:: Misc ::

RSA and Diffie Hellman explained (posted 01/04/07 by MatD)

Cryptosystems explained

I was always upset to see that cryptography was made so difficult because so many cryptical maths signs were used.
I wrote this tiny article to easily explain how RSA and Diffie Hellman key exchange are working. This article is not targeted on mathematics fans so they won't find something new here;-) I hope it can help you to start in the cryptosystem field.

Modulo calculation

The first thing to know in cryptography, is how to handle with modulo calculations. Modulo, what ? No fear it's very easy to understand.

Just take this example : when a friend of you is telling : "*I'll be back home in 32 hours*" how are you converting this in days ?

32 hours = 24 hours + 8 hours

This is equivalent to write : 32 = 24x1 + 8 . Inconsciously you made : 32 divided by 24 and the remainder is 8

So if we turn this in a mathematical way you can write : 32 mod 24 = **8**

Another example : If you say "*I have a train in 125 minutes*". How are you proceeding to know how many hour(s) and minutes you have to wait ? Very easy : you are going to divide 75 by 60 and then 15 will remain. You have a "full" hour and 15 minutes :

125 = 60x2 + **5** . You can also write : 125 mod 60 = **5**

Here is another example : **125** mod **32** = **29**. Why ? because 125 = **32** x **3** + **29**

Just think about "doing **3** packets of **32** starting at **125** " and if the remainder is not 0, the number remaining (the red one) will be the solution of your equation.

24 mod 12 = **0** because we have a remainder that equals **0** => 24 = 2 x 12 + **0**

Prime numbers

Many properties in cryptography are based on multiplication and factorisation of prime numbers. A primer number is a whole number (integer) that can only be divided evenly by itself.

e.g. 2, 3, 5, 7,11,13,17,19,23,27 are prime numbers

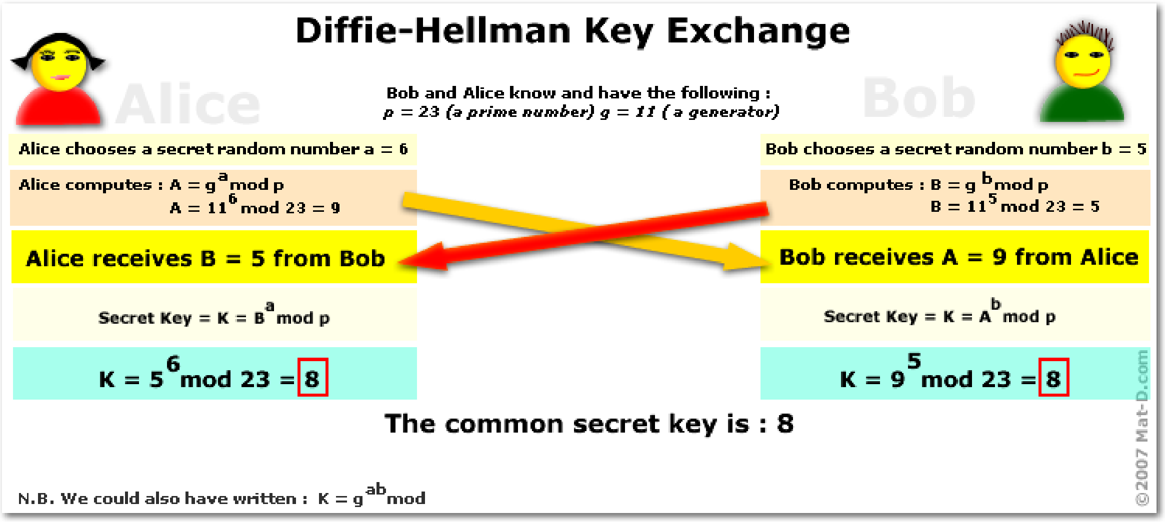
24 is not a prime number because it can be divided by 2,3,4,8,12

Generating big prime numbers enables us to have a higher security because they are very hard to find (due to the property of being divisible by itself). Make the test by yourself : find a prime number higher than 2000. You will need a few seconds to verify the divisors so do computers for very huge prime numbers.

If you are interested, here is a page listing the 1000 first prime numbers : http://primes.utm.edu/lists/small/1000.txt

The Diffie-Hellman key exchange

It's always a problem when you want to share a key to another person, because you can't be sure that the "line" or transmission mode is sure enough. That's why Diffie-Hellman key exchange algorithm was created. The following picture explains the whole process :



RSA

The acronym RSA stands for Ron Rivest, Adi Shamir and Leon Adleman, the three "inventor" of this system. It was in 1977.

