RSA – Public Key Implementation

RSA is the implementation of the idea that Diffie and Hellman invented about the possibility to achieve confidentiality or authentication without the exchange of any key, i.e. the Public Key Infrastructure. In order to learn RSA, we need to use several mathematical properties.

Let *p* and *q* two **large** prime numbers. Denote their product $N=pq$.

Please note that N is not prime (it can be factorized in p and q), but it is a *semiprime*, i.e. a product of two primes.

Then let's consider the multiplicative group $Z_N^* = Z_{pq}^*$. The multiplicative group Z_n^* contains all the integers between [1,pq-1] that are coprime with both p and q. But its cardinality is not pq-1 (N is not prime). In fact, in Z_{pq}^* there are (p-1) multiples of q and (q-1) multiples of p, and therefore its cardinality is:

$$\phi(pq) = |Z_{pq}^*| = pq-1-(p-1)-(q-1) = pq-(p+1)+1 = (p-1)(q-1)$$

Therefore, by the Euler's theorem, for every $x \in Z_{pq}^*$ we have that $x^{(p-1)(q-1)} = 1$.

We have noticed that not all the integers between [1,pq-1] are in Z_{pq}^*. This means that each element in [1,pq-1] does not necessarily have a unique inverse in Z_{pq}^*. But we want a number *e* between 1 and (p-1)(q-1) such that the power $x \rightarrow x^e$ is a one-to-one operation in Z_{pq}^* (i.e. there is only a multiplicative inverse *d* of *e* in Z_{pq}^* such that $x^{de}=1$). Since an element in [1,pq-1] can have more than one inverse in Z_{pq}^*, we cannot choose *e* at random in Z_{pq}^*, so the question is how to choose *e*.

The answer is: if *e* is relatively prime to (p-1)(q-1) then $x \rightarrow x^e$ is a one-to-one operation in Z_{pq}^*.

This is exactly what we need when we have to define a pair of asymmetric keys.

Therefore *e* has to be accurately choosen. From all this it comes the RSA protocol for deciding a pair of Public/Private Keys.

RSA protocol

- Choose two prime numbers *p* and *q* and let N = pq be their product
- Choose $1<e<\phi(N)$ such that $gcd(e, \phi(N))=1$ (it means that $1<e<(p-1)(q-1)$ and e is coprime with $\phi(N)=(p-1)(q-1)$ as we stated above)
- Let *d* be such that $de \equiv 1 \bmod \phi(N)$ (d is the multiplicative inverse of e mod (p-1)(q-1))
- Choose (e,N) as Public Key
- Choose (d,N) as Private key

When we have to encrypt $M \in Z_N$ we perform $C = E(M) = M^e \bmod N$

When we have to decrypt $C \in Z_N$ we perform $M = D(C) = C^d \bmod N$

Please note that when encrypting by means of $C = M^e \bmod N$, the decryption $D = C^d \bmod N = M^{ed} \bmod N$ only works if *e* and *d* satisfy the equation $de \equiv 1 \bmod \phi(N)$

Proof of correctness

We have that *p* and *q* are primes and $n=p*q$. We choose *e* in the range [1, Φ(n)], and *d* such that $ed \equiv 1 \bmod \Phi(n)$. Since $ed \equiv 1 \bmod \Phi(n)$, it means that $ed = 1 + h\,\Phi(n)$ for $h \geq 0$. Therefore, for a message $m \in [1,n-1]$ we have that $m^{ed} \bmod n = m^{1+h\,\Phi(n)} \bmod n = m m^{h\,\Phi(n)} \bmod n$.

We have to choose the plaintext *m* such that $m<n$. We candistinguish two cases:

- If $m \in Z_n^*$, for the Euler theorem, it holds that $m^{h\Phi(n)} \equiv 1 \bmod n$, and therefore the latter part of the decryption, the green one, disappears leaving only *m* and so the decryption works.
- If $m \notin Z_n^*$, then since $m<n$ it means that either *m* is a multiple of *p* or *m* is a multiple of *q*. So in the case it is a multiple of *p*, it follows that $m \equiv 0 \pmod p$, and $m^{ed} \equiv 0 \pmod p$, so both parts are congruent to 0. Same thing happens if *m* is a multiple of *q*. Finally, since $gcd(p,q)=1$, the previous equations hold for *n* too, so $m^{ed} \equiv m \pmod n$.

The strength of the algorithm relies on the fact that even knowing N and e (they compose the public key), the attacker is not able to compute d, because in order to compute d the attacker must know $\phi(N)$, but in order to get it he must know *p* and *q* but the attacker knows only N, and it is pretty hard to factorize it in order to retrieve *p* and *q*.

