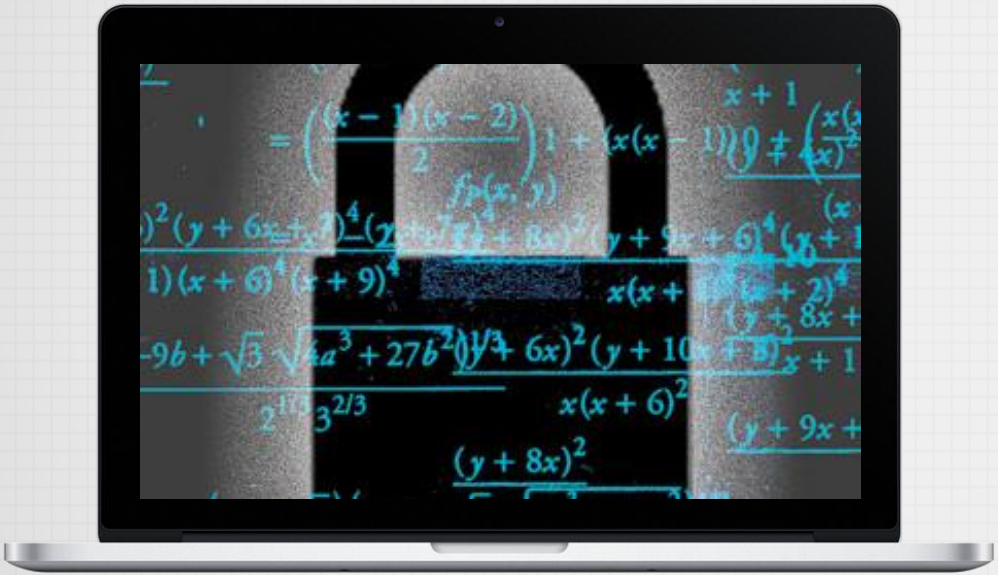
CRYPTOGRAPHIE

ISN | 2012/2013

GREGORY LINFORD

INTRODUCTION: LA CRYPTOGRAPHIE



LA CRYPTOGRAPHIE ASSURE LA CONFIDENTIALITÉ, L'AUTHENTICITÉ ET L'INTÉGRITÉ DES MESSAGES.

CRYPTOGRAPHIE SYMETRIQUE

On parle d'algorithme symétrique lorsque la même clé sert à la fois à chiffrer et déchiffrer un message.

CRYPTOGRAPHIE ASYMETRIQUE

Dans ce cas le message est chiffré par une clé dite publique et déchiffré par une clé dite privée.

