L10.1

3/11/19

Today: Pedersen Commitment

PK Encryption

El-Gamal Enc

Recall: Previous lecture we covered DH Key Exchange.

G is a cyclic group, g generator

$$\begin{array}{ccc} A & & B \\ x \leftarrow \{0,1,..,|G|-1\} & \xrightarrow{g^x} & y \leftarrow \{0,1,..,|G|-1\} \\ & \xleftarrow{g^y} & \end{array}$$

$$Key = g^{xy}$$

The key is ind. from random $\underbrace{\text{in } G}_{\text{for an eavesdropper who sees } g^x, g^y}$ if:

$$(g^x, g^y, g^{xy}) \cong (g^x, g^y, g^r)$$

DDH Assumption

For which groups is it reasonable to assume the DDH Assumption?

Eg: $\mathbb{Z}_p^*$ is a cyclic group (assuming p is a prime)

Moreover, we assume Discete Log is hard in $\mathbb{Z}_p^*$

i.e. $g^x \xrightarrow{\text{HARD}} x$ for $x \leftarrow \{0,1,..,p-1\}$ & g fixed generator.

<u>However</u>: DDH Assumption does <u>not</u> hold in $\mathbb{Z}_p^*$

<u>Example of cyclic group that we believe DDH holds</u>:

Let p, q large prime numbers s.t. $q / p-1$ (eg. $p = 2q+1$ ← safe prime)

Let g be a generator of a subgroup of $\mathbb{Z}_p^*$ of order q.

$$\langle g \rangle = \{1, g, g^2, .., g^{q-1}\}.$$

(eg. $p = 2q+1$ g any quadratic residue s.t. $g \neq 1$)

We believe DDH holds in $\langle g \rangle$

Both constructions that we will see today:

(Pedersen Commitment & El-Gamal Enc,

use a cyclic gp of <u>prime order</u>, which is a subgroup of $\mathbb{Z}_p^*$.

For El-Gamal Enc we will rely on DDH

For Pedersen Com we will rely on DL ← will use the fact that $GF[|\langle g \rangle|]$ is field

Pedersen Commitment Scheme

Recall: A commitment scheme consists of to phases:

Commit(x) $\rightarrow$ Alg that given x produces a "commitment" to x

Reveal (com) $\rightarrow$ "Opens" the commitment com, i.e, reveals x.

Properties:

Hiding: Commit(x) reveals nothing about x

Binding: Can only open com in one way (i.e, cannot change x).

sometimes we want non-malleability:

Given Commit(x) cannot generate a commitment to a related value, say $x+1$.

Pedersen Commitment

Setup: p, q large primes s.t. $q \mid p-1$

g a generator of order q subgroup of $\mathbb{Z}_p^*$,

i.e. $g \in \mathbb{Z}_p^*$ s.t. $|\langle g \rangle| = q$.

choose at random $a \leftarrow \{1, 2, \ldots q-1\}$, and let $h = g^a$.

Note that h also generates $\langle g \rangle$.

$\text{Commit}_{g,h}(x)$: For $x \in \mathbb{Z}_q$ $(x \in \{0,1,\ldots q-1\})$:

Choose at random $r \leftarrow \mathbb{Z}_q$

Output $com = g^x h^r \bmod p$.

Reveal (com) : reveal x, r

Receiver verifies that $com = g^x h^r \bmod p$

Hiding: Perfect hiding : Hiding holds even against an unbounded adversary.

For any $x \in \mathbb{Z}_q$ $g^x \cdot h^r$ is uniformly distributed in $\langle g \rangle$ independent of x

In other words $\forall\, c \in \langle g \rangle$ $\forall x \in \mathbb{Z}_q$ there exists unique $r \in \mathbb{Z}_q$ s.t. $g^x h^r = c = g^z$ for some $z \in \mathbb{Z}_q$

$g^x h^r = g^{x+ar}$

$\Rightarrow x + a \cdot r = z$

$\Rightarrow r = a^{-1}(z - x) \bmod q$.