RSA uses a variable-length key, 512 bits is common. It is slower then DES and therefore RSA is not usually used to encrypt long messages, it is used instead for encrypt a secret key, which then will be used for confidentiality purposes.

A simple example is the following:

$p=3$, $q=11$ and so $N = p*q = 33$

$\phi(N) = (p-1)*(q-1) = 2*10 = 20$

I choose $e=7$ since $e < (p-1)*(q-1) = 20$, and e is coprime with 20. Then I take $d = 3$ since $ed = 1 \pmod{20}$ because $ed = 3*7 = 21 = 1 \pmod{20}$

So the public key is $(e,N) = (7,33)$ and the private key is $(d,N) = (3,33)$

If I want to encrypt a message, for example $m=15$, I perform $c = m^e \bmod N = 15^7 \bmod 33 = 27$

In order to decrypt I perform $c^d \bmod N = 27^3 \bmod 33 = 15$

RSA implementation

Find *p* and *q* randomly s.t. they are large. Then choose the exponent *e*. Then compute *d* with the Euclid's extended algorithm. Then, when encrypting a message M, perform the fast exponentiation algorithm in order to compute $M^e \bmod N$.

ATTACKS AGAINST RSA

Factorization

A first, trivial attack against RSA could be to try to factorize N in order to get *p* and *q* and from that $\phi(N)$ and then d. But if N is large enough, it is too hard to factorize. It is required that *p* and *q* have at least 100 digits and have not to be too close each other.

Weak Messages

There are weak messages that are not good messages to be encrypted with RSA protocol and they are:

- m=0 because 0 to the power of anything is 0 and so the ciphertext will be exactly equal to the plaintext
- m=1 for the same reason as above
- m=N-1 this is slightly harder to understand. If the messange is N-1 and the exponent is *e=3*, then it happens that:

$$(N-1)^3 \bmod N = (N-1)^2(N-1) \bmod N$$
$$(N-1)^2 \bmod N = 1 \bmod N$$
$$(N-1)^2(N-1) \bmod N = (N-1) \bmod N$$

and therefore the ciphertext is equal to the plaintext. Actually this works for every *odd e*, because if *e* is even then $(n-1)^e \bmod n = 1 \bmod n$, and therefore for every *odd e* it happens that:

$$(n-1)^e \bmod n = (n-1)^{e-1}(n-1) \bmod n = (n-1) \bmod n$$

Small *m* and small *e*

If both *m* and *e* are small it could happen that $m^e < N$ and therefore $m^e \bmod N = m^e$

Small *e* and similar messages

If only *e* is small and two messages are similar, suppose one is m and the other is m+1, then the two ciphers will be $c_1 = m^3 \bmod n$ and $c_2 = (m+1)^3 \bmod n$, but then it is easy to write a system of equation for which it cames out that:

$$\mathrm{m} = \frac{(c_2 + 2c_1 - 1)}{(c_2 - c_1 + 2)}$$

And the adversary will be able to get the two messages easly.

Actually this works for every pair m_1 and $m_2 = \alpha m_1 + \beta$

Chinese Reminder Attack

Another nice attack is when three users have same public key, let's say (e,n_1) , (e,n_2), (e,n_3) and another user send the same message to all of them. Adversary will know $m^3 \bmod n_1$, $m^3 \bmod n_2$ *and* $m^3 \bmod n_3$ so he can compute $m^3 \bmod n_1 n_2 n_3$ by means of Chinese Reminder Theorem. Moreover, m is less than n_1,n_2 and n_3 and so $m^3 < n_1 n_2 n_3$ and therefore $m^3 \bmod n_1 n_2 n_3 = m^3$.

Same N

Another attack is when two users chosed the same N. So they chosed same *p* and *q*. So one of them can try to discover *e* of the other user (he also has *d* because it is public). The solution is that every one has to choose a different *n* (probability to choose the same *n* is very low).

Multiplicative property of RSA

Another weak property is that if a message M is the product of two other messages M_1 and M_2, then by the invariance over multiplication we have that:

$$M^e \bmod N = (M_1 * M_2)^e \bmod N = ({M_1}^e \bmod N * {M_2}^e \bmod N) \bmod N$$

…

At the end of all this discussion, we got that RSA presents some weaknesses, and therefore a good approach that has to be followed when using RSA is the following:

- choose good numbers *p,q,e,d* such that they satisfy the properties discussed so far
- before encrypting message M, perform some preprocess on M in order to obtain M' (M' has not to change the meaning of M) and then perform the encryption of M'

For all this reasons, some standard have been invented in order to perform a good RSA usage. We are going to see some of them.

