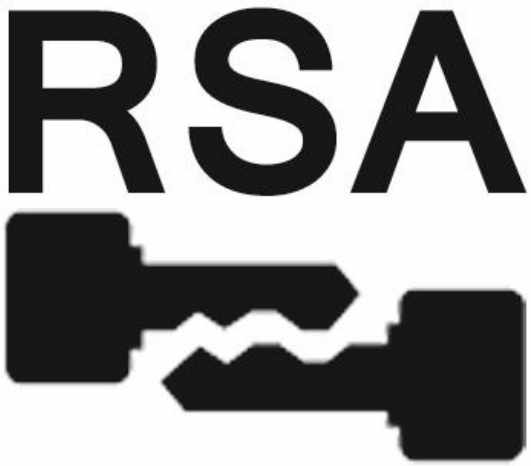
2010-2011

Renaud Berthe & Anthony Monier
Sous la tutelle de Mr Sokol

IUT de Béthune Département R&T
2010 - 2011



[RSA : Rivest Shamir Adleman]

Présentation du RSA, Comment génère t'on les clés publique, privé ?, Algorithme de Bézout, Pseudo-code RSA, Présentation pratique en JAVA, Etude de la sécurité.

Sommaire

Introduction & phasing

Durant notre deuxième année à l'IUT de Béthune, nous avons l'opportunité d'effectuer notre projet tutoré sur l'étude de la cryptographie RSA et sur le développement d'un logiciel de communication tchat sécurisé avec cet algorithme de cryptographie.

Pour mettre à bien notre projet, nous avons dans un premier temps mis en place un phasing, que vous trouverez à la page suivante.

Ce rapport présentera toutes les démarches techniques effectuées durant ce projet tutoré. Il est répartit sur 6 chapitres qui commenceront par une présentation général de la cryptographie, suivie d'une explication sur le RSA, son histoire, ses usages. Ensuite il décrira l'ensemble des compléments mathématiques nécessaire à sa réalisation puis il abordera les étapes de développement du logiciel de tchat. Ce rapport fera une bref analyse de la sécurité du RSA et conclura sur ceux qui nous aura apporté durant le projet.

Phasing du projet tutoré : Etude de la cryptographie RSA

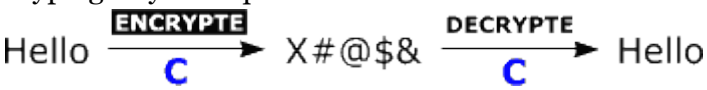
04/10/10 au 10/10/10	Generalité sur la cryptographie : Histoire & developement de la cryptographie
11/10/10 au 17/10/10	Focalisation RSA: principe de fonctionnement généralisé
18/10/10 au 24/10/10	Algèbre (1/2) : Nombre premiers & étude du théorème de Bézout
25/10/10 au 31/10/10	Vacance de la Toussaint
01/11/10 au 7/11/10	Algèbre (2/2) : Nombre premiers & étude du théorème de Bézout Explication à Mr Sokol
08/11/10 au 14/11/10	Principe des Clés Privées & Clées Publiques
15/11/10 au 21/11/10	
22/11/10 au 28/11/10	
29/11/10 au 5/12/10	Creation du plan du logiciel du transfert de données par cryptographie RSA
6/12/10 au 12/12/10	Mise en place des algoritmes (1/2)
13/12/10 au 19/12/10	Mise en place des algoritmes (2/2)
20/12/10 au 26/12/10	Vacance de Noël
27/12/10 au 2/01/11	
03/01/11 au 9/01/11	Programmation général du logiciel (1/3)
10/01/11 au 16/01/11	Programmation général du logiciel (2/3)
17/01/11 au 23/01/11	Programmation général du logiciel (3/3)
24/01/11 au 30/01/11	
31/01/11 au 06/02/11	Modification logiciel pour la visualisation (1/2)
07/02/11 au 13/02/11	Modification logiciel pour la visualisation (1/2) (Essai de finir le projet pour la porte ouverte)
14/02/11 au 20/02/11	
21/02/11 au 27/02/11	Vacance d'hiver
28/02/11 au 06/03/11	
07/03/11 au 13/03/11	
18 mars 2010 =>	Rapport projet tutorée doit être fini.
25 mars 2010 =>	Soutenance au matin

I] Introduction à la Cryptographie

La cryptographie assure la confidentialité, l'authenticité et l'intégrité des messages. Elle recourt à des secrets qui sont bien souvent des clés. Les premières utilisations de la cryptographie remontent à l'antiquité, notamment avec le chiffre de César, en référence à Jules César, qui l'utilisait pour protéger ses communications. Mais la cryptographie est bien plus ancienne que cela.

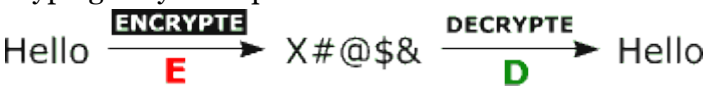
De nos jours, son utilisation s'est démocratisée par l'informatique.

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



Mais le problème est que si l'on demande un nombre important de clé pour un cryptage symétrique, il faudrait alors énormément de ressources pour que deux clés ne soit pas identiques.

