Cryptosystems Based on
Discrete Logarithms
Outline

• Discrete Logarithm Problem
• Cryptosystems Based on Discrete Logarithm
  – Encryption
  – Digital signature
Discrete logarithm problem (DLP)

• A group $G$ is **cyclic** if there is an element $\alpha \in G$ of order $|G|$. In this case, $G = \{\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{|G|-1}\}$; $\alpha$ is called a generator.

• Let $(G,*)$ be a finite group (not necessarily cyclic). Let $\alpha \in G$ be an element of order $n$. Then,
  
  $\langle \alpha \rangle = \{\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{n-1}\}$ is a cyclic (sub)group of order $n$.

• For any $y \in \langle \alpha \rangle$, there is a unique $x \in Z_n$ such that $\alpha^x = y$. This integer $x$ is called the discrete logarithm (or index) of $y$ with respect to base $\alpha$. We write $\log_\alpha y = x$.

• The DLP is to compute $\log_\alpha y$ for a given $y$. 
Frequently used settings

• $G = \mathbb{Z}_p^*$. $\langle \alpha \rangle = \{\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{p-2}\} = G$, where $p$ is a large prime, and $\alpha$ is a generator of $G$. ($\mathbb{Z}_p^*$ is cyclic when $p$ is prime.)

• $G = \mathbb{Z}_p^*$. $\langle \alpha \rangle = \{\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{q-1}\} \subset \mathbb{Z}_p^*$, where $\alpha \in \mathbb{Z}_p^*$ is an element of prime order $q$.

• Elliptic curves defined over finite fields.

• For these settings, there is no polynomial-time algorithm for DLP.
Example 1

\[ G = \mathbb{Z}_{19}^* = \{1, 2, \ldots, 18\} \].

2 is a generator. That is, \( \mathbb{Z}_{19}^* = \langle 2 \rangle \).

\[ 2^0 = 1, \ 2^1 = 2, \ 2^2 = 4, \ 2^3 = 8, \ 2^4 = 16, \ 2^5 = 13, \]
\[ 2^6 = 7, \ 2^7 = 14, \ldots \]
\[ \log_2 7 = 6 \]
\[ \log_2 14 = 7 \]
\[ \log_2 12 = ? \]
Example 2

\[ G = \mathbb{Z}_{11}^* = \{1, 2, \ldots, 10\}. \]

\[ G' = \langle 3 \rangle = \{1, 3, 9, 5, 4\}. \]

3 is a generator of \( G' \), but not a generator of \( \mathbb{Z}_{11}^* \).

\[ \log_3 5 = 3 \]

\[ \log_3 10 = \text{not defined} \]
DLP in $Z_p^*$

- Let $\alpha$ be a generator of $Z_p^*$ (a primitive root of unity modulo $p$).
- $Z_p^* = \{1, 2, \ldots, p - 1\} = \{\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{p-2}\}$.
- $Z_{p-1} = \{0, 1, 2, \ldots, p - 2\}$.
- Given $y \in Z_p^*$, find the unique $x \in Z_{p-1}$ such that $y = \alpha^x \mod p$.
- That is, given $\alpha^x \in Z_p^*$, find $x$.
- There is a subexponential-time algorithm for DLP in $Z_p^*$
  - Index Calculus, $O\left(2^{O(\sqrt{n \log n})}\right)$, where $n = \log p$. 
DLP in \((Z_n,+)
\)

Let \(\alpha\) be a generator of \(Z_n\) (any element coprime to \(n\)).

Given \(y \in Z_n\), find the unique \(x \in Z_{n-1}\) such that
\[
y = x\alpha \mod n.
\]

Easy or hard?
RSA vs. Discrete Logarithm

• RSA is a one-way trapdoor function:
  \[ x \xrightarrow{\text{RSA}} x^e \quad \text{(easy)} \]
  \[ x \xleftarrow{\text{RSA}^{-1}} (x^e)^d \quad \text{(d is a trapdoor)} \]

• Logarithm is the inverse of exponentiation:
  \[ x \xrightarrow{\exp_\alpha} \alpha^x \quad \text{(easy)} \]
  \[ x \xleftarrow{\log_\alpha} \alpha^x \quad \text{(difficult)} \]

• log is hard to compute, so exp is a one-way function, but without a trapdoor.

