Today: Public key encryption

Recall: Diffie-Hellman Key Exchange:

Let G a finite cyclic group of order n (i.e., $|G|=n$).

Cyclic means that it has a generator g

s.t. $G=\{g^1, g^2, \dots, g^n\}$

Eg. $G=\mathbb{Z}_p^*$ which is $\{1,\dots,p-1\}$ with mult. mod p

in which case $|G|=n=p-1$.

Let g be a generator of G: $G=\{g^1, g^2, \dots, g^n\}$

A		B
Choose at random a in $\{1,\dots,n\}$,	$\xrightarrow{A=g^a}$	Choose at random b in $\{1,\dots,n\}$,
	$\xleftarrow{B=g^b}$

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

How do we choose a generator from $\mathbb{Z}_p^*$?

The order of an element x in G is smallest t s.t. $x^t = 1$

Theorem:

The order of each element divides the order of the group.

For $\mathbb{Z}_p^*$: the order of each element g divides $p-1$.

Choose p to be a safe prime: $p-1=2q$, where q is a prime.

Thus, each element g in $\mathbb{Z}_p^*$ is of order 1, 2, q, or $2q$.

There are only 2 elements of order 1,2: 1 and $p-1$

(since degree 2 polynomial $f(x)=x^2$ has at most 2 roots).

The remaining $p-3$ elements are of order q or $2q=p-1$,

half of the remaining are of order q and half are of order $2q$:

Consider the function $f: \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$ where $f(x)=x^2 \bmod p$.

The image of this function is of size $(p-1)/2$,

since each element x in the image has exactly two roots

x and $p-x$.

The image is the set of all quadratic residues (by def),

and each element in the image is of order 1 or q.

There is only one element of order 1 and hence $(p-3)/2$ of order q.

Thus, there are $(p-3)/2$ of the elements that are not of the form x^2

and all these are generators (i.e. of order $p-1$).

To choose a generator of Z_p^* (where $p=2q+1$ is a safe prime)

choose a random g, and check that $g^q \neq 1$ and that $g^2 \neq 1$.

If this is not the case try again.

Discrete Log Assumption:

Given a group G with generator g, it holds that given g

for a random x in $\{1,\dots,n\}$ where $n=|G|$.

it is hard to find x.

Namely, the function $f(x)=g^x$ is a one way function.

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:

Challenger		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 =/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: $G = \{g^1, g^2, \dots, \underset{\overset{\shortparallel}{1}}{g^n}\}$ both determined in a preprocessing phage

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

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 $(g^b, K \oplus m)$.

Dec(sk, (u,v)):

Compute $K = H(u^{sk})$ and output $m = K \oplus v$

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

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

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).

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).

CCA security?

No! Given Enc(pk,m) it is easy to generate Enc(pk, $m \oplus m'$)

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

Note:

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

(Go to 6.875 for details!)