C'est pourquoi, dans les années 1970, la cryptographie asymétrique a été mise au point par Whitfiel Diffie et Martin Hellman. Le principe de fonctionnement se repose sur une paire de clés privée et publique. La seconde étant dérivée de la première et peut être distribué à n'importe qui. Tandis que l'autre doit être tenu secrète. Dans ce cas le message est chiffré par la clé publique et déchiffré par la clé privée.
Cryptage Asymétrique :



Le problème est que le cryptage asymétrique demande une grosse puissance de calcul. Par exemple, dans notre cas l'algorithme asymétrique RSA est 1000 fois plus lent qu'un chiffrage symétrique.

C'est pourquoi l'on utilise souvent le chiffrage asymétrique pour débuter le dialogue entre deux machines et d'établir une connexion sécurisée, pour ensuite générer une clé secrète qui va servir de clé symétrique pour chiffrer les échanges et ainsi alléger la charge de calculs.

> Il est courant d'appelé le chiffrement « cryptage ». Or le chiffrement est le procédé qui consiste à rendre un texte inintelligible à la personne qui ne possède pas la clé de déchiffrement. Le terme cryptage n'apparait pas dans le dictionnaire français, en revanche on peut utiliser le mot crypter par exemple pour parler des chaines payants comme Canal +.
> De la même façon, le décryptage est l'action de déchiffrer un texte dont on ne possède pas la clé, alors le déchiffrage est le décodage d'un message légitime.

II] Introduction du RSA

1] Un peu d'Histoire

Le système de cryptage RSA a été inventé en 1978 par Ron **R**ivest (Cryptologue américain), Adi **S**hamir (Expert en cryptanalyse israélien) et Len **A**dleman (Chercheur en informatique et biologie moléculaire américain), d'où le sigle RSA. Ces trois auteurs avaient décidé de travailler ensemble sur le fait que le nouveau système de codage révolutionnaire de W.Diffie et M.Hellman dénommé : "Système à clé publique"(Cryptage Symétrique) était une impossibilité logique (qu'il présentait des failles). Même si ils n'ont pas réussi à l'époque à le prouvé, ils mirent en place un nouveau système de cryptage qui supplanta très vite celui de Diffie et Hellman, le RSA (le premier cryptage dit « asymétrique »).

1978	Invention du système de cryptage RSA
1983	Le RSA est breveté par le MIT aux Etats-Unis
21 septembre 2000	Le brevet expire et le RSA tombe dans le domaine publique

2] Le RSA aujourd'hui

Les RSA est aujourd'hui un système universel servant dans une multitude d'applications. Au fil des années, il a supplanté tous ses concurrents, soit parce que l'on y a trouvé des points faibles, soit parce que ils n'ont pas assez été étudié pour prouver leurs efficacité face au RSA. La technologie du RSA était protéger par un brevet jusqu'en septembre 2000, aujourd'hui il est dans le domaine publique, elle a été commercialisée par plus de 350 entreprise et dans plus de 300 millions de programmes vous l'utiliser sans aucun doute tous les jours sans le savoir. On l'utilise notamment dans les transactions sécurisées sur internet, il est aussi implanter dans de nombreux standards informatique comme le Society for Worldwide Interbank Financial Télécommunication Standard, le French Financial Industry's ETEBAC-5 ou encore le X9.44 draft Standard for the U.S. Banking Industry. Le RSA est aussi implanter dans les systèmes d'exploitation de Sun, Apple, Microsoft et Nobel. Egalement intégré dans les cartes Ethernet, les Cartes à puce Bancaires, mais aussi dans grand nombre d'institutions gouvernementales, militaires et universitaires. Bref, le RSA est partout.

III] Principe du RSA

1] Présentation général.

Le principe du RSA est relativement simple. Utilisons deux utilisateur classique, Alice et Bob, avec Bob qui veut envoyer un message à Alice mais en utilisant le RSA.

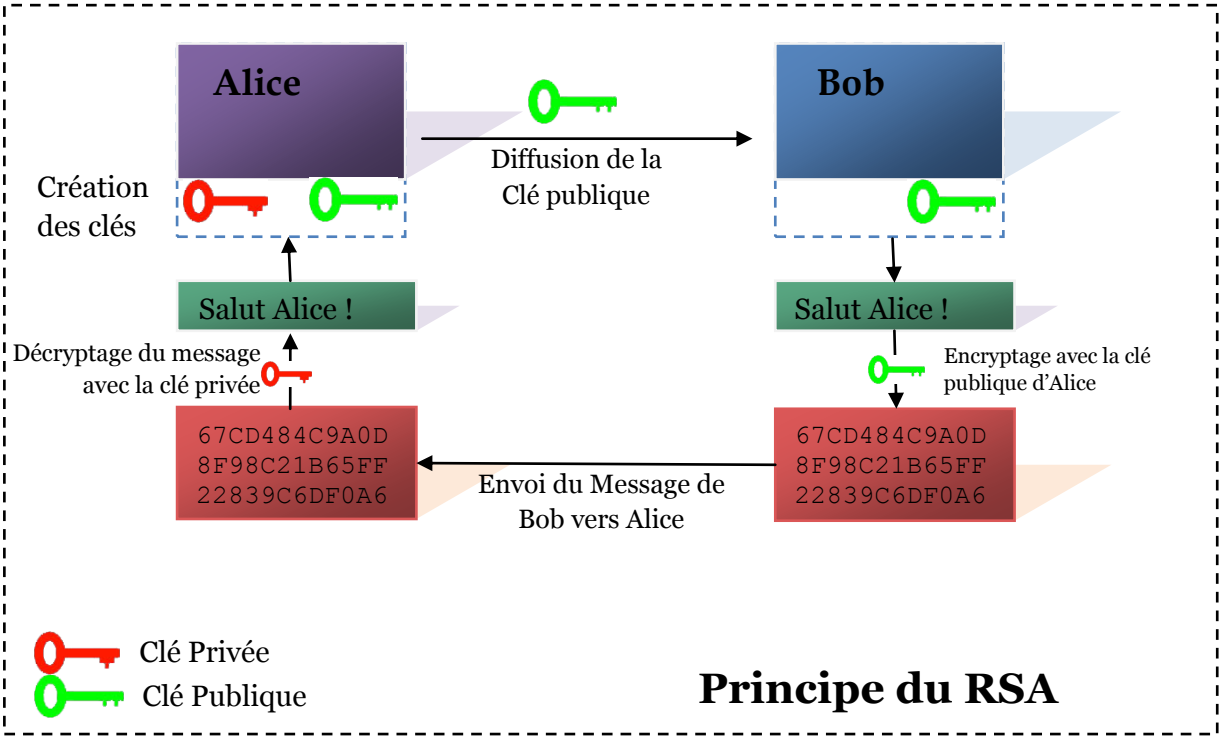
Alice va générer deux clés :

