Introduction to Public key cryptography

So far, everything we did required a shared secret between the participants. This has the great disadvantage that we need to be able to securely exchange secrets between participants that are potentially far apart. The objective of these two lectures is too introduce public key cryptography via three very important instances:

- public key encryption.
- key exchange.
- signature.

1 Public key encryption (RSA)

The first instance of a public key encryption scheme is also the most widely used nowadays. Here is the setting:

- PK: Public key: $N = pg$ for two secret primes p, q and $e \in \mathbb{Z}$ that is invertible mod $\phi(N) = (p-1)(q-1)$.
- SK: Secret Key: $d = e^{-1} \mod (p-1)(q-1)$.

The encryption goes like that:

$$\begin{aligned} \text{Enc}(m, Pk) &= m^e \mod N = c \\ \text{Dec}(c, Sk) &= c^d \mod N \end{aligned}$$

The validity of the scheme is ensured by the following proposition:

Proposition 1 (Validity of the decryption)**.** $(m^e)^d = m \mod N$.

Proof. We first need to calculate the cardinality of the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^*$. Let us look at the morphism

$$\begin{array}{ccc} \mathbb{Z}/N\mathbb{Z} & \xrightarrow{f} & \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z} \\ x \mod N & \longrightarrow & (x \mod p, x \mod q) \end{array},$$

The morphism is injective because if $x \mod p = 0$ and $x \mod q = 0$ (i.e., $f(x) = 0$), then $p|x$ and $q|x$ so $pq = N|x$ and $x \mod N = 0$ (i.e., $Kerf = \{0\}$). Moreover $|\mathbb{Z}/N\mathbb{Z}| = N = pq = |\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}|$ so the morphism is actually a bijection. $\mathbb{Z}/N\mathbb{Z} \simeq \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$. Since multiplication goes coefficient-wise, $(\mathbb{Z}/N\mathbb{Z})^* \simeq (\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/q\mathbb{Z})^*$. The only element of $\mathbb{Z}/pZ$ that does not have a multiplicative inverse is 0,

so $|\mathbb{Z}/p\mathbb{Z}^*| = p-1$. Therefore $|\mathbb{Z}/N\mathbb{Z}^*| = |\mathbb{Z}/p\mathbb{Z}^*| \times |\mathbb{Z}/q\mathbb{Z}^*| = (p-1)(q-1)$. This means in particular that $m^{(p-1)(q-1)} = 1 \mod N$. Thus,

$$\begin{aligned}(m^e)^d \bmod N &= m^e d \mod N \\ &= m^{1+k((p-1)(q-1)} \bmod N \qquad (\textit{since } \ d = e^{-1} \mod (p-1)(q-1)) \\ &= m(m^{(p-1)(q-1)})^k \bmod N \\ &= m \mod N\end{aligned}$$

□

Example 1. $p = 61,\ q = 53,\ N = 3233,\ e = 17,\ \phi(N) = 3120,\ d = e^{-1} = 2753 \mod \phi(N)$

Encryption of $m = 65$*;* $c = 65^{17} \mod 3233 = 2790$.

Decryption of c*:* $c^d = 2790^{2753} \mod 3233 = 65 = m$.

Remark. *If one can factor* N*, then one can recover* $(p-1)(q-1)$*, and thus compute* $d = e^{-1} \mod (p-1)(q-1)$*. It is unclear if it is a necessary condition.*

2 Diffie-Hellman key exchange protocol

The typical scenario in encryption is to exchange a key (via a potentially insecure channel), and then t use a stream cipher for encrypting/decrypting data. The Diffe-Hellman protocol was the first instance of public key cryptography. The goal is for Alice and Bob to create a shared secret while being listened to by Eve.

Public paramaters: A group G and $g \in G$

Alice	*Bob*
Computes $a \in \mathbb{Z}$	*Computes* $b \in \mathbb{Z}$
sends $g^a \longrightarrow$	
$\longleftarrow$ *sends* g^b	
computes $(g^b)^a = g^{ab}$	*computes* $(g^a)^b = g^{ab}$

Eve

Remark. *We rely on the hardness of finding* a *from* g^a *and* g*. It is the discrete logarithm problem. It seems to be a hard problem in certain groups such as* $(\mathbb{Z}/p\mathbb{Z})^*$ *or the group of points of an elliptic curve.*

Remark. *The previous remark does not mean that the DH is as hard to break as solving the DLP. It is sufficient, but we don't know if it is necessary.*

Example 2. $G = (\mathbb{Z}/101\mathbb{Z})^*$ $g = 2$

Alice	*Bob*
draws 43	*draws* 21
sends $2^{43} \mod 101 = 86 \longrightarrow$	$\longleftarrow$ *sends* $2^{21} \mod 101 = 89$
Computes $89^{43} \mod 101 = 8$	*Computes* $86^{21} \mod 101 = 8$

Shared Secret : 8

3 RSA Signature

The signature schemes we have seen so far require both the signer and the verifier to share a key. If we use the RSA encryption procedure backward, we can provide a signature scheme where the signer has a secret key and everyone can verify using a public key.

Public key: N, e coprime to $(p-1)(q-1)$

Private key: $d = e^{-1} \mod (p-1)(q-1)$

Signature: $S(m, Sk) = m^d \mod N = t$

Verification: $V(m, Pk, t) =$'true' if $t^e = m \mod N$, 'false' otherwise.

Example 3. $p = 61,\ q = 53,\ N = 3233,\ \phi(N) = 3120,\ e = 17,\ d = e^{-1} \mod \phi(N) = 2753$

To sign $m = 65$ *we do* $m^d = 65^{2753} \mod 3233 = 1393 = s$.

To verify s, *we do* $s^e = 1393^{17} = 63$.

4 Semantic security of RSA

This is an informal section to point out the issues in terms of semantic security that are raised by public key encryption. Normally (in the one-time key scenario), the challenger draws the key $h \leftarrow k$ and outputs $Enc(m, h)$. Here, the key Pk is public and can be used by everyone. So no one can prevent the adversary to compute $Enc(m_0, Pk), Enc(m_1, Pk)$ and to compare them to the challenge.

Necessary conditions: $Enc(m_1, Pk)$ produced by the challenger has to be different from $Enc(m_1, Pk)$ produced by the adversary. So a given message m has to have many (ex: 2^{128}) different encryptions under the public key. With RSA, one way to do that is to use some random padding. Alice wants to encrypt m. She draws a random sequence $n \in \{0,1\}^{128}$ and constructs $m' = n \mid\mid m$. Then $Enc(m, Pk) = m'^e \mod N$. To decrypt, Bob first decrypts m' and then discards the first 128 bits of randomness.