LE RSA ? UN PEU D'HISTOIRE

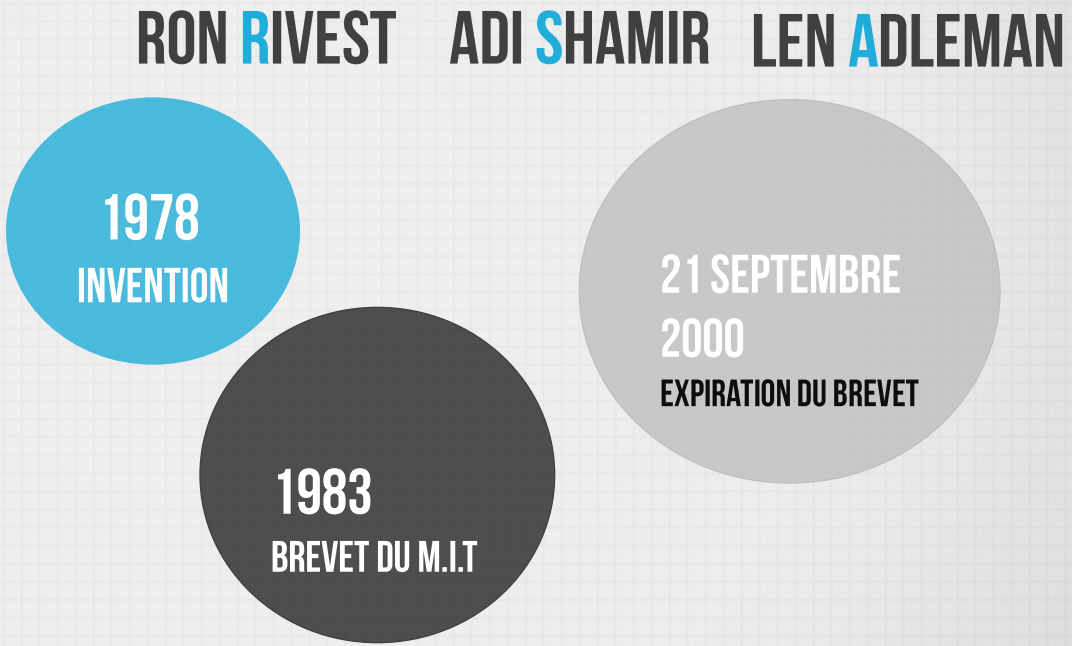
A L'ORIGINE

Le système de cryptage RSA a été inventé en 1978 par Ron **R**ivest (Cryptologue américain), Adi **S**hamir 78 (Expert en cryptanalyse israélien) et Len **A**dleman (Chercheur en informatique et biologie moléculaire américain).
Ce fut le premier cryptage dit « asymétrique »

AUJOURD'HUI ?

Le RSA est aujourd'hui un système universel servant dans une multitude d'applications comme:

- Les télécommunications
- Transaction sécurisée sur Internet
- Finance
- Carte a puce
- Etc…



PRINCIPE DE FONCTIONNEMENT DU RSA

1. GENERATION DES CLES

- Choisir 2 nombres premiers **p** et **q** du même ordre de grandeur.
- On calcule **n : n = p*q**
- On calcule **e** tel que **e** n'ai aucun facteur commun avec **(p-1)(q-1)**.
- On calcule **d** tel que **ed mod (p-1)(q-1) = 1**

2 . CRYPTAGE

- Convertion d'un message **M** en nombre (Ascii ou autre)
- On découpe **M** pour que chaque partie soit inferieure à la longueur de **n**.
- On encrypte chaque partie avec la clé publique **(e, n)** : **Nx^e mod n**
- On rassemble les parties pour former la chaine cryptée **C**.

3 . DECRYPTAGE

- On redécoupe le message crypté **C** en formant des nombres inferieurs à **n**.
- On le decrypte avec la clé privée **(d, n)** : **Cx^d mod n**
- On rassemble les parties et on retrouve notre message **M**.

EXEMPLE : RSA

On prend deux nombres premiers : **p = 3** et **q = 47**
(Plus les nombres sont longs plus le « crackage » des clés est difficile.)

n = p*q = 3 * 47 = **141**

f = (p-1)(q-1) = **92**

e = ?

Algorithme d'Euclide :
Prenons nos deux nombres : 47 et 3
47 > 3 donc : 47 = 3 *15 + 2
3 > 2 donc 3 = 2*1 + 1
Donc PGCD(47 ; 3) = 1

```
1   import java.math.BigInteger;
2   void main(){
3   // Les variables
4               int p = 47;
5               int q = 3;
6                 int n = p*q ;
7                 int f = (p-1)*(q-1) ;
8                 int e = 14; //Theoreme de Bezout
9                 int c = 0;
10                int d = -1;
11  ////////////////////////////////////
```

EXEMPLE : RSA (SUITE)

p = 3
q = 47
n = 141
f = 92

e = ?

Théorème de Bézout :

Le Théorème de Bézout est très important dans le RSA
car il permet de calculer **e** facilement. Il prouve l'existence d'une solution
a*u + b*v = PGCD(a,b)
On cherche à calculer u et v tel que 47*u + 3*v = 1
(a) 1 = 3 - 2 * 1 (Démontré précédemment avec Euclide)
(b) 2 = 47 - 3 * 15
On remplace (b) dans (a) :
1 = 3 - (47 - 3 * 15) * 1
1 = 3 (1 + 15) + 47 * (-1)
Donc on a bien 47*u + 3*v = 1
avec u= -1 et v= 16
e = u + v

e = -1 + 16 = 15

```
12  // calcul du pgcd de p et q , et affichage
13      System.out.println("pgcd(" + p + "," + q + ") = " + pgcd(p, q));
14  if (pgcd(p, q) == 1) {
15  //calcul du e
16  while(pgcd(f, e) != 1) {
17  e = e + 1 ;
18  }
```

EXEMPLE : RSA (SUITE - 2)

p = 3
q = 47
n = 141
f = 92
e = 15

On calcule d de la même manière que e mais avec comme nombres premiers e et f.

