Today

Public-key Cryptography

- Basic group theory
- Diffie-Hellman key exchange
- Definition of public key cryptography
- El-Gamal encryption scheme

Group Theory - Recap

Definition: A group $G$ consists of a set of elements & an operation $\cdot : G \times G \rightarrow G$ s.t.
- $\forall a,b,c \in G \quad a \cdot (b \cdot c) = (a \cdot b) \cdot c$ (associative)
- $\exists$ identity $1 \in G$ s.t. $1 \cdot a = a \cdot 1 = a \quad \forall a \in G$
- $\forall a \in G \quad \exists a^{-1} \in G$ s.t. $a \cdot (a^{-1}) = 1$ (inverse)

A group is commutative if $\forall a,b \in G \quad a \cdot b = b \cdot a$

* All groups we will work with are Commutative.

Common Groups: $\mathbb{Z}_p$, $\mathbb{Z}_n$, $\mathbb{Q}_p$, $\mathbb{Q}_n$, Elliptic curves.
$$\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\} \text{ mult. mod } p$$

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid n-1 \} \text{ s.t. } \gcd(a, n) = 1$$

$n = p \cdot q$ product of 2 primes (used in RSA)

**Def:** The order of a group is the number of elements in the group

$$|\mathbb{Z}_p^*| = p-1$$

$$|\mathbb{Z}_n^*| = n-1 - (p-1) - (q-1) = n - p - q + 1 = (p-1)(q-1) \equiv \varphi(n)$$

**Note:** The order of $\mathbb{Z}_n^*$ is hard to compute given only $n$. $\varphi$ is used in security of RSA.

$\star$ For crypto applications we often need a group of prime order.

**Note:** $\mathbb{Z}_p^*$ is not prime order.

$$Q_p = \{a^2 : a \in \mathbb{Z}_p^*\}.$$
Claim: $|Q_p| = \frac{p-1}{2}$

Consider $f : \mathbb{Z}_p^* \rightarrow Q_p$ defined by $f(a) = a^2 \mod p$.

By Fundamental Theorem of Algebra every degree $d$ poly over a field $K$ has at most $d$ roots.

$\Rightarrow a^2$ has only two pre-images $a$, $p-a$.

(since $g(x) = x^3 - x$ is a deg 2 poly over the field $GF[p]$.)

(This is not true over $\mathbb{Z}_n^*$, since it is not a field.)

$\Rightarrow |Q_p| = \frac{p-1}{2}$

If $p$ is a prime $\Rightarrow g^\frac{p-1}{2}$ is prime then $|Q_p|$ is prime.

Such $p$ is called safe prime.

Recall: Exponentiation can be done efficiently (repeated squaring).

Inverses can be computed efficiently.
By Fermat's Little Thm: \( a^{p-1} = 1 \mod p \)
\[ \Rightarrow a^{-1} = a^{p-2} \mod p \]

- A prime (or safe prime) can be chosen efficiently
  by choosing a random element in \( \mathbb{Z}_p \)
  and testing if it is prime (or safe prime).

Order of Elements & Generators

\( \forall \) finite group \( G \), consider the subgroup
\[ \langle a \rangle = \{ a, a^2, \ldots, a^u \} \]
subgroup generated by \( a \).

**Lagrange Thm:** A finite group \( G \)
\[ \forall a \in G \quad a^{|G|} = 1 \]

**Def:** order \( (a) = |\langle a \rangle| \) = least \( u \geq 1 \) s.t. \( a^u = 1 \).

**Corollary:** \( \forall a \in G \quad \text{order}(a) \mid |G| \)

(If \( \text{order}(a) = u \) & \( |G| = au + \beta \quad \beta \in \mathbb{Z}_1 \sim u-1 \))

then \( 1 = a^{|G|} = a^{au + \beta} = a^\beta \) - contradiction.)
Def: If \( \langle a \rangle = G \) then \( a \) is a generator of \( G \).

Def: A finite group is cyclic if it has a generator as \( G \).

Thm: \( \mathbb{Z}_n^* \) is cyclic iff \( n \) is 2, 4, \( p^m \) or \( 2p^m \).

When we use \( \mathbb{Z}_p^* \) we often use it together with a generator \( g \) so that

\[ f_g: x \mapsto g^x \]

is a bijection from \( \mathbb{Z}_{p-1} \) to \( \mathbb{Z}_p^* \).

\[ g^x \mapsto x \]

discrete log, believed to be hard.