Une *clé publique* qu'elle diffusera aux personnes voulant lui parler. Cette clé sert à crypter et uniquement crypter les messages, comme nous le verront plu tard on ne peut pas décrypter les messages avec la clé publique.

Une *clé privée* qu'Alice gardera bien cacher des autres utilisateurs. Cette clé sert à décrypter tous les messages qui ont été crypté avec sa clé publique.

Alice envoie donc sa clé publique a Bob pour qu'il puisse lui envoyer en message crypter. Puis Alice récupère le message crypté de Bob et le décrypter à l'aide de sa clé Privé.

Illustration : Message de Bob vers Alice



2] Génération des Clés.

Maintenant que le principe est compris passons maintenant à l'aspect mathématique du RSA, comment génère-t-on les clés, comment crypter le message et comment le décrypter.
Soit **m** le **message en clair** (non crypté).
Soit **c** le **message encrypté**.
Soit **(e,n)** le couple qui constitue la **clé publique**.
Soit **(d,n)** le couple qui constitue la **clé privé**.

- On choisit deux grands nombres premiers **p** et **q** du même ordre de grandeur.
- On calcule **n : n= p*q**
- On calcul **e** tel que **e** n'ai **aucun facteur commun avec (p-1)(q-1).**
  Pour cela on va utiliser l'**algorithme de Bachet - Bézout.**
  (Conféré : *Compléments Mathématiques*).
  Apres avoir calculé **e**, on a la **clé publique (e,n)**.
- On calcule **d** tel que **ed mod (p-1)(q-1) = 1**
  (l'opération modulo (mod) est décrite dans les *Compléments Mathématiques*).
  Apres avoir calculer **d**, on a la clé **privé (d,n)**.

Maintenant que nous avons nos couples de clés (publique et privé) nous pouvons encrypter nos message et les décrypter.
Pour cela on effectue les opérations suivantes :
Pour **encrypter** le message : **c = m^e mod n**
Pour **décrypter** le message : **m = c^d mod n**

3] Exemple.

On choisit d'illustrer l'algorithme sous forme d'un exemple simple :
On prend deux nombres premier : **p = 47** et **q = 71**
Bien sûr en réaliser ces nombres est beaucoup plus grand, 1024 chiffres dans notre réalisation en java.
On calcul **n = p*q = 47 * 71 = 3337**
(p-1)(q-1) = 3220
e = 79 (démonstration avec le Théorème de Bezout)
On calcul d = = **1019**
On vérifie que e*d = 80501 = 1 mod 3220
Prenons un message préalablement transformé en nombre garce par exemple à la table ASCII
m = 6882326879666683
On découpe **m** : m_1=688 m_2=232 m_3=687 m_4=966 m_5=668 m_6=3
On calcul chaque bloc : $\mathbf{c_x = m_x^e \bmod n}$ (Grace a au couple **(e,n), clé publique**)
On trouve c_1=1570 c_2=2756 c_3=2091 c_4=2276 c_5=2423 c_6=158
Le message crypter est donc : c = 157027562091227624231 58
On retrouve m en calculant $\mathbf{m_x = c_x^d \bmod n}$ (Grace a au couple **(d,n), clé privé**)

IV] Compléments Mathématiques.

1] Algorithme d'Euclide.

Avant d'aborder le théorème de Bachet - Bézout il faut déjà avoir compris le l'algorithme d'Euclide, nous nous proposons donc de l'étudier rapidement ici.
L'algorithme d'Euclide sert à calculer le PGCD (Plus Grand Commun Diviseur) mais aussi à calculer les différents coefficients dans la formule de Bezout.
Le calcul du PGCD de a et b, deux nombres entiers naturels utilise la division euclidienne de a par b, tel que **a=b*q+r** avec **r<b.** Tout diviseur de et b divise r=a-b.q. Par réciproque, tout diviseur commun de b et r divise aussi a=b.q+r. Le calcul du PGCD de a et b se ramène au calcul de b et r. En réitérant, le dernier reste non nul est le PGCD recherché.