92-6*15 = 2
15 - 7*2 = 1
15 - 7*(92-6*15) = 1
15 - 7*92 + 43 * 15 = 1
43 * 15 - 7*92 = 1
d = 43 car -7<0

```
19  // verification des pgcd de f et e
20  System.out.println("pgcd(" + f + "," + e + ") = " + pgcd(f, e));
21  //calcule du d
22  while(c != 1) {
23  d = d + 1 ;
24  c = (e*d)%f;
25  }
```

EXEMPLE : RSA (SUITE -3)

p = 3
q = 47
n = 141
f = 92
e = 15
d = 43

(e, n)

(15, 141)

CLE PUBLIQUE

(d, n)

(43, 141)

CLE PRIVEE

EXEMPLE : RSA (SUITE -4)

On transforme un mot en ASCII.
isn = 105 115 110
Puis il nous faut découper la chaine 105115110 en « tranche » équivalent à la longueur de n-1 soit pour n=141 on découpera en « tranche » de 2.
La chaine n'est pas un multiple de 2, je complète donc par un 0.
La chaine finale prête à encrypter est : 1051151100

```
33  // la chaine a encrypter en ASCII
34  println("Entrer un mot a encrypter");
35  String mots = readString();
36  char[] pc = mots.toCharArray();
37  int i = (int) pc[0];
38  int s = (int) pc[1];
39  int nn = (int) pc[2];
40  println("ASCII : " + i + s + nn);
```

```
60  //  On complete par des 0 ci necessaire
61  String codestringfinal = null ;
62  if(codestring.length()%(n-1) == 0) {
63      codestringfinal = codestring;
64  } else {
65  int gg =  codestring.length()%(nstring.length()-1);
66  int aa = Integer.parseInt(codestring);
67  int bb = (int) (aa) * (int)(Math.pow(10, gg));
68  codestringfinal = String.valueOf(bb);
69  }
```

p = 3
q = 47
n = 141
f = 92
e = 15
d = 43
(15, 141)
(43, 141)

EXEMPLE : RSA (SUITE -5)

p = 3
q = 47
n = 141
f = 92
e = 15
d = 43
(15, 141)
(43, 141)