• An encryption scheme based on the difficulty of log will not simply encrypt \( x \) as \( \alpha^x \).
**Diffie-Hellman key agreement**

- $Z_p^* = \{\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{p-2}\}$. $Z_{p-1} = \{0, 1, 2, \ldots, p - 2\}$.
- Alice and Bob wish to set up a secret key.
  1. Alice and Bob agree on a large prime $p$ and a primitive root (generator) $\alpha \in Z_p^*$. ($p$, $\alpha$, not secret)
  2. Alice $\rightarrow$ Bob: $\alpha^a \mod p$, where $a \in R Z_{p-1}$.
  3. Alice $\leftarrow$ Bob: $\alpha^b \mod p$, where $b \in R Z_{p-1}$.
  4. They agree on the key: $\alpha^{ab} \mod p$.
- Diffie-Hellman problem: given $\alpha^a, \alpha^b \in Z_p^*$, compute $\alpha^{ab}$.
Ideas behind ElGamal encryption in $\mathbb{Z}_p^*$

0. Bob is to send a message $m$ to Alice, who has private key $x$ and public key $y := \alpha^x$.

1. Regard $m$ as an element in $\mathbb{Z}_p^*$.

2. Use Diffie-Hellman to set up a temporary key $key \in \mathbb{Z}_p^*$.
   - Bob generates $\alpha^k$ and computes $key := y^k (= \alpha^{xk})$.

3. Bob uses $key$ to encrypt $m$ as $c := m \cdot key$.

4. Bob sends $\alpha^k$ along with $c$ so that Alice can compute $key$. 
ElGamal encryption in $Z_p^*$

1. Key generation (e.g. for Alice):
   - choose a large prime $p$ and a primitive root $\alpha \in Z_p^*$, where $p - 1$ has a large prime factor.
   - randomly choose a number $x \in Z_{p-1}$ and compute $y = \alpha^x$;
   - set $sk = (p, \alpha, x)$ and $pk = (p, \alpha, y)$.

2. Encryption: $E_{pk}(m) = (\alpha^k, my^k)$, where $m \in Z_p^*$, $k \leftarrow_R Z_{p-1}$.

3. Decryption: $D_{sk}(a, b) = ba^{-x}$.

4. Remarks: All operations are done in $Z_p^*$, i.e., modulo $p$.
   The encryption scheme is non-deterministic.
Security of ElGamal encryption against CPA

- Based on the Diffie-Hellman assumption.
- Open problem: discrete logarithm ≤ Diffie-Hellman?

Theorem: If the Diffie-Hellman assumption is true, then the ElGamal encryption scheme is CPA-secure.
Security of ElGamal encryption against CCA

- A function \( f : G \rightarrow G' \) is **homomorphic** if \( f(xy) = f(x)f(y) \).

- ElGamal encryption is homomorphic, \( E(mm') = E(m) \cdot E(m') \), in the following sense:
  
  If \( E(m) = (\alpha^k, my^k) \) and \( E(m') = (\alpha^{k'}, m'y^{k'}) \), then \( E(m) \cdot E(m') = (\alpha^k, my^k) \cdot (\alpha^{k'}, m'y^{k'}) = (\alpha^{k+k'}, mm'y^{k+k'}) \) is a valid encryption of \( mm' \).

- As such, ElGamal encryption is **not CCA-secure** (i.e., not indistinguishable against CCA).
ElGamal signature in $\mathbb{Z}_p^*$

1. Key generation: same as in ElGamal encryption.
   - a large prime $p$ and a primitive root $\alpha \in \mathbb{Z}_p^*$.
   - a randomly chosen number $x \in \mathbb{Z}_{p-1}$ and $y = \alpha^x \mod p$;
   - $sk = (p, \alpha, x)$ and $pk = (p, \alpha, y)$.

2. To sign a message $m \in \mathbb{Z}_{p-1}$,
   - randomly choose $k \in \mathbb{Z}_{p-1}^*$;
   - compute $r = \alpha^k \mod p$ and $s = (m - rx)k^{-1} \mod (p - 1)$;
   - the signature is $S_{sk}(m) = (r, s) \in \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$.