Exemple :
Prenons deux nombres : 383 et 127
383 > 127 donc : 383 = 127 *3 + 2
127 > 2 donc 127 = 2*63 + **1**
Donc PGCD(383 ; 127) = **1**
On rappelle que si le PGCD de deux nombre est égale à 1, alors ils sont premiers entre eux.

2] Théorème de Bachet – Bézout.

[Retour à Génération des Clés.]

Le Théorème de Bachet - Bezout 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)
Pour le RSA on cherche à calculer e tel qu'il n'est aucun facteur commun avec (p-1)(q-1), c'est à dire que le PGCD est égale à 1 (premier entre eux).

Exemple :
Reprenons l'exemple précédent avec 383 et 127.
On cherche à calculer **u** et **v** tel que **383*u + 127*v = 1**
(a) 1 = 127 - 2 * 63 (Démontré précédemment avec Euclide)
(b) 2 = 383 - 127 * 3
On remplace (b) dans (a) :
1 = 127 - (383 - 127 * 3) * 63 (on développe)
1 = 127 - 383*63 + 127*189
1 = 127 (1 + 189) + 383 * (-63)
Donc on a bien **383*u + 127*v = 1** avec **u= -63 et v= 190**
Pour le RSA, on prendra e=v=190 car u<0.

3] L'opération Modulo (mod).

[Retour à Génération des Clés.]

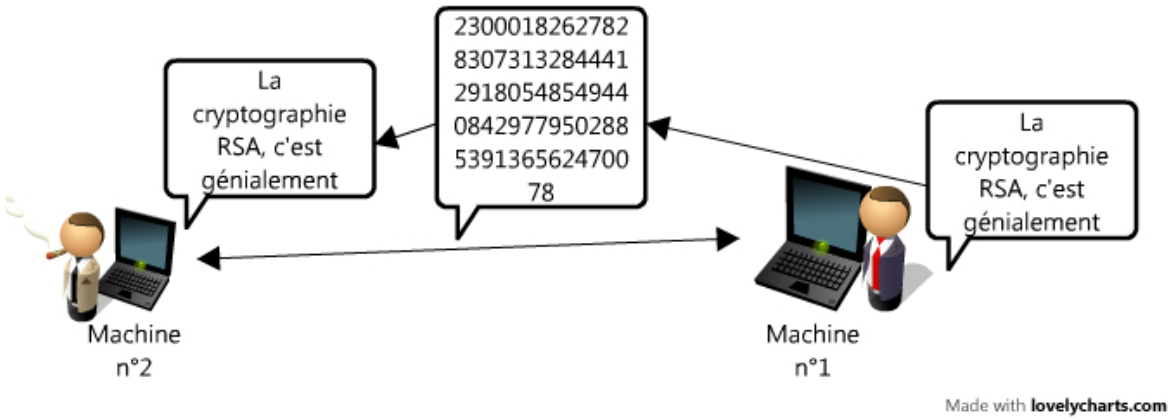
L'opération Modulo est une fonction fondamentale dans l'arithmétique modulaire en mathématique utilisé par exemple dans le RSA également très utilisé en informatique car une très grande partie des calculs réalisés en informatique sont des calculs d'arithmétique modulaire.
L'opération Modulo est tout simplement le reste d'une division euclidienne. Par exemple, en divisant 5 par 2 l'on obtient 5= 2*2+1 ceux qui correspond à 5 = 1[2].

V] Réalisation d'un programme utilisant le RSA

1] Présentation du programme.

1.1] But de notre programme.

Le but de notre programme est de faire une communication entre deux utilisateurs, avec chaque message crypté avec l'algorithme RSA, comme le montre le schéma ci-dessous.



La machine n°1 envoie un message à la machine n°2, cette dernière le reçois comme-ci rien ne s'était passé alors qu'en réalité sur le transfert du message, celui-ci a été crypté puis restitué normalement.

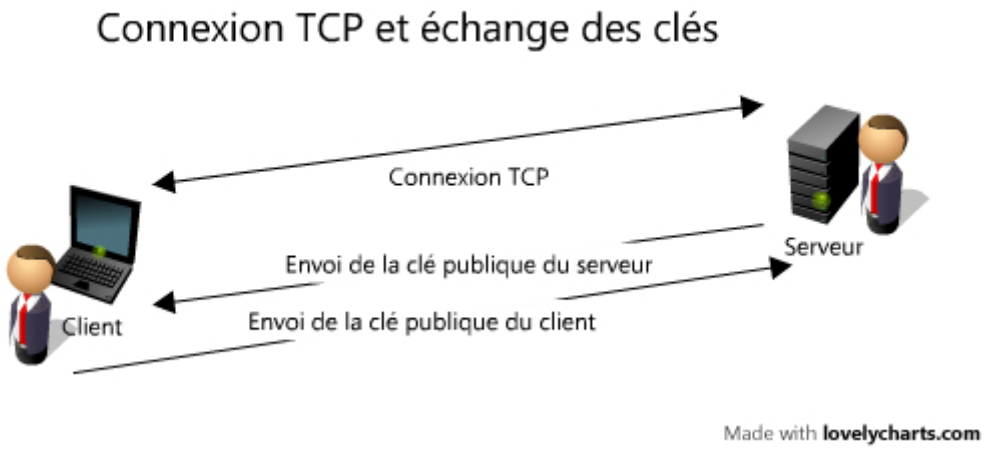
Notre programme devait répondre à plusieurs critères :

- Crypté les messages à envoyer et décrypté les messages que l'on recevait
- Permettre d'afficher les messages cryptés, en console
- Le programme doit savoir gérer les caractères non connus comme &#%$£@…
- Avoir une interface graphique simple et efficace

1.2] Choix du protocole de communication

Notre première problématique durant le développement était le choix du protocole de transfert entre l'UDP et le TCP. Effectivement, même si l'UDP offre un transfert plus rapide, celui-ci reste moins sécurisé et nos paquets peuvent se perdre. De plus le TCP établie une connexion puis échange les paquets qui non aucun chance de se perdre et/ou d'altération. Cela ressemble bien au principe d'application du RSA en général. Pour économiser des ressources de calculs, le RSA est utilisé pour établir la communication et puis un cryptage symétrique pour l'échange de paquet.