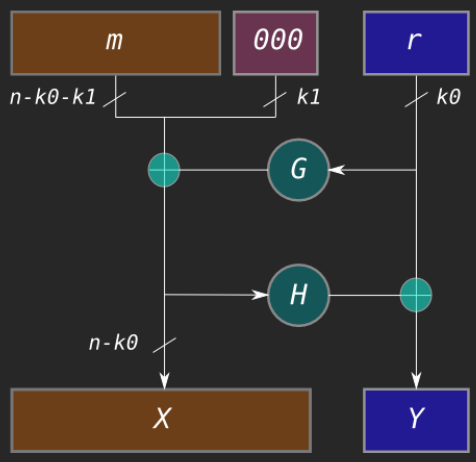
PKCS – Public Key Crypto System

It is a set of standards that specify how RSA should be used. We will be seeing only PKCS#1:

m = 0 || 2 || at least 8 non-zero bytes || 0 || M

where M is the original message. The first byte setted to 0 ensures that the message is less than N. The second byte equals to 2 denotes encryption of a message (if it is 1 it denotes digital signature). The random bytes imply that the same message sent to different people will result in different ciphertext. Actually this standard is quite obsolete.

OAEP – Optimal Asymmetric Encryption Padding



It is the version 2 of the previous standard. *m* is the original message (n-k0-k1 bits), k0 and k1 fixed by the protocol. *G* and *H* are hash functions defined by the protocol. Message *m* is padded with k1 zeroes, *r* is a random string of k0 bits. G expands *r* from k0 to *n-k0* bits, then the XOR is performed between It and the padded message, which results in *X*. Then the hashing function H reduces *x* to a *k0* bits string, that will be XORed with *r* in order to get Y. The output will be X||*Y* which will be then encrypted with RSA. The decryption will be a reverse procedure, and the outcome will be decoded in order to retrieve *m* in the following way: *r* = Y ⊕ H(X) and then m|000… = X ⊕ G(r), then discard the k1 zeroes from m|000… and obtain the message.

This scheme ensures what is called ALL-OR-NOTHING security, that means that in order to retrieve *m* we must both X and Y, because we need X in order to retrieve *r* from Y, and we need *r* in order to retrieve *m* from X. Moreover, one bit of error on the ciphertext will completely damage the message.

EL GAMAL Encyrption

It is a mode of encryption strongly based on Diffie-Hellman Key Exchange approach, indeed we will see that the underlying mathematic is the same. El Gamal is an alternative to RSA and so is able to generate a pair Public/Private Keys in order to perform *encryption* and *decryption*. The protocol is the following:

Given a prime *p* and a generator of the Z_p cyclic group of order *p*, Alice chooses *x* in the range [1,p-1] as the private key, and she chooses the number $g^x mod\ p$ as the public key. If Bob wants to send a message $m \in Z_p$ to Alice, he chooses a number *y* in the range [1,p-1] and he sends to Alice this pair of messages:

$(g^y mod\ p,\ m*g^{xy}\ mod\ p)$

He can send the second message because he knows *m,y* and $g^x\ mod\ p$ which is the public key of Alice. When Alice gets the pair of messages, she uses the first one in order to compute $g^{xy} mod\ p$. Then she finds a number $(g^{xy})^{-1}\ mod\ p$ and so she can get the message as follows:

$$m = m*g^{xy}(g^{xy})^{-1}\ mod\ p = m\ mod\ p = m$$

This scheme requires two exponentiations per each transmitted block, and therefore it is a little bit demanding in terms of computational effort.

Actually this was a first proposal, not really employed, what happens in the reality is that all the asymmetric algorithms proposed by the literature based on the public key are somewhat limited and they should be compared to great schemes of symmetric encryption. Practical users, they all of them prefer symmetric encryption rather than asymmetric encryption.

Hybrid Cryptography

At the end of all this topic, we ask ourself which is the real usage of RSA or El Gamal. The answer is simple. We have understood that the Public Key Cryptography provides several benefits but it is too demanding under a computational point of view, and therefore what the people prefer is using the Hybrid Cryptography. The Hybrid Cryptography is an approach to *confidentiality* where the Public Key Cryptography is used in order to encrypt and exchange a key K which will be used for the Symmetric Encryption schemes. This because symmetric encryption is much faster than asymmetric encryption and so in order to exchange messages it is preferred to use symmetric encryption. Summing up, one uses RSA or El Gamal in order to communicate the key K that will be later used for confidentiality purposes. Indeed, the secret key sent is small compared with messages in a confidential environment, and therefore since only one secret key is needed in symmetric encryption, RSA or El gamal can be performed in order to encrypt and decrypt such a 'small' key. So the final message is:

Even if public key is providing new powerful results it's much more convenient using symmetric key cryptography for confidentiality purposes, and you can use public key cryptography for sharing and establishing keys to be employed for the symmetric encryption.