3. Verification: $V_{pk}(m, r, s) = \text{true if and only if}$
   $$(m, r, s) \in \mathbb{Z}_{p-1} \times \mathbb{Z}_p^* \times \mathbb{Z}_{p-1} \text{ and } \alpha^m \equiv r^s y^r \mod p.$$
Ideas behind ElGamal signature

- $Z_p^* = \{\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{p-2}\}$. $Z_{p-1} = \{0, 1, 2, \ldots, p-2\}$.
- To sign a message $m$, Alice needs to show that she knows $x$. Besides, non-deterministic signature is desired. So, the signature should be a function of $(m, x, k)$, where $k$ is randomly selected.
- The secret key $x$ is an exponent; i.e., $x \in Z_{p-1}$. This suggests that $m$ and $k$ also be exponents, namely $m, k \in Z_{p-1}$.
- So, let the signature be a function of $(m, x, k)$ whose validity can be verified by using $\alpha^m, \alpha^x, \alpha^k$.
- The signer needs to send $\alpha^k$ along with the signature.
• The main part of ElGamal signature is some value $s$ satisfying $s = (m - rx)k^{-1} \mod (p - 1)$.

• To verify the validity of the signature, the verifier checks if the above condition is satisfied without knowing $x$ and $k$:

\[
s = (m - rx)k^{-1} \mod (p - 1)
\]

\[
\Rightarrow m = ks + rx \mod (p - 1)
\]

\[
\Rightarrow \alpha^m \equiv \alpha^{ks} \alpha^{rx} \mod p
\]

\[
\Rightarrow \alpha^m \equiv r^s y^r \mod p
\]

• Why $s = (m - rx)k^{-1}$?

• What about $s = km + k'x$, $s = km + rx$, $s = (m + rx)k^{-1}$, $s = (m + x)k^{-1}$? ($k'$ is another random number.)
Security of ElGamal signature

- Based on the assumed intractability of discrete logarithm.
- Knowing the random \( k \) implies knowing the secret key \( x \) with high probability: since \( s = (m - rx)k^{-1} \mod (p - 1) \), \( x = (m - sk)r^{-1} \mod (p - 1) \) if \( r^{-1} \mod (p - 1) \) exists.
- Use a new \( k \) for each signing, or the adversary can compute \( k \) from two signatures \( s = (m - rx)k^{-1} \) and \( s' = (m' - rx)k^{-1} \), by solving \( s - s' \equiv (m - m')k^{-1} \mod (p - 1) \).
Security of ElGamal signature (cont'd)

- Existential forgery. Construct a message $m$ and a valid signature $(r, s)$ as follows.
  a) choose $k, c \in \mathbb{Z}_{p-1}^*$.
  b) set $r = \alpha^k y^c \mod p$, $s = -rc^{-1} \mod (p - 1)$, and $m = -rkc^{-1} \mod (p - 1)$.

- Countermeasure: hash then sign.
Digital Signature Algorithm (DSA) - an NIST standard

0. Shnorr's idea: working in a subgroup of order $q \ll p$ will shorten the signature, desired for Smart Card applications.

- ElGamal signature scheme uses:

  \[ Z_p^* = \{ \alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{p-2} \} . \quad Z_{p-1} = \{ 0, 1, 2, \ldots, p-2 \} . \]

  A signature is \((r, s) \in Z_p^* \times Z_{p-1} \approx Z_{p-1} \times Z_{p-1} . \)

- DSA uses:

  \[ \langle g \rangle = \{ \alpha^0, \alpha^b, \alpha^{2b}, \alpha^{3b}, \alpha^{4b}, \ldots, \alpha^{(q-1)b} \} = \{ g^0, g^1, \ldots, g^{q-1} \} , \]

  where \( g = \alpha^b , \quad b = (p-1)/q . \)

  A signature is \((r, s) \in Z_q \times Z_q \)
1. Key generation

- choose two primes $p$ and $q$ such that $q \mid (p - 1)$. 
  $(q \ll p$, e.g., $|q| = 160, \ |p| = 1024.)$
- let $g \in Z_p^*$ be an element of order $q$.
- randomly choose $x \in Z_q^*$ and compute $y = g^x \mod p$;
- system parameters: $(h, p, q, g)$
- $sk = (x)$ and $pk = (y)$.

