exists a generator g such that $G = \{g^0, g^1, \dots, g^{|G|-1}\}$.

Definition. For an element $g \in G$, we write $\langle g \rangle = \{g^0, g^1, \dots, g^{|G|-1}\}$ to denote the set generated by g (which need not be the entire set). The cardinality of $\langle g \rangle$ is the order of g (i.e., the size of the "subgroup" generated by g)

Example. Consider $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$. In this case,

$$\langle 2 \rangle = \{1, 2, 4\} \quad [2 \text{ is not a generator of } \mathbb{Z}_7^*] \quad \text{ord}(2) = 3$$

$$\langle 3 \rangle = \{1, 3, 2, 6, 4, 5\} \quad [3 \text{ is a generator of } \mathbb{Z}_7^*] \quad \text{ord}(3) = 6$$

\hookrightarrow means that $g^{\text{ord}(g)} = 1$

Lagrange's Theorem. For a group G , and any element $g \in G$, $\text{ord}(g) \mid |G|$ (the order of g is a divisor of $|G|$).

\hookrightarrow For \mathbb{Z}_p^* , this means that $\text{ord}(g) \mid p-1$ for all $g \in G$

Corollary (Fermat's Theorem): For all $x \in \mathbb{Z}_p^*$, $x^{p-1} = 1 \pmod{p}$

Proof. $|\mathbb{Z}_p^*| = |\{1, 2, \dots, p-1\}| = p-1$

\hookrightarrow for integer k

By Lagrange's Theorem, $\text{ord}(x) \mid p-1$ so we can write $p-1 = k \cdot \text{ord}(x)$ and so $x^{p-1} = (x^{\text{ord}(x)})^k = 1^k = 1 \pmod{p}$

Implication: Suppose $x \in \mathbb{Z}_p^*$ and we want to compute $x^y \in \mathbb{Z}_p^*$ for some large integer $y \gg p$

\hookrightarrow We can compute this as

$$x^y = x^{y \pmod{p-1}} \pmod{p}$$

since $x^{p-1} = 1 \pmod{p}$

\hookrightarrow Specifically, the exponents operate modulo the order of the group

\hookrightarrow Equivalently: group $\langle g \rangle$ generated by g is isomorphic to the group $(\mathbb{Z}_f, +)$ where $f = \text{ord}(g)$

$$\langle g \rangle \cong (\mathbb{Z}_f, +)$$

$$g^x \mapsto x$$

Notation: g^x denotes $\overbrace{g \cdot g \cdots g}^{x \text{ times}}$

g^{-x} denotes $(g^x)^{-1}$ [inverse of group element g^x]

$g^{x^{-1}}$ denotes $g^{(x^{-1})}$ where x^{-1} computed mod $\text{ord}(g)$ — need to make sure this inverse exists!

Computing on group elements: In cryptography, the groups we typically work with will be large (e.g., 2^{256} or 2^{1024})

- Size of group element (# bits): $\sim \log |G|$ bits (256 bits / 2048 bits)

- Group operations in \mathbb{Z}_p^* : $\log p$ bits per group element

addition of mod p elements: $O(\log p)$

multiplication of mod p values: naively $O(\log^2 p)$

Karatsuba $O(\log^{1.71} p)$

Schönhage-Strassen (GMP library): $O(\log p \log \log p \log \log \log p)$

best algorithm $O(\log p \log \log p)$ [2019]

\hookrightarrow not yet practical ($> 2^{4096}$ bits to be faster...)

exponentiation: using repeated squaring: $g, g^2, g^4, g^8, \dots, g^{\log p}$, can implement using $O(\log p)$

multiplications [$O(\log^3 p)$ with naive multiplication]

\hookrightarrow time/space trade-offs with more precomputed values

division (inversion): typically $O(\log^2 p)$ using Euclidean algorithm (can be improved)

Computational problems: in the following, let G be a finite cyclic group generated by g with order q

- Discrete log problem: sample $x \xleftarrow{r} \mathbb{Z}_q$

given $h = g^x$, compute x

- Computational Diffie-Hellman (CDH): sample $x, y \xleftarrow{r} \mathbb{Z}_q$

given g^x, g^y , compute g^{xy}

- Decisional Diffie-Hellman (DDH): sample $x, y, r \xleftarrow{r} \mathbb{Z}_q$

distinguish between (g, g^x, g^y, g^{xy}) vs. (g, g^x, g^y, g^r)

Each of these problems translates to a corresponding computational assumption:

Definition. Let $G = \langle g \rangle$ be a finite cyclic group of order q (where q is a function of the security parameter λ)

The DDH assumption holds in G if for all efficient adversaries A :

$$\Pr[x, y \xleftarrow{r} \mathbb{Z}_q : A(g, g^x, g^y, g^{xy}) = 1] - \Pr[x, y, r \xleftarrow{r} \mathbb{Z}_q : A(g, g^x, g^y, g^r) = 1] = \text{negl}(\lambda)$$

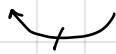
The CDH assumption holds in G if for all efficient adversaries A :

$$\Pr[x, y \xleftarrow{r} \mathbb{Z}_q : A(g, g^x, g^y) = g^{xy}] = \text{negl}(\lambda)$$

The discrete log assumption holds in G if for all efficient adversaries A :

$$\Pr[x \xleftarrow{r} \mathbb{Z}_q : A(g, g^x) = x] = \text{negl}(\lambda)$$

Certainly: if DDH holds in $G \Rightarrow$ CDH holds in $G \Rightarrow$ discrete log holds in G



there are groups where CDH
believed to be hard, but DDH is
easy



Major open problem: does this hold?