Notre programme en TCP fonctionne sur le principe d'un client-serveur. Le serveur se lance est attend une demande de connexion sur un port précis. Le client va donc demander une connexion au serveur et attendre une réponse de celui-ci. En fin de compte, une fois la connexion établis entre les 2 clients, le principe du RSA va s'appliquer, l'échange des clés va alors se faire et ensuite les 2 machines (le serveur et le client) vont pouvoir s'échanger des messages cryptés sur le réseau.



1.3] Etude des trames échangés de notre programme

Grace à Wireshark, l'on peut voir l'établissement de la connexion, l'échange des clés et des données, en voici une capture.

Notre client va demander une connexion au serveur sur le port : 1234

```
Transmission Control Protocol, Src Port: 53925 (53925), Dst Port: search-agent (1234), Seq: 0, Len: 0
    Source port: 53925 (53925)
    Destination port: search-agent (1234)
```

Le serveur lui renvoi un accusé de réception

```
Transmission Control Protocol, Src Port: search-agent (1234), Dst Port: 53925 (53925), Seq: 0, Ack: 1, Len: 0
    Source port: search-agent (1234)
    Destination port: 53925 (53925)
    [Stream index: 2]
    Sequence number: 0    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 40 bytes
  ▸ Flags: 0x12 (SYN, ACK)
    Window size: 32768
  ▸ Checksum: 0xc074 [validation disabled]
  ▸ Options: (20 bytes)
  ▸ [SEQ/ACK analysis]
```

Le serveur envoie au client sa clé publique (clé E), ici surligné en jaune.

```
Transmission Control Protocol, Src Port: search-agent (1234), Dst Port: 53925 (53925), Seq: 1, Ack: 1, Len: 79
    Source port: search-agent (1234)
    Destination port: 53925 (53925)
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 83 77 c5 40 00 40 06  c4 ad 7f 00 00 01 7f 00   ..w.@.@. ........
0020  00 01 04 d2 d2 a5 38 ea  9b 0a 38 cb 11 21 80 18   ......8. ..8..!..
0030  02 00 fe 77 00 00 01 01  08 0a 00 15 ec cf 00 15   ...w.... ........
0040  ec cf 31 39 39 30 37 39  32 33 38 30 38 34 34 32   ..199079 23808442
0050  30 36 33 34 35 30 37 30  36 30 34 33 34 35 33 39   06345070 60434539
0060  30 38 31 39 39 37 38 34  34 37 37 30 38 38 37 34   08199784 47708874
0070  39 32 31 39 37 31 37 31  31 39 32 37 39 31 33 31   92197171 19279131
0080  35 31 34 35 36 31 39 37  33 35 36 34 37 39 31 39   51456197 35647919
0090  0a                                                .
```

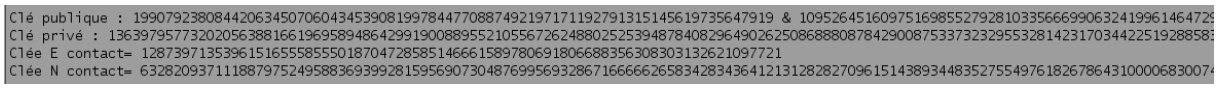
Le client fait de même

```
Transmission Control Protocol, Src Port: 53925 (53925), Dst Port: search-agent (1234), Seq: 1, Ack: 80, Len: 79
    Source port: 53925 (53925)
    Destination port: search-agent (1234)
```

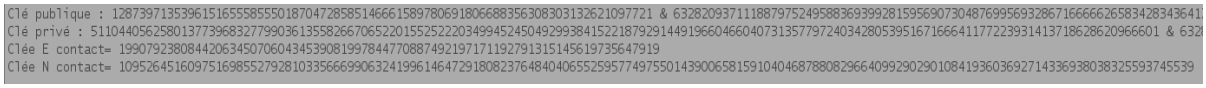
```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 83 fd f2 40 00 40 06  3e 80 7f 00 00 01 7f 00   ....@.@. >.......
0020  00 01 d2 a5 04 d2 38 cb  11 21 38 ea 9b 59 80 18   ......8. .!8..Y..
0030  02 01 fe 77 00 00 01 01  08 0a 00 15 ec d1 00 15   ...w.... ........
0040  ec cf 31 32 38 37 33 39  37 31 33 35 33 39 36 31   ..128739 71353961
0050  35 31 36 35 35 35 38 35  35 35 30 31 38 37 30 34   51655585 55018704
0060  37 32 38 35 38 35 31 34  36 36 36 31 35 38 39 37   72858514 66615897
0070  38 30 36 39 31 38 30 36  36 38 38 33 35 36 33 30   80691806 68835630
0080  38 33 30 33 31 33 32 36  32 31 30 39 37 37 32 31   83031326 21097721
0090  0a                                                .
```

Ensuite le serveur et le client font de même avec la partie N de leur clé publique. Ainsi ils ont tous les deux reçus la clé publique de l'un et de l'autre.

