Today: Public key encryption

Recall: Diffie-Hellman Key Exchange:

Let G a finite cyclic group ($G=\mathbb{Z}_p^*$) of order n (i.e., $|G|=n$).

$\mathbb{Z}_p^* = \{1,\ldots,p-1\}$ with mult. mod p

Let g be a generator of G: $G=\{g, g^2,\ldots,g^n\}$

$$
\begin{array}{ccc}
A & & B \\
\text{Choose at random } a \text{ in } \{1,\ldots,n\}, & \xrightarrow{A=g^a} & \text{Choose at random } b \text{ in } \{1,\ldots,n\}, \\
 & \xleftarrow{B=g^b} &
\end{array}
$$

$$K = g^{ab} = A^b = B^a$$

Computational Diffie-Hellman (CDH) Assumption:

Given g^a, g^b, it is hard to compute g^{ab}, except with negl probability.

A passive adv cannot guess K assuming CDH!

This naturally lends itself to public key encryption!

Definition:

A public key encryption scheme consists of three efficient (randomized) algorithms: Gen, Enc, Dec, with the following syntax:

1. Gen takes as input security parameter and outputs a pair of secret and public keys (sk,pk).
2. Enc takes as input a public key pk and a msg m (from the msg space) and outputs a ciphertext ct.
3. Dec takes as input a secret key sk and a ciphertext ct and outputs a message m (from the message space) or abort.

Correctness:

For every (sk,pk) generated according to Gen, and for every msg m (from the msg space),

$$\Pr[\mathrm{Dec}(sk, \mathrm{Enc}(pk,m))=m]=1.$$

Note:

A public key encryption scheme is a digital analog of a locked box, where only the receiver has the key.

Applications of public key encryption:

1. Key-exchange:

Server sends a public key pk to browser.

Browser chooses random K and sends Enc(pk,K) to server.

Now the server share a symmetric key and use that for communication!

2. Secure email:

A user A want to encrypt an email to another user B.

If A has pk, then she can use it to send encrypted emails to B.

Security:

As in the symmetric key setting, we consider two flavors of security:

CPA (Chosen Plaintext Attack) security and

CCA (Chosen Ciphertext Attack) security.

CPA Security (a.k.a semantic security):

For every m and m' (from the msg space),

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

for a randomly chosen pk chosen according to Gen.

Note:

This definition is much simpler than CPA definition in the symmetric setting!

The reason is that in the public-key setting, the adversary can encrypt msgs on his own using pk!

CCA security:

Any efficient adv. wins in the following game only with prob.

$1/2$ + negligible:

Challanger | Adv

Generate (pk, sk) by running Gen $\xrightarrow{pk}$

$\xleftarrow{ct}$

$\xrightarrow{Dec(sk,ct)}$

$\xleftarrow{m_0,\, m_1}$

Choose a random bit b, let $ct_b = Enc(pk, m_b)$ $\xrightarrow{ct^* = Dec(sk, ct)}$

$\xleftarrow{ct}$

only if $ct \neq ct^*$ $\longrightarrow$ $\xrightarrow{Dec(sk,ct)}$

$\xleftarrow{b'}$

Adv wins if $b = b'$

El-Gamal Encryption scheme:

Let G be a finite cyclic group ($G = \mathbb{Z}_p^*$) of order n (i.e., $|G| = n$).

Let g be a generator: $G = \{g, g^2, \dots, \underset{\substack{\shortparallel \\ 1}}{g^n}\}$.

Let $H: G \longrightarrow \{0,1\}^*$ be a hash function (modelled as a random oracle).

Let (E,D) be a symmetric authenticated encryption scheme.

Gen:

Choose at random a in $\{1, \dots, n\}$, set $sk = a$ and $pk = g^a$.

Enc(pk,m):

Choose at random b in $\{1, \dots, n\}$. Let $K = H(pk^b)$.

Output $(g^b, E(K,m))$.

Dec(sk, (u,v)):

Compute $K = H(u^{sk})$ and output $m = D(K,v)$.

Correctness: For any pair $(pk, sk) = (g^a, a)$ and every msg m:

$$\mathrm{Dec}(a, (g^b, E(H(g^{ab}), m)) = D(H(g^{ba}), E(H(g^{ab}), m)) = m \qquad \checkmark$$

Performance:

To encrypt: 2 exponentiations: g^b, pk^b.

To decrypt: 1 exponentiation: u^{sk}

Exponentiation is slow! (A few miliseconds on modern processors.)

At first it seems like decryption is twice as fast.

But g^b can be computed efficiently by precomputing $\{g^{2^i}\}_{i=1}^{\log n}$

If we encrypt often to the same pk, then computing pk^b

can be done efficiently as well (with the same precomputation).

Security:

Semantic Security:

For semantic security, all we need to argue is that given $pk=g^a$,

and given the first part of the ct, g^b,

the symmetric key $H(g^{ab})$ is ind. from random:

$$(g^a, g^b, H(g^{ab})) \cong (g^a, g^b, U)$$

This assumption is called Hash Diffie-Hellman (HDH).

It is stronger than the Computational Diffie-Hellman Assumption.

But is equivalent to it in the ROM (Random Oracle Model).

Note: For semantic security it is sufficient to take (E,D) to be one-time secure, which was the proposal in the original El-Gamal scheme

CCA security?

In the CCA game the adversary gets additional infromation: Decryption oracle.

Adv can send (g^b, c) to the challenger who replies with $m = D(H(g^{ab}), c)$.

Suppose the underlying (E,D) is an authenticated encryption.

Intuitively, this seems to imply that the resulting El-Gamal scheme is CCA secure:

the decryption oracle is useless, since it decrypts (g^b, c) only if adv knows g^{ab}.

This is the case, since o.w., the key $K = H(g^{ab})$ is random and the fact that (E,D)

is an authenticated encryption implies that the adv cannot produce a valid ct

corresponding to the key K.

But if the adv knows g^{ab}, then the decryption oracle is useless!

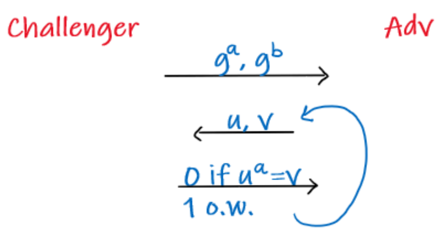
Nevertheless, we can't prove that El-Gamal is CCA secure under CDH

(in the random oracle model).

The reason is that the adversary may "not know if he knows g^{ab}" and the decryption oracle

will give the adv this information.

We can prove that it is CCA secure under the interactive DH assumption:



Adv cannot learn g^{ab} except with negl prob.

Note:

There are variants of El-Gamal that are CCA secure under CDH,

and also ones that do not rely on ROM! (Go to 6.857 and 6.875 for details!)