1 Cyclic Groups & Cryptographic Applications

- Reading: Gallian Chapter 4.
- Classification of Subgroups of Cyclic Groups (Thms 4.2, 4.3) and Corollaries.
- Example: subgroups of $\mathbb{Z}$, $\mathbb{Z}_{12}$, $\mathbb{Z}_{13}^*$.

- Computations in a finite cyclic group $G = \langle g \rangle$ are easy if we can “access” the exponents.

- For what follows, let $G = \langle g \rangle$ a cyclic group of known order $q$ with a known generator $g$ in which group elements can be represented by bitstrings of length $n = O(\log_2 q)$, and where multiplication (given $a, b \in G$, compute $ab \in G$) and inversion (given $a \in G$, compute $a^{-1} \in G$) can be done in efficiently (in time poly$(n)$).

  - Example: $G = \mathbb{Z}_p^*$ for an $n$-bit prime $p$.

- **Exponentiation in** $G$: given $x \in \mathbb{Z}_q$, compute $g^x \in G$.
  
  - $O(\log q) = O(n)$ multiplications in $G$ (PS1).

- **Discrete Logarithm Problem in** $G$: given $a \in G$ (selected uniformly at random), compute the (unique) $x \in \mathbb{Z}_q$ such that $g^x = a$.

  - Naive algorithm: $O(q \log q) = O(n \cdot 2^n)$ multiplications in $G$.
  - PS2: $O(\sqrt{q} \log q) = O(2^{n/2} \cdot n)$ multiplications in $G$.
  - Best known for $G = \mathbb{Z}_p^*$: time $2^{O(n^{1/3} \log^{2/3} n)}$.
  - Best known for $G =$ “elliptic curve group”: time $O(2^{n/2})$.

- $\Rightarrow$ exponentiation seems to be a “one-way function,” easy in forward direction, hard in reverse direction.
- $\Rightarrow$ useful for cryptography (build algorithms that are easy to use but hard to break)
• **Diffie–Hellman Key Exchange Protocol** for parties $A$ (=Alice) and $B$ (=Bob) to generate a shared secret over an insecure communication line.

1. $A$ chooses random $x \in \mathbb{Z}_q$, computes $a = g^x$ and sends it to $B$.
2. $B$ chooses random $y \in \mathbb{Z}_q$, computes $b = g^y$ and sends it to $A$.
3. Both parties compute the shared key $k = g^{xy} = b^x = a^y$.

• **Security of Diffie–Hellman**

1. Eavesdropping adversary $E$ (=Eve) is given $a = g^x$ and $b = g^y$ and needs to compute $g^{xy}$. Not obvious how to do without computing discrete logarithms.
2. Usually want no information about $k$ to leak.
3. **Decisional Diffie–Hellman (DDH) Assumption**: No efficient (e.g. poly($n$)-time) algorithm $E$ can distinguish $(g^x, g^y, g^{xy})$ from $(g^x, g^y, g^z)$ for uniformly random and independent $x, y, z \in \mathbb{Z}_q$. That is, 
   \[ |\Pr[E(g^x, g^y, g^{xy}) = 1] - \Pr[E(g^x, g^y, g^z) = 1]| \leq \varepsilon(n), \]
   for some “negligible” function $\varepsilon(n) \to 0$.
4. DDH false if $q = |G|$ has small factors (PS 2), so usually take $|G|$ prime. For example, $G$ is the subgroup of order $q$ in $\mathbb{Z}_p^*$ where $p = 2q + 1$ and $p, q$ both primes.

• **El Gamal Public-Key Encryption Scheme**

1. $A$ chooses random $x \in \mathbb{Z}_q$, publishes $a = g^x$ as her public key, and keeps $x$ as her secret key.
2. To send a message $m \in G$, $B$ chooses random $y \in \mathbb{Z}_q$, sends $b = g^y$ and $c = m \cdot k = m \cdot g^{xy}$.
3. Given $(b, c)$, $A$ can recover $m$.

• **Security of El Gamal**

1. If DDH is true, then encryptions of any two messages $m_0, m_1 \in G$ are indistinguishable.
2. An encryption of $m_0$ is $(g^x, g^y, m_0 \cdot g^{xy})$, which is indistinguishable from $(g^x, g^y, m_0 \cdot g^z)$, which has the same distribution as $(g^x, g^y, g^w)$, where $x, y, w, z$ are all uniformly random and independent elements of $G$.
3. **Key fact**: Multiplying a fixed group element $(m_0)$ by a uniformly random group element $(g^z)$ gives a uniformly random group element $(g^w)$.