Maintenant allons voir ce qui se passe du coté de notre programme, notamment de la partie console.
Serveur :



Client :

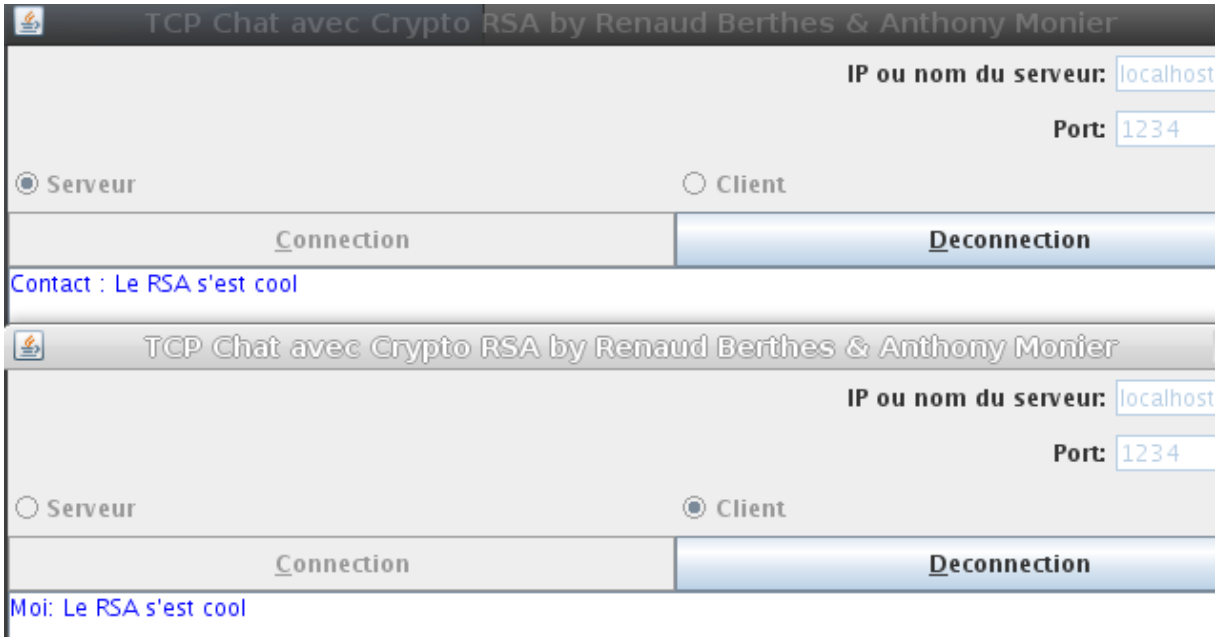


La partie console de notre programme nous permet d'avoir des informations supplémentaires, notamment les clés ainsi que les messages chiffrés reçus.

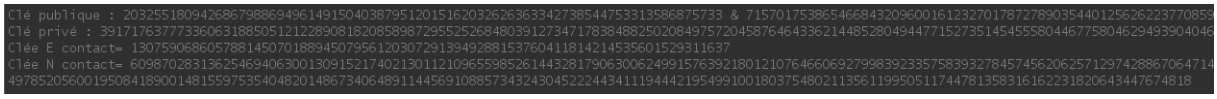
1.4] Que se passe-t-il quand l'on envoie un message ?

Maintenant que notre connexion est établie, que nos clés publiques sont échanges. Comment circule nos messages sur le réseau ?

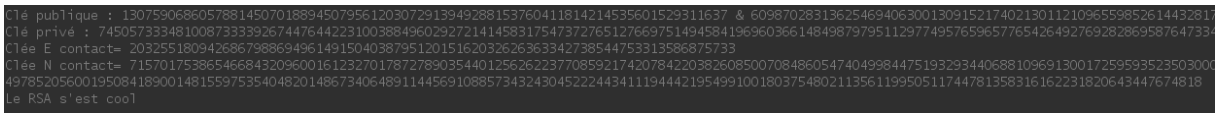
Envoyons par exemple le message : « Le RSA, c'est cool »



En dernière ligne de la console du client l'on voit notre message envoyé en crypté :



Notre serveur reçoit cette suite de chiffre, le déchiffre avec sa clé privé et affiche le message en clair.



