Lecture 11

- El Gamal
- DDH is easy, not IND-CPA for \( cx \cdot (z^p_x) \).

But what we showed is DDH \( \Rightarrow \) El Gamal IND-CPA

if we take \((z^p_x)\) in a sub group of \((\mathbb{Z}_p^*)\).

\[ \mathbb{Z}_p^* \cap \mathbb{Q}_p = \mathbb{Q}_p \]

If \( x = y^2 \text{ mod } p \), if we take up elements which are squares, then DDH becomes hard.

We also showed El Gamal \( \not\sim \) IND-CPA primary reason is homomorphic, i.e. malleable.

\[ \text{private key } \]
\[ r_c z_q \]
\[ x \in \mathbb{Z}_q \]
\[ x \in \mathbb{Z}_q \]

\[ K = g^x \text{ mod } p. \quad g = g^x \text{ mod } p. \]

\[ c = my^x \text{ mod } p. \]

\[ (k/c) \]
Problem with IND-CCA is because there is no randomization. We can construct another msg

\[(k, r),\]

\[g, \text{msg}\]

\[g^r, g^r, \text{msg}, y^r, y^s\]

\[m^r\]

\[\ \text{Challenger}\]

\[g^r, \text{msg}, y^r\]

\[b.\]

\[\ \text{IND-CCA Dec Cyan}\]

\[\text{Cramer-Shoup Encryption}\]

In Cramer we use some kind of hash \(f\) on the msg so when we try to
RSA can also be mangled.

Textbook RSA is not IND-CPA because its deterministic:

$$c = \text{Enc}_{K}(m_0)$$

Since deterministic scheme then $m_0 = c_0$.

Any deterministic scheme is not IND-CPA.

Also textbook RSA not IND-CPA.

$$c = (m^e \mod N)$$

So we have $c$.

Now multiply $c$ by $(r^e \mod N)$.

$$c' = r^e \mod N \cdot m^e \mod N$$

We get $(rm)$.

But we picked $r$ so we get back $m$. 
some game & as prev. IND-CCA.

How to use RSA scheme and build a secure RSA-OAEP. → Next Class.

Signature.

Authentication of the Public Key (Trusted Certification).

Alice \( \rightarrow \) Bob

\[ \text{Enc } \text{PK}_b (m) \]

Verisign

\( \text{Bob} \)

\( \text{PK}_b \)

\( \text{Sig}_{\text{PK}_b} (\text{Bob}, \text{PK}_b) \)

Bob

Certification

Challenge, give me your cert.

Alice \( \rightarrow \) Bob

\[ \text{Cert} \text{CEBZH, Sig } \text{SK}_B (\text{Challenge}) \]

\[ \text{Enc } \text{PK}_B (m) \]
Certification Authority is trusted by Alice
someone like Verisign.

Digital Signatures

It achieves similar properties
as achieved by MAC.

Digital signatures are counterpart for MAC in PKI.

It achieves:

→ Authentication.

→ Integrity.

If I tamper with \((m, s)\) pair

I should come up with \((m', s')\) pair.

→ It achieves Non Repudiation.

\[ \text{sig} (m) \]

I have signed a msg and I
give it out to someone.

Since only I know my secret key
then I cannot deny it.

in (MAC) we don't achieve it because both parties know (K).

So, if Bob goes to Court saying Alice signed something, then Court cannot say for sure that Alice signed the check because both Alice and Bob share the key K.

Definition of Digital Signature Schemes:

(keygen, sign, verify)

keygen: create secret key, & corres. pk.
in any setting (DL, RSA etc).

sign: on input of m, s = sign sk(m).

output = (m, s).

verify: input = (m, s) → output = true if correct sign of (m).

if (PK (SK (m)) = m)

output (1);

else output (0).
Security Notion for a Digital Sig. Scheme

For MAC, we had a CMA security notion.

For MAC - Existential Forgery under adaptively chosen msg attacks.

For signatures, we have a similar notion.

