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

- **Discrete log problem**: \( \text{sample } x \in \mathbb{Z}_q \)
  given \( h = g^x \), compute \( x \)

- **Computational Diffie-Hellman (CDH)**: \( \text{sample } x,y \in \mathbb{Z}_q \)
  given \( h_1 = g_1^x, h_2 = g_2^y \), compute \( g_1^y, g_2^x \)

- **Decisional Diffie-Hellman (DDH)**: \( \text{sample } x,y \in \mathbb{Z}_q \)
  distinguish between \( (g^x, g^y, g^{x+y}) \) vs. \( (g, g^x, g^y, g^z) \)

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 \( \lambda \))

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

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

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

\[
\Pr[x, y \in \mathbb{Z}_q : A(g^x, g^y, g^y) = g^{x+y}] = \text{negl}(\lambda)
\]

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

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

**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**
  - Best attack is General Number Field Sieve (GNFS) - runs in time \( \widetilde{O}(p^{1/3}) \) time
  - Much better than brute force - \( 2^{128} \)
  - Need to choose \( p \) carefully having small prime factors (e.g., avoid cases where \( p-1 \) is smooth)
  - \( p-1 \) is also a prime (\( p \) is a “safe prime”) and work in the subgroup of order \( q \) in \( \mathbb{Z}_p \) \( \mathbb{Z}_p^\times \) has order \( p-1 = 2q \)

  - 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 \( \mathbb{Z}_p^\times \) \( \mathbb{Z}_p^\times \) has order \( p-1 = 2q \)

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

- **Best attack is generic attack and runs in time \( 2^{128} \)** [\( \mathcal{g} \)-algorithm - can discuss at end of semester]
  - \( \mathcal{g} \)-algorithm - can discuss at end of semester
  - \( \text{NIST P256, P384, P521} \)
  - \( \text{Dan Bernstein's curves: Curve25519} \)
  - \( \text{can discuss more at end of semester} \)
  - \( \text{or in advanced crypto class} \)
  - \( \text{widely used for key exchange + signatures on the web} \)

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

\[
\begin{align*}
\text{Alice} & : x \in \mathbb{Z}_p \quad g^x \\
\text{Bob} & : y \in \mathbb{Z}_p \quad g^y
\end{align*}
\]

\[
\begin{align*}
\text{compute } \quad g^y \quad \text{compute } \quad g^x
\end{align*}
\]

\[
\begin{align*}
\quad \text{shared secret: } \quad g^{xy} \leftarrow
\end{align*}
\]

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

\( g^y \) has \( \log p \) bits of entropy, so should be able to obtain a random bit-string with \( k \leq \log p \) 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” \([\text{Heuristics: SHA-256}(g, g^x, g^y, g^{xy})]\)

Solution to use a “random oracle” or an “ideal hash function” \([\text{Heuristics: SHA-256}(g, g^x, g^y, g^{xy})]\)

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 \( H: \mathbb{G} \rightarrow \{0,1\}^n \) and \( r \in \{0,1\}^n \)

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

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

- Setup \( (1^n) \rightarrow (pk, sk) \) [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: \( \forall m \in M : \Pr[(pk, sk) \leftarrow \text{Setup}(1^n) : \text{Decrypt}(sk, \text{Encrypt}(pk, m)) = m] = 1 \)

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

\[
\begin{align*}
\text{adversary} & \quad \text{challenger} \downarrow \\
& \quad (pk, sk) \leftarrow \text{Setup}(1^n) \\
& \quad m_o, m, c_o \leftarrow M \\
& \quad C_o \leftarrow \text{Encrypt}(pk, m) \\
& \quad c \leftarrow \text{Encrypt}(pk, c)
\end{align*}
\]

SSAdv \([A, T_{\text{PKE}}] = \Pr[A \text{ outputs } 1 \mid b = 0] - \Pr[A \text{ outputs } 1 \mid b = 1] \)