RSA and EL gamal only

Information Systems Security
Series 5 - RSA, Rabin, ElGamal

November 17th, 2021

Asymmetric Algorithms

This series is about the three following asymmetric algorithms : RSA, Rabin and ElGamal.

Exercice 1 : RSA

We have the public key (n, e) and private key (n, d) (n is built as the product of two prime numbers p and q, e is the exponent for encryption, and d the exponent for decryption).

1. We have p = 11, q = 17. Create a pair of keys (n,e) and (n,d) for RSA.
2. We have the public key of Alice (n=247, e=41). Send Alice the message "28" encrypted with her key. **Use fast exponentiation** (see Annex A).
3. Now, we want to find Alice's private key. Factorise n, then find the private exponent d (you can use the Euclide's extended algorithm, see Annex B). Then, test it by deciphering the ciphertext from the previous question (you should obviously find the message originally sent, which was "28").

Exercice 2 : Rabin

Let us recall the Rabin encryption scheme :

- *Key generation* : Alice chooses two big different prime numbers p and q, kept secret, such that $p \equiv q \equiv 3 \mod 4$ (to ease decryption). Alice computes $n = p \cdot q$.
- *Key Distribution* : Alice gives her public key n to Bob (her secret key is (p, q)).

- *Encryption* : Bob ciphers the message m by computing $c = m^2 \mod n$. Bob sends c to Alice.
- *Decryption* : Alice computes $m = \sqrt{c} \mod n$ the following way :
  1. Alice starts by computing $m_p = \sqrt{c} \mod p = c^{\frac{p+1}{4}} \mod p$, and m_q similarly (this is why we want $p \equiv q \equiv 3 \mod 4$, to have integer powers of c).
  2. Alice then solves the equation system $m = m_p \mod p$; $m = m_q \mod q$, and obtains 4 potential solutions for message m (we note $p_1 = p^{-1} \mod q$ et $q_1 = q^{-1} \mod p$) :
     $m_1 = (m_p \cdot q \cdot q_1 + m_q \cdot p \cdot p_1) \mod n$
     $m_2 = (m_p \cdot q \cdot q_1 - m_q \cdot p \cdot p_1) \mod n$
     $m_3 = (-m_p \cdot q \cdot q_1 + m_q \cdot p \cdot p_1) \mod n$
     $m_4 = (-m_p \cdot q \cdot q_1 - m_q \cdot p \cdot p_1) \mod n$

You are Bob, and you receive Alice's public key, which is n=253 :

1. Cipher the message m=134.
2. Factorize n to find p and q.
3. Test p and q by deciphering the ciphertext you sent to Alice : Compute m_p et m_q, then find the 4 potential messages m_1, m_2, m_3 and m_4 (You should find m=134 as one of these four results).

Exercice 3 : ElGamal

- *Key generation* : Alice generates a big prime number p, and a generator α of the multiplicative group $\mathbb{Z}_p^*$. Alice then generates a random number $a < p - 1$, and computes $\alpha^a \mod p$.
  Alice's public key is (p, α, $\alpha^a \mod p$), and her private key is a.
- *Encryption* : Bob ciphers the message m ($m < p$) by generating a random number $k < p - 1$. Bob then computes two values : $\lambda = \alpha^k \mod p$ and $\sigma = m \cdot (\alpha^a)^k \mod p$. Bob sends these two values as the ciphertext : c = (λ,σ).
- *Decryption* : Alice first computes $x = \lambda^{p-1-a} \equiv \lambda^{-a} \equiv \alpha^{-ak} \mod p$. Then, Alice computes the message as $m' = x \cdot \sigma \mod p$ (and we can see that $m' = \alpha^{-ak} \cdot m \cdot \alpha^{ak} = m$, and that's the original message).

You receive Alice's public key, which is (17,3,12) :

1. Cipher the message 2.
2. With α and $\alpha^a \mod p$, find the private key a.
3. Test the a found by deciphering the previous cipher.

Exercice 4 : Conclusion

1. Considering you just found all of Alice's private keys, are these algorithms unbreakable ?
2. Yet, these algorithms are used everywhere, and are considered as some of the most reliable algorithms today. Why are they usually hard to break, and why was it easy in our examples ? Will these algorithms still be reliable in the future ?

Annex A - Fast Exponentiation

To compute efficiently big exponents in a modular environment, we use what is called fast exponentiation :

Let's say we want to compute $3^{42} \mod 25$. The fast exponentiation idea is to use the exponents which are a power of two :

$3^1 \mod 25 = 3$
$3^2 \mod 25 = 9$
$3^4 \equiv 3^2 \cdot 3^2 \equiv 9 \cdot 9 \equiv 81 \mod 25 = 6$
$3^8 \equiv 6 \cdot 6 \equiv 36 \mod 25 = 11$
$3^{16} \equiv 121 \mod 25 = 21$
$3^{32} \equiv 21 \cdot 21 \equiv 441 \mod 25 = 16$

Then, we can write the power as $42 = 32 + 8 + 2$, with powers of two, and we can easily compute :

$$3^{42} \equiv 3^{32} \cdot 3^8 \cdot 3^2 \equiv 16 \cdot 11 \cdot 9 \equiv 1584 \mod 25 = 9$$

Annex B - Euclide's extended algorithm

This algorithm is used to find the decryption exponent d for RSA.

d is the inverse of e modulo $\phi(n)$ ($ed \mod \Phi(n) = 1$ implies $x^e d \mod n = x^(k\phi(n) + 1) \mod n = x$).

With this algorithm, for two integers a and b, $a \geq b$, we can obtain $r = pgdc(a, b)$ and s, t such that $s \cdot a + t \cdot b = r$ (these are called Bezout coefficients). If a and b are co-prime, then s is the inverse of a modulo b, and t is the inverse of b modulo a.

So in this case, we will use it with $a = \phi(n)$ and $b = e$. We will find the inverse t of $e \mod \phi(n)$. Then $d = t \mod Phi(n)$ (we need this to verify that t is such that $0 < t < Phi(n)$, as Bezout may return negative numbers).

Here is the algorithm : ($\div$ is the **integer division** !):

r_0 := a;
r_1 := b;
$s_0 := 1$;
$s_1 := 0$;
$t_0 := 0$;
$t_1 := 1$;
$q_1 := r_0 \div r_1$;

repeat until $r_{i+1} == 0$
$r_{i+1} := r_{i-1} - q_i * r_i$;
$s_{i+1} := s_{i-1} - q_i * s_i$;
$t_{i+1} := t_{i-1} - q_i * t_i$;
$q_{i+1} := r_i \div r_{i+1}$;
end repeat;

return r_i, s_i, t_i;