(Remark: The DLP will be working in $\langle g \rangle$.)


2. **Signing:** to sign a message $m$,
   - randomly choose $k \in \mathbb{Z}_q^*$; compute $r = (g^k \mod p) \mod q$.
   - compute $s = (h(m) + rx)k^{-1} \mod q$.
     (use a different $k$ if $r = 0$ or $s = 0$.)
   - $(m, r, s)$ is the signed message.

3. **Verification:** accept $(m, r, s)$ iff $r,s \in \mathbb{Z}_q^*$ and
   
   $$r = \left( (g^{h(m)} y^r)^t \mod p \right) \mod q, \text{ where } t = s^{-1} \mod q.$$
DSA Correctness: if the message is signed correctly, the signature will be verified/accepted.

\[ s = (h(m) + rx)k^{-1} \mod q \]

\[ \Rightarrow k = (h(m) + rx)t \mod q \quad (t = s^{-1} \mod q) \]

\[ \Rightarrow g^k \equiv (g^{h(m)} g^{rx})^t \mod p \quad \text{(since } g^q \equiv 1 \mod p) \]

\[ \Rightarrow g^k \equiv (g^{h(m)} y^r)^t \mod p \]

\[ \Rightarrow g^k \mod p = (g^{h(m)} y^r)^t \mod p \]

\[ \Rightarrow r = (g^k \mod p) \mod q = ((g^{h(m)} y^r)^t \mod p) \mod q \]
Remarks on DSA

- SHA-1 is suggested for use as the hash function.
- $|q| = 160$ bits, $|p| \leq 512 + 64t$, $0 \leq t \leq 8$.
- Because a hash is used, there is no restriction on $m$.
- Why not work in $Z_p^*$?
- $\langle g \rangle$ is a subgroup of $Z_p^*$ of order $q$.
- Shorter message length: $|(r, s)| = 2|q|$ bits, rather than $2|p|$.
- Why not work in $Z_q^*$? The message length would be $2|q|$, too.
- Reason: The Index Calculus method works for $Z_q^*$ but not for $\langle g \rangle$. 
The index calculus method for $\mathbb{Z}_p^*$

Factor base: $B = \{p_1, p_2, \ldots, p_b\}$, a set of small primes.

Ideas: Suppose $\log_\alpha p_j$ is known for all $j$, $1 \leq j \leq b$.

If $y \in \mathbb{Z}_p^*$ factors over $B$, say, $y = \prod_{i=1}^{b} p_i^{e_i}$, then

$$\log_\alpha y = \sum_{i=1}^{b} e_i \log_\alpha p_i \mod (p - 1).$$
Step 1: calculate $\log_{\alpha} p_j$, $1 \leq j \leq b$ as follows:

- choose $c > b$ random numbers $x_1, \ldots, x_c \in \mathbb{Z}_{p-2}$ such that

  $\alpha^{x_i}$ factors over $B$: $\alpha^{x_i} \equiv \prod_{j=1}^{b} p_j^{e_{i,j}} \mod p$, $1 \leq i \leq c$

- solve $x_i \equiv \sum_{j=1}^{b} e_{i,j} \log_{\alpha} p_j \mod (p-1)$, $1 \leq i \leq c$
Step 2: to compute $\log_\alpha y$, where $y \in Z_p^*$:

1. Choose $r \in \mathbb{R} \cdot Z_{p-1}$ and try to factor $y\alpha^r$ over $B$. Repeat this until success (or giving up).

2. Then we have $y\alpha^r = \prod_{i=1}^{b} p_i^{e_i} \mod p$, and

   $$\log_\alpha y + r \equiv \sum_{i=1}^{b} e_i \log_\alpha p_i \mod (p-1)$$

   $$\log_\alpha y = \left(-r + \sum_{i=1}^{b} e_i \log_\alpha p_i\right) \mod (p-1)$$