Can we find a group where discrete log is hard
but CDH is easy?

Instantiations: Discrete log in \mathbb{Z}_p^* when p is 2048-bits provides approximately 128-bits of security

\hookrightarrow Best attack is General Number Field Sieve (GNFS) - runs in time $2^{\tilde{O}(\sqrt[3]{\log p})}$ time

Much better than brute force - $2^{\log p}$

\hookrightarrow Need to choose p carefully

(e.g., avoid cases where $p-1$ is smooth)

for DDH applications, we usually set $p = 2q + 1$ where
 q is also a prime (p is a "safe prime") and work in the
subgroup of order q in \mathbb{Z}_p^* (\mathbb{Z}_p^* has order $p-1 = 2q$)

having small prime factors

group operations all
scale linearly (or worse) in
bitlength of the modulus

\uparrow cube root in exponent not ideal!
if we want to double security,
need to increase modulus by $8x$!
(e.g., 16384-bit modulus for 256 bits
of security)

Elliptic curve groups: only require 256-bit modulus for 128 bits of security

\hookrightarrow Best attack is generic attack and runs in time $2^{\log p/2}$

\hookrightarrow Much faster than using \mathbb{Z}_p^* : several standards

- NIST P256, P384, P512

- Dan Bernstein's curves: Curve 25519

\hookrightarrow Widely used for key-exchange + signatures on the web

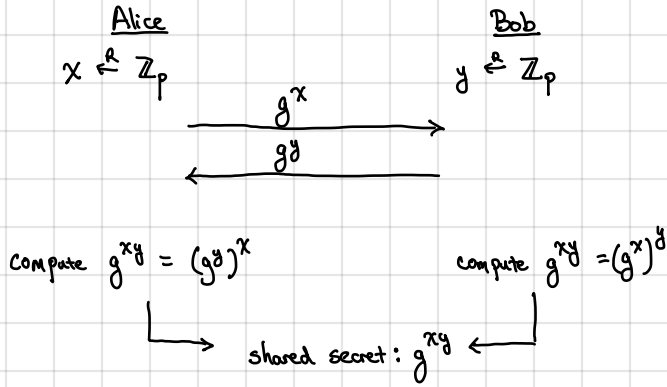
[p -algorithm - can discuss at end of semester]

} can discuss more at end of semester
(or in advanced crypto class)

When describing cryptographic constructions, we will work with an abstract group (easier to work with, less details to worry about)

Diffie-Hellman key exchange

- Let G be a group of prime order p (and generator g) - choice of group, generator, and order fixed by standard



But usually, we want a random bit-string as the key, not random group element

↳ Element g^x has $\log p$ bits of entropy, so should be able to obtain a random bit-string with $l < \log p$ bits

↳ Solution is to use a "randomness extractor"

↳ Information-theoretic constructions based on universal hashing / pairwise-independent hashing (loses some bits of entropy)

↳ Use a "random oracle" or an "ideal hash function" [Heuristic: $\text{SHA-256}(g, g^x, g^y, g^{xy})$] [binds the key to the entire transcript]

↳ Arguing security: 1. Rely on HashDH assumption $(g, g^x, g^y, H(g, g^x, g^y, g^{xy})) \approx (g, g^x, g^y, r)$ where $H: G^4 \rightarrow \{0,1\}^n$ and $r \xleftarrow{R} \{0,1\}^n$

2. Model H as ideal hash function $H: G^4 \rightarrow \{0,1\}^n$ (i.e., random oracle) and rely on CDH in G [inability to evaluate H on $g^{xy} \Rightarrow$ output is random string]

Public-key encryption: Encryption scheme where encryption is public (does not require shared secrets)

- Setup $(1^\lambda) \rightarrow (pk, sk)$ [generates a public/private key-pair - also called KeyGen]

- Encrypt $(pk, m) \rightarrow c$

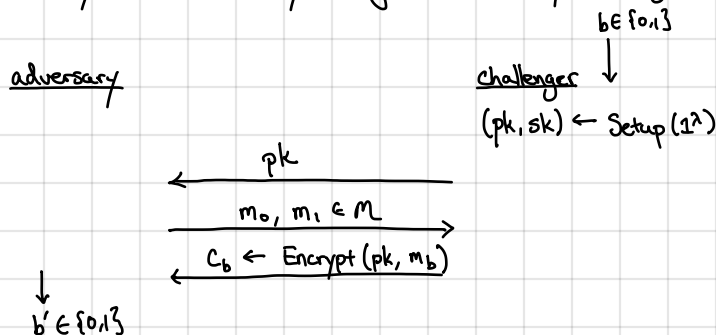
- Decrypt $(sk, c) \rightarrow m$

Everyone can publish a public key (in a directory)

↳ Can encrypt to anyone without exchanging keys (recipient can be offline)

Correctness: $\forall m \in \mathcal{M}: \Pr[(pk, sk) \leftarrow \text{Setup}(1^\lambda) : \text{Decrypt}(sk, \text{Encrypt}(pk, m)) = m] = 1$

Security: semantic security from secret-key setting, but adversary also gets public key



$$\text{SSAdv}[A, \Pi_{\text{PKE}}] = \left| \Pr[A \text{ outputs } 1 \mid b=0] - \Pr[A \text{ outputs } 1 \mid b=1] \right|$$