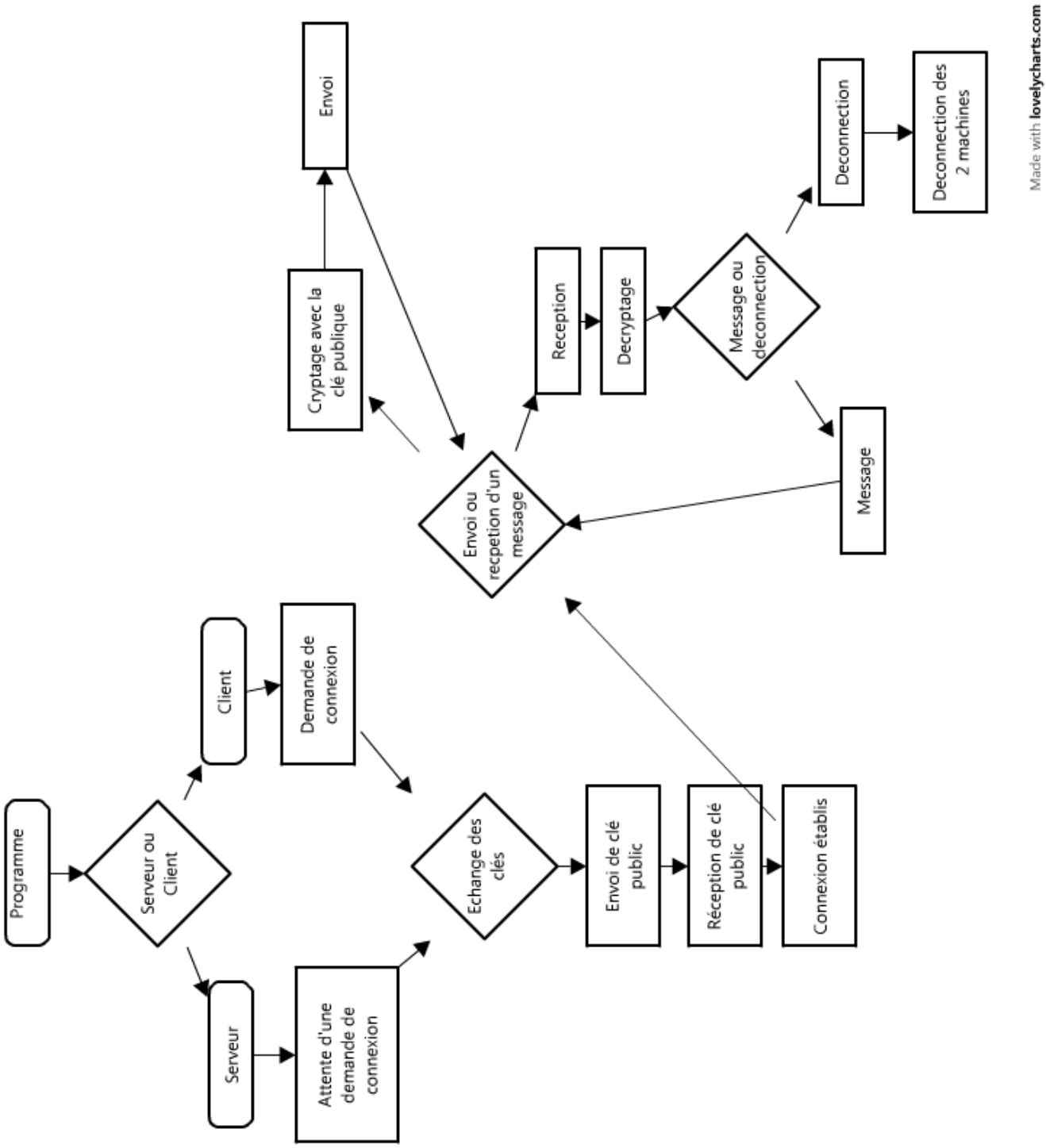

```
........ ........
..497852 05600195
08418900 14815597
53540482 01486734
06489114 45691088
57343243 04522244
34111944 42195499
10018037 54802113
56119950 51174478
13583161 62231820
64344767 4818.
```

Sur le réseau, l'on voit cette même suite de chiffre, et donc sans la clé privée du destinataire l'on ne peut déchiffrer le message.

2] Schéma du programme

Créer un schéma pour un programme nous permet de définir facilement les fonctions à mettre ne place dans notre programme.

Le fonctionnement de notre programme vous a été présenté, mais comment notre programme traite-t-il les informations ?



3] Réalisation en Java

Nous avons choisi de programmer notre programme en Java car d'une part c'est le langage que nous connaissons le mieux et d'autre part, cela permet à notre programme de tourner sur tous les types de machines (Linux, Windows, IOS, …).

En java, il existe déjà des classes pour implémenter la cryptographie RSA dans un programme mais nous avons décidé de créer nos propres classes. Disponible en annexe

3.1] Notre table ASCII

Le problème dans notre programme c'est que nous devons traiter des chiffres en RSA mais échanger des messages en caractères, pour cela nous avons dû coder notre propre table ASCII, une variable de l'ASCII connu.

Lorsqu'on envoi notre chaine de caractères, nous devons découper la chaine pour chaque caractères ensuite chaque caractère est converti en entier avec notre table ASCII. Une fois que chaque caractère est codé en entier, l'on ajoute à la suite les nombres de chaque caractère.

Ainsi une fois notre message convertit en entier, l'on pourra appliquer différents calculs dessus, notamment ceux nécessaires pour l'algorithme RSA.

Comme la table ascii donne pour certain caractère un entier sur 3 chiffres, chaque caractère de notre classe sera codé sur 2 entiers, l'on décale le codage de la table ASCII normale.
Cela évite aussi les erreurs dans le décryptage d'un message dû à un décalage dans le découpage du chiffre.

Nous avons donc vu ce que fait la classe ASCII lors de l'envoi d'un message. Mais que l'on reçoit le message, que fait-on ?

D'abord, c'est la classe RSA qui va nous retourner le long entier, qui correspond à notre chaine de caractères en ASCII, ensuite nous allons découper cet entier en groupe de 2 chiffres et transforme chaque groupe de 2 en caractères.