RSA is very easy to understand. We will have two persons (say Alice and Bob). They want to communicate together but don't want to be listened by someone else. So Alice will need to encrypt her message and Bob to decrypt it.

We will first choose two prime numbers **p** and **q** so that **p** and **q** have no common divider. In other words the greatest common divider of **p** and **q** is 1.

Finding p and q

Let's take as an example **p** = 7 and **q** = 3. It's clear that gcd(7,3) = 1

Now we are going to find **n**. **n** equals **p** x **q** . So in our example we will have :

n = **p** x **q** = 7 x 3 = **21**

Alice also needs to compute her private (or so called secret) key and the public key. Bob will also have a private key. With his secret/private key and Alice's public key Bob will be able to decipher Alice's message.

In order to compute both keys we need to find **e** and **d** so that: **e** x **d** mod **phi**(n) = 1.

Don't panic ;-)

phi(n) = (**p**-1)(**q**-1) = (**7**-1)(**3**-1) = **12**

Now we must find **e** so that **e** has no common factor with **phi**(n). We can also say that gcd(e,phi(n)) = 1 or gcd(e,12).

Let's take **e** = 7 (a quick check gives uses gcd(7, 12) = 1) for the sake of showing how the process works.

Now we need to find out **d** . Finding **d** is perhaps the most tricky thing of the whole RSA algorithm.

We can write this : **7 d mod 12**. To solve this we simply write :

12 = **7** x 1 + **5** (then we pull down the numbers 5 and 2)

7 = **5** x 1 + 2

5 = 2 x 2 + 1 => Stop !

Extended Euclidean Algorithm

When you have a 1 on the most right side of your equation you stop ! What we just did is called "Extended Euclidean algorithm". "Funny" name, but easy to solve ;-) Now that we have the number 1 on the right side we can "invert" our equation by writing :

1 = **5** - **2** x 2

and then you will replace by **2** by 7 = **5** x 1 + **2**

1 = **5** - (7 - **5** x 1) x 2

1 = 5 x 3 - 2 x 7

and now we will replace 5 by 5 = **12 - 7 x 1**

1 = **(12 - 7 x 1)** x 3 - **2** x 7

and by replacing **2** by 5 = 12 - 7 x 1

If we group the 7s and the 12s we have :

1 = **12** x 3 - 5 x **7**

What we in fact were trying to reach is a result with following form 1 = 12 x A - 5 x B with A and B unknown. These two unknown numbers are in maths called Bezout coefficients.

Don't forget that we are working in the ensemble Z12. We can "erase" 3 x 12 because we already have seen that 3 x 12 = 36 and 36 mod 12 = 0

Now we are going to watch the **5** x 7 part. we see the number **5** coming from **e** . But we want **d** !

we can now write in this form : -**5** + 12 = 7 . This is called the inverse of -**5** mod 12. So inverse of -**5** mod 12 = 7

and **d** is going to be : 7 + 12 because we have in our equation *mod 12*

this gives **d** = **19**.

What happens next if instead of -5 we had a positive number such as 3 ? You would have calculated : d = 3 + 12 = 15

We can now check our result : **e** x **d** = 7 x 19 = **133 .** And if we decompose **133** we have **: 133 = 12 x 11 + 1** The remainder is **1,** our calculation is correct.

Let's encrypt !

In order to simplify the calculation, we are going to take following values for **n**, **p**, **q**, **e** and **d**. We take 5 for the value of **e** because gcd(5,phi(n)) = gcd(5,12) = 1. This will simplify the process.

To sum up : **n** = 21 **p** = 7 **q** = 3 **e**= 5 **d** = 19

The couple (**d**,**n**) represents the private key (decrypting key)
The couple (**e**,**n**) represent the public key (encrypting key)

Now we are going to compute a message say a number **m** = 12 (we take this value because this value must be smaller than **n**) and **c** our encrypted message that is going to be sent to Bob.

We just need to write $c = m^e \bmod n$

The result of this formula is : $\mathbf{c} = 12^{5} \mod 21$ and then we see $\mathbf{c} = 3$

Decrypting is easy, you just need **d** (the private key) and apply following formula : $\mathbf{m} = \mathbf{c}^{\mathbf{d}} \mathbf{mod\ n}$
So $\mathbf{3}^{5} \mod 21 = 12$ <=> $\mathbf{m} = 12$ Bingo ;-)

What you just need to know about RSA :

Encrypting

$\mathbf{c} = \mathbf{m}^{\mathbf{e}} \mathbf{mod\ n}$ with the couple (e,n) the public key

Decrypting

$\mathbf{m} = \mathbf{c}^{\mathbf{d}} \mathbf{mod\ n}$ with the couple (d,n) the private key

Matthieu DERU

Contact me via iChat !

Contact