For MAC:

\[ \text{Sign}(\text{oracle}) \text{ (had a key)} \]

Adversary tries to come up with

\[ (m, \text{MAC}_k(m)) \]

\[ (m \neq m_i) \]

Adversary

\[ m, \text{MAC}_k(m) \]

Verification Oracle
signatures

\[ \text{Signature} \]

\[ \text{Oracle SK}_{i} \]

\[ \sigma \xrightarrow{\text{sig}} \]

\[ A \xrightarrow{(m, s)} \]

\[ \text{if } \text{Verify}_{PK}(m, s) = 1 \]

\[ \text{then (Adversary) wins} \]

Here there is no Verification Oracle as Public key is known by all.

\[ \text{Constructions} \]

\[ \text{DL} \quad \text{RSA} \]
Schnorr Signatures.

Key Gen:

We can be working in both groups $\mathbb{Z}_p^*$ or subgroup $\mathbb{Z}_p^* \cong \{g, g', \ldots, g^q\}$.

\[ \text{Key Gen: } \quad x \in \mathbb{Z}_q. \]

\[ g = g^x \mod p. \]

\[ \text{PK: } \]

\[ \text{sign: } \rightarrow \text{ Pick Random } k \text{ in } \mathbb{Z}_q. \]

\[ \rightarrow \text{ Compute } r = g^k \mod p. \]

\[ \rightarrow \text{ Concatenate } \]

\[ \rightarrow \quad c = H(m || r) \]

\[ \rightarrow \quad s = (k + cx) \mod q. \]

Send out $(m, c, s)$. 

Verify:

\[ s = (k + cx) \mod q \]

\[ g^s = g^k \cdot g^{cx} \mod g_p \]

\[ g^s = r \cdot y^c \mod p. \]

We are given \((m, c, s)\).

So compute \(r' = g^s \cdot y^c \mod p\).

Then check if \(c^2 = H(m || r') \mod (p-1)\).

else 0.

If in \(\mathbb{Z}_p^*\) group.

\[ \alpha \in \mathbb{Z}_{(p-1)} \]

\[ g^\alpha = g^x \mod p \]

pk.

sign \((m)\)

Pick \(k \in \mathbb{Z}_{p-1}\)

\[ r = g^k \mod p. \]

\[ c = \alpha \cdot H(m || r) \]

\[ s = (k + cx) \mod (p-1) \]

output \((m, c, s)\).
Verify: \[ r' = g^s \cdot g^{-c} \mod p. \]

Check if \( c \overset{?}{=} H(\|m\| ||r') \).

Security for Schnorr
we will look in group \( \mathbb{Z}_q \).

Theorem: If DL is hard then
Schnorr is CMA Secure, in
Random Oracle Model [ROM].

Proof: Contradiction.

\[ \Rightarrow \] Schnorr \Rightarrow DL.

CMA Sec

If an attacker can break Schnorr
CMA Security, then we
can construct another attacker
\( B \) which breaks DL problem.
A is a CMA attacker so it will interact with signing oracle.

B doesn't have the secret key but has to somehow simulate the private key signatures.

\[ K \in \mathbb{Z}_q. \]
\[ r = g^K \mod p. \]
\[ s = (K + cX) \mod q. \]

So what we do is pick any \((s, c)\) randomly and we calculate
\[ r = g^s y^{-c} \mod p. \]
B ran A once, A output forgery as \((m, c, s)\). 

If A is a good forgery:

B now runs A again.

A now A outputs a forgery on same message \(m\) \((m, c', s')\).

So given this happened, we can simply use them to compute DLP.

The value used in both is same.

We got \((m, c, s)\) so \(x = g^{y-c} \mod p\).

\((m, c', s')\) \(x = g^{s'} \cdot y^{c'} \mod p\) with same.

\(g^{y-c} \mod p = g^{s'} \cdot y^{c'} \mod p\).

\(y = g^x\)

\(g^{s-yc} = g^{s'} \cdot g^{yc'} \mod p\).

\(s-yc = c'-yc \mod q\).

\(x = s-s' (c-c')^{-1} \mod q\).
Getting \((m, e, s')\) is done by the Forking Lemma.

First run of algo.

\[
\begin{align*}
& s_1 & \leftarrow & q_1 \\
& s_2 & \leftarrow & q_2 \\
& s_3 & \leftarrow & q_3 \\
& s_4 & \leftarrow & q_4
\end{align*}
\]

This means that one of the queries corresponds to \((m, r)\).

Now, next time you run an adversary, the responses will be the same as previous instance.

\[
\begin{align*}
& s_1 & \leftarrow & q_1 \\
& s_2 & \leftarrow & q_2
\end{align*}
\]

So \(e^2\)

\[
q / h
\]