Note that a has an inverse mod q since $GF(q)$ is a field (since q is prime!)

Binding: Computationally binding

An all powerful adv. may open in two different ways, but a bounded adv (who cannot break DL) cannot.

Suppose adv. generates $c \in \langle g \rangle$ together w. $x \neq x'$ & r, r' s.t.

$$c = g^x h^r = g^{x'} h^{r'} \mod p$$

$$\Longrightarrow \quad x + ar = x' + ar' \mod q$$

$$\Longrightarrow \quad a = \frac{x - x'}{r - r'} \mod q$$

Note that $r - r' \neq 0$ (since o.w. it must be that $x = x'$). Hence $r - r'$ has an inverse mod q since $GF(q)$ is a field (for prime q)

Hence, such an adv broke DL in $\langle g \rangle$.

Given $h = g^a$ he found a.

Non-malleable : No

Given $c = \text{commit}(x; r) = g^x h^r$

randomness

one can easily compute

$$c \cdot g = g^{x+1} h^r = \text{commit}(x+1; r).$$

Not all applications need non-malleability.

Public key Encryption

Consists of three algorithms: KeyGen, Enc, Dec.

KeyGen: Takes as input a security parameter λ in unary (1^λ). $\lambda \approx$ key size

The reason for giving it in unary is only syntactic: so we can say that KeyGen runs in poly time (in input length).

KeyGen(1^λ) outputs a pair (PK, SK).

Enc: Takes as input (PK, m) and outputs ciphertext c.

msg in msg space $\mathcal{M}$

Dec: Takes as input (SK, c) and outputs m.
ciphertext

Correctness

$\forall (PK, SK)$ generated according to KeyGen, $\forall m \in \mathcal{M}$

$$Dec(SK, Enc(PK, m)) = m$$

(often correctness is relaxed to hold with high prob over $(PK, SK) \leftarrow KeyGen(1^\lambda)$)

"Semantic Security": $\forall m, m' \in \mathcal{M}$

$$(PK, Enc(PK, m)) \cong (PK, Enc(PK, m'))$$

In the formal def m & m' can be generated adv. as a function of PK.
(will be defined formally later)

Note: For semantic security to hold

KeyGen must be randomized &

Enc must be randomized.

Usually Dec is deterministic.

El-Gamal Enc Scheme (1984)

Let $G = \langle g \rangle$ be a cyclic gp w. generator g

for which we believe DDH holds

$$(g^x, g^y, g^{xy}) \cong (g^x, g^y, g^r)$$ for x, y, r random in $\{0, 1, .., |G|-1\}$.

An example of such group

the set of all quadratic residues in $\mathbb{Z}_p^*$, for p which is a safe prime $(p = 2q+1)$.

<u>KeyGen</u>: Choose at random $x \leftarrow \{0, 1, .., |G|-1\}$

Let $SK = x$

$PK = g^x$

Output (PK, SK)

<u>Enc</u>: Given $PK = g^x$ & msg $m \in G$:

↑ randomized

Choose at random $y \leftarrow \{0, 1, .. |G|-1\}$

Compute $K = g^{xy}$

Output $(g^y, K \cdot m)$

[Note: K can be thought of a key obtained from a DH key Exchange, w. g^x & g^y]

<u>Dec</u>: Given $SK = x$ & ciphertext

(a, b) $(= (g^y, g^{xy} \cdot m))$

output $m = b / a^x$

Correctness

Follows since

$$b/a^x = g^{xy}\cdot m / a^x = g^{xy}\cdot m / g^{yx} = m \checkmark$$

Security: To prove security we need to prove that given $PK = g^x$ & given g^y and m, $K = g^{xy}$ is uniformly distributed in G, and hence is a "good mask" for m.

This follows from the security of the DH Key Exchange, which is known to be secure under DDH.

$$\xleftarrow{\quad PK = g^x \quad}$$

$$\xrightarrow{\quad g^y \quad}$$

$$m \cdot \underbrace{g^{xy}}_{\text{key}}$$

Encrypt by multiplying by key
Decrypt by dividing by key