In \( \mathbb{Z}_p^* \), the fastest alg for computing discrete log takes time \( \geq 2^{\log p + \varepsilon} \), sub-exp alg.

How do we efficiently find a generator?

Note: A random element in \( \mathbb{Z}_p^* \) is not a generator w.p. at least \( \frac{1}{2} \). (if it is in \( \mathbb{Q}_p \) it is not a generator.)
- It is easy to find generator in a prime order group! Every element except the identity is a generator.

- In \( \mathbb{Z}_p^* \) we need to know the factorization of \( p-1 \) to find a generator.

**Diffie-Hellman Key Exchange**

(precursor to public-key crypto)

Allows Alice & Bob to share a secret key in the presence of a passive eavesdropper.

Let \( G \) a cyclic group w. generator \( g \)

(i.e. \( G = \{ g, g^2, \ldots, g^{1613} \} \))

\( G, g \) fixed & public

\[ A \xrightarrow{\text{Choose random}} g^x \xrightarrow{\text{A}} B \]

\[ \text{Choose random} \ y \in \mathbb{Z}_{1613} \]

\[ g^y \xleftarrow{\text{B}} \]

\[ \text{Shared Secret} : \ K = g^{x+y} \]
Computation Diffie-Hellman Assumption (CDH)

Given $g^x, g^y$, it is hard to compute $g^{x+y}$
(i.e., there is only negl probability of succeeding).

CDH => Eve doesn't learn k except w. negl prob.

This guarantee is not strong enough to then use
k as a secret key, since Eve may learn \( \frac{1}{2} \)
the bits of k.

Decisional Diffie-Hellman Assumption (DDH)

Giving $g^x, g^y$, it is hard to distinguish $g^{xy}$
from $g^u$, where $u$ is random in $\mathbb{Z}_{1+\text{length}}$.

Thm: DDH => DH key exchange is secure, i.e.,
Eve cannot dist. between k and a fresh random
key.

(Follows immediate from DDH assumption)
DDH does not hold in $\mathbb{Z}_p^*$ (H.W.)

We believe DDH holds in a prime order subgroup of $\mathbb{Z}_p^*$. (e.g. $\mathbb{F}_p$ for $p=2g+1$ safe prime.)

Public-Key Encryption

Consists of 3 PPT algorithms: KeyGen, Enc, Dec.

**KeyGen**: Takes as input security parameter $1^\lambda$ (in unary, so that KeyGen will run in poly time).

$\lambda \approx$ key-size. It outputs $(PK, SK)$.

**Enc**: Takes as input $(PK, m)$, outputs a ciphertext $C$ msg in msg space $M$.

**Dec**: Takes as input $(SK, C)$ and outputs $m$ ciphertext.

**Correctness**: $\forall (PK, SK) \leftarrow \text{KeyGen}(1^\lambda), \forall m \in M$

$$\Pr\left[ \text{Dec}(SK, \text{Enc}(PK, m)) = m \right] = 1$$

**Semantic Security (CPA-security)**: $\forall m_0, m_1 \in M, \text{Im}(m_0) \neq \text{Im}(m_1)$

$$(PK, \text{Enc}(PK, m_0)) \not\approx (PK, \text{Enc}(PK, m_1))$$
- More generally, $m_0, m_1$ can be adv chosen after seeing $pk$.

Note: We do not give the adv oracle access to $Enc(pk, \cdot)$ since it can be computed from $pk$.

**El-Gamal Enc. Scheme**

Let $G$ be a cyclic group w. generator $g$ st.
we believe DDH holds: $(g^x, g^y, g^{xy}) \approx (g^x, g^y, g^u)$

**Key Gen:** Choose $x \leftarrow [1, \ldots, lG13]$  
Let $sk = x$, $pk = g^x$

(Formally, choosing $G$ & $g$ should also be)  
part of Key Gen: Choose safe prime $p$  
$G = Q_p$, $g$ = any generator (any $g \in Q_p \setminus \{1\}$).

$Enc(pk,m)$: Choose random $y \leftarrow [1, \ldots, lG13]$  
$G$  
Output $(gy, g^{xy \cdot m})$  
DH key.
\[ \text{Dec}(x, (g^y, g^{xy}\cdot m)) \]
\[ \frac{\text{output}}{(a, b)} \]
\[ \frac{b}{a^x} \]

Semantic Security follows immediately from DDH:

\[ (g^x, g^y, g^{xy}\cdot m_0) \approx (g^x, g^y, g^{y'_x}) \approx (g^x, g^y, g^{xy}\cdot m_1) \]