La chaine finale :
1051151100
Découpage :
10 51 15 11 00
On effectue le chiffrement avec la clé publique.
M^e mod n
Pour notre exemple:
10^15 mod 141 = 40
On fait de même pour toutes les parties et on les réuni.
Notre chaine encryptée est: 40 81 30 137 0

```
74  /*
75  FAIRE L'ENCRYPTAGE (Decouper la chaine en n-1 ,
76  ex: si n fait 3 caractere
77  decouper tout les 2 caractere XX XX XX XX ,
78  dans ce cas n fait 3 caracteres (141)
79  */
80  String s1 = codestringfinal.substring(0, nstring.length()-1);
81  println("Premiere partie: " + s1 );
82  String s2 = codestringfinal.substring(nstring.length()-1, (nstring.length()-1)*2);
83  println("Deuxieme partie: " + s2 );
84  String s3 = codestringfinal.substring((nstring.length()-1)*2, (nstring.length()-1)*3);
85  println("Troisieme partie: " + s3 );
86  String s4 = null;
87  if (((nstring.length()-1)*4) > codestringfinal.length()) {
88  s4 = null;
89  } else {
90  s4 = codestringfinal.substring((nstring.length()-1)*3, (nstring.length()-1)*4);
91  println("Quatrieme partie: " + s4 );
92  }
93  String s5 = null;
94  if (((nstring.length()-1)*5) > codestringfinal.length()) {
95  s5 = null;
96  } else {
97  s5 = codestringfinal.substring((nstring.length()-1)*4, (nstring.length()-1)*5);
98  println("Cinquieme partie: " + s5 );
99  }
100 // convertion en INT des parties
101 int num1= Integer.parseInt(s1);
102 int num2= Integer.parseInt(s2);
103 int num3= Integer.parseInt(s3);
104 int num4= Integer.parseInt(s4);
105 int num5= Integer.parseInt(s5);
106 //encrypter la partie decouper^d modulo n
107 BigInteger num1mix = BigInteger.valueOf(num1).modPow(BigInteger.valueOf(e), BigInteger.valueOf(n));
108 BigInteger num2mix = BigInteger.valueOf(num2).modPow(BigInteger.valueOf(e), BigInteger.valueOf(n));
109 BigInteger num3mix = BigInteger.valueOf(num3).modPow(BigInteger.valueOf(e), BigInteger.valueOf(n));
110 BigInteger num4mix = BigInteger.valueOf(num4).modPow(BigInteger.valueOf(e), BigInteger.valueOf(n));
111 BigInteger num5mix = BigInteger.valueOf(num5).modPow(BigInteger.valueOf(e), BigInteger.valueOf(n));
112 // la chaine encrypter
113     System.out.println("Notre chaine encrypter est : " + num1mix + " " + num2mix + " " + num3mix+ " " + num4mix + " " + num5mix);
114
```

EXEMPLE : RSA (SUITE -6)

p = 3
q = 47
n = 141
f = 92
e = 15
d = 43
(15, 141)
(43, 141)

Notre chaine encryptée est : 40 81 30 137 0
Pour décrypter notre chaine, on utilise la clé privée.
Cette fois ci on découpe de manière à ce qu'aucun nombre ne soit supérieur à n (141).
On décrypte de la manière suivante :
C^d mod n
Dans notre exemple : 40^43 mod 141 = 10
On effectue le même calcul avec les autres parties, notre chaine décryptée est :
105115110
On retrouve bien notre mot ‘’isn’’ en ASCII.

```
115  //// DECRYPTAGE
116  // question decryptage oui ou non
117  System.out.println("Voulez vous decrypter la chaine ?");
118  String reponse1 = readString();
119  String oui = "oui" ;
120  //convertion oui et reponse1 en numero (hash)
121  int ouioui= oui.hashCode();
122  int reprep= reponse1.hashCode();
123  // si OUI
124  if (ouioui == reprep) {
125  // Decryptage avec la clé privée
126  BigInteger decr1 = num1mix.modPow(BigInteger.valueOf(d), BigInteger.valueOf(n));
127  BigInteger decr2 = num2mix.modPow(BigInteger.valueOf(d), BigInteger.valueOf(n));
128  BigInteger decr3 = num3mix.modPow(BigInteger.valueOf(d), BigInteger.valueOf(n));
129  BigInteger decr4 = num4mix.modPow(BigInteger.valueOf(d), BigInteger.valueOf(n));
130  BigInteger decr5 = num5mix.modPow(BigInteger.valueOf(d), BigInteger.valueOf(n));
131  System.out.println("Ascii de la chaine decrypter: " + decr1 + decr2 + decr3 + decr4 + decr5 );
132  } else {
133    System.out.println("....");
134  }
135
136  } else {
137      System.out.println("Les nombre ne sont pas premiers");
138  }
139
140  ////////////////////PGCD
141  }
142    static int pgcd (int a, int b) {
143
144      if(a<b) // on prend le premier argument plus grand
145        return (pgcd(b,a));
146      else if(b==0) // arrêt
147        return (a);
148      else // on poursuit l'algorithme
149        return (pgcd(b,a%b));
150        }
```

THE END

Source:

- Wikipedia
- http://fr.wikipedia.org/wiki/Rivest_Shamir_Adleman
- http://villemin.gerard.free.fr/Crypto/RSA.htm
- http://docs.oracle.com/javase/6/docs/api/java/lang/Character.html

Logiciel:

- Java's Cool
- Eclipse
- Microsoft Powerpoint 2010