Definition. A group \( G \) is cyclic if there exists a generator \( g \) such that \( G = \{ g^0, g^1, ..., g^{\text{ord}(g)} \} \).

Definition. For an element \( g \in G \), we write \( \langle g \rangle = \{ g^0, g^1, ..., g^{\text{ord}(g)} \} \) 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, 
\[
\langle 2 \rangle = \{1, 2, 4, \} \quad \text{[2 is not a generator of } \mathbb{Z}_7^* \}\]
\[
\langle 3 \rangle = \{1, 3, 2, 6, 4, 5\} \quad \text{[3 is a generator of } \mathbb{Z}_7^* \}\]

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

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

Proof. \( |\mathbb{Z}_p^*| = |\{1, 2, ..., p-1\}| = p-1 \) for integer \( p \).

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 \( y \gg p \).

\( \rightarrow \) We can compute this as \( x^y = x^{y \cdot (p-1)} \pmod{p} \).

Since \( x^{p-1} \equiv 1 \pmod{p} \),

\( \rightarrow \) Specifically, the exponents operate modulo the order of the group.

\( \rightarrow \) Equivalently: group \( \langle g \rangle \) generated by \( g \) is isomorphic to the group \( \langle \mathbb{Z}_p^*, + \rangle \) where \( g = \text{ord}(g) \).

\( \langle g \rangle \cong \langle \mathbb{Z}_p^*, + \rangle \)

\( g^x \mapsto x \).

Notation: \( g^x \) denotes \( g \cdot g \cdot ... \cdot g \).

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

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

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): \( \approx \log \text{l} \text{bb bits (} 2^{256} \text{ bits / } 2^{1024} \text{ bits} \)

- 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: namely \( O(\log^2 p) \),

  Karatsuba: \( O(\log^3 p) \).

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

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

  \( \rightarrow \) not yet practical (\( > 2^{1966} \) bits to be faster...)

  exponentiation: using repeated squaring: \( g, g^2, g^4, g^8, ..., g^{2^{1024}} \), can implement using \( O(\log p) \) multiplications \( [O(\log^2 p) \text{ with naive multiplication}] \).

  \( \rightarrow \) time/space trade-offs with more precomputed values

  division (inversion): typically \( O(\log p) \) using Euclidean algorithm (can be improved).
- **Discrete log problem**: sample \( x \in \mathbb{Z}_p \),
given \( h = g^x \), compute \( x \)
- **Computational Diffie-Hellman (CDH)**: sample \( x, y \in \mathbb{Z}_p \),
given \( g^x, g^y \), compute \( g^{xy} \)
- **Decisional Diffie-Hellman (DDH)**: sample \( x, y, r \in \mathbb{Z}_p \),
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 \( G = (g) \) be a finite cyclic group of order \( q \) (where \( q \) is a function of the security parameter \( \lambda \))

The DDH assumption holds in \( G \) if for all efficient adversaries \( A \):
\[
\Pr [x, y, r \in \mathbb{Z}_q : A(g, g^x, g^y, g^r) = 1] = \negl(\lambda)
\]

The CDH assumption holds in \( G \) if for all efficient adversaries \( A \):
\[
\Pr [x, y \in \mathbb{Z}_q : A(g, g^x, g^y) = g^{xy}] = \negl(\lambda)
\]

The discrete log assumption holds in \( G \) if for all efficient adversaries \( A \):
\[
\Pr [x \in \mathbb{Z}_q : A(g, g^x) = x] = \negl(\lambda)
\]

Certainly: if DDH holds in \( G \) \( \implies \) CDH holds in \( G \) \( \implies \) 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?

**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

Alice
\[
\begin{align*}
x & \in \mathbb{Z}_p \\
g^x & \in \mathbb{Z}_p
\end{align*}
\]

Bob
\[
\begin{align*}
y & \in \mathbb{Z}_p \\
g^y & \in \mathbb{Z}_p
\end{align*}
\]

\[
\begin{align*}
\text{compute } g^{xy} & = (g^x)^y \\
\text{compute } g^{xy} & = (g^y)^x
\end{align*}
\]

\[
\text{shared secret: } g^{xy} \leftarrow
\]

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

\( \Leftarrow \) Element \( g^x \) has \( \log p \) bits of entropy, so should be able to obtain a random bit-string with \( \lambda < \log p \) bits

\( \Leftarrow \) Solution is to use a "randomness extractor"

\( \Leftarrow \) Information-theoretic constructions based on universal hashing / pairwise-independent hashing (loses some bits of entropy)
Use a “random oracle” or an “ideal hash function” \( \text{Heuristic: SHA-256 (} g, g^x, g^y, g^z) \) \[ \text{binds the key to the entire transcript} \]

- Arguing security: 1. Rely on HashDH assumption \( (g, g^x, H(g, g^y, g^z, g^w))^\ast \approx (g, g^x, g^y, r) \)
  - where \( H: G \to \{0,1\}^n \) and \( r \in \{0,1\}^n \)

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

**Instantiations:**
- Discrete log in \( Z_p^* \) when \( p \) is 2048-bits provides approximately 128-bits of security
  - Best attack is General Number Field Sieve (GNFS) - runs in time \( 2^{64(1.57)} \) time
  - Much better than brute force - \( 2^{127} \)
  - Need to choose \( p \) carefully (e.g., avoid cases where \( p-1 \) is smooth)
    - having small prime factors
    - group operations all scale linearly (or worse) in \( \log \text{bitlength of the modulus} \)

- Elliptic curve groups: only require 256-bit modulus for 128 bits of security
  - Best attack is generic attack and runs in time \( 2^{57.1} \) \( \beta \)-algorithm - can discuss at end of semester
  - Much faster than using \( Z_p^* \): several standards
    - NIST P256, P384, P512
    - Dan Bernstein’s curves: Curve 25519 (or in advanced crypto class)

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

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