3.2] Notre classe RSA

Nous avons ensuite codé une classe qui va nous permettre de coder et décoder nos messages, la classe RSA.java. Cette dernière créer les clés publiques et privés et gère aussi le cryptage et décryptages des messages.
Or un autre problème rencontrée en java, est que de travailler sur de grand nombre entier est impossible avec la classe int, ni même avec la classe long.

Nous avons donc découvert la classe BigInteger et une fonction de celle-ci vraiment très intéressant qui est « probablePrime » qui permet de générer un BigInteger très grand PROBABLEMENT premier.
Création d'un entier probablement premier d'une taille définie et aléatoire :

```
BigInteger p = BigInteger.probablePrime(tailleCle, new Random());//Nombre Premier de taille 512 p
```

On test si p et q sont premiers entre eux.

```
while ((!this.e.gcd(w).toString().equals("1")) || (this.e.compareTo(w) != -1)) { // e premier avec w et e < w
            e = new BigInteger(tailleCle + 1, new Random());
```

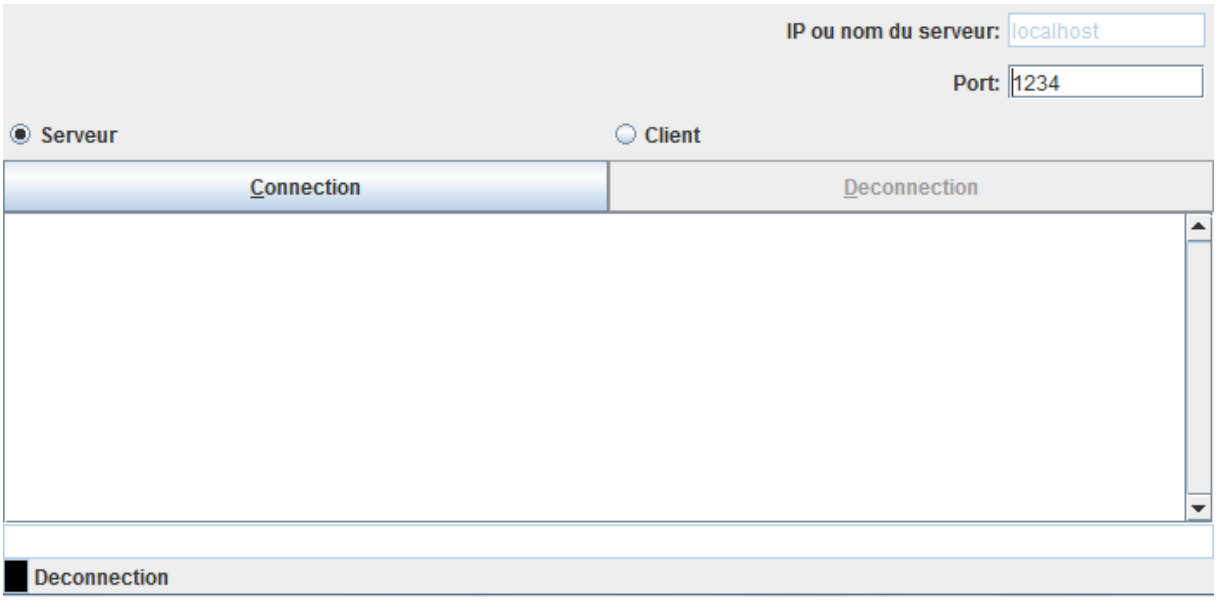
De plus la classe BigInteger intègre toutes les fonctions de calculs nécessaires pour créer l'algorithme RSA.
Par exemple pour le cryptage des données :

```
public static BigInteger cryptage2(BigInteger message, BigInteger e2, BigInteger n2) {
        return message.modPow(e2, n2); // On crypte notre message : message(Puissance : clé public) modulo(n = p*q)
    }
```

Une fois que l'on a ce qu'il faut pour transformer une chaine de caractère en chiffre et chiffrer nos messages, il ne nous restait plus qu'à utiliser cela dans un programme graphique.

3.3] Notre programme TCPchat

Notre programme doit gérer la réception et l'émission de message, le tout dans une interface simple.



Le programme nous permet d'être soit client soit serveur, l'on gère le IP et le port du serveur à utiliser.
Pour éviter que notre programme freeze, nous avons mis en place des threads pour gérer indépendamment chaque fonction de notre programme.

De plus, si l'une des deux machines se déconnectent, l'autre en fait de même automatiquement.

VI] La sécurité du RSA

- Attaque par Factorisation

L'attaque par factorisation consiste à essayer de factoriser le module N et donc comme objectif de retrouve p et q. Avec cela, l'on pourra calculer d pour déchiffrer les messages.

La société RSA Security qui a été créée pour vendre le cryptosystèmes RSA a créé en 1991 un concours de factorisation. Les concours ont été arrêter en 2007 mais certain chercheurs essaient encore de factoriser des clés de toutes tailles.

En janvier 2010 une clé RSA de 768 bits à était cracker par factorisation, cela a coûter 2 ans et demi de recherche et de calculs, 5To de données traitées, des calculs distribués sur plus de 1700 cœurs, environ 10^{20} opérations.

VII] Conclusion

Ce projet nous a permis aussi, au final, de nous distinguer dans différents domaines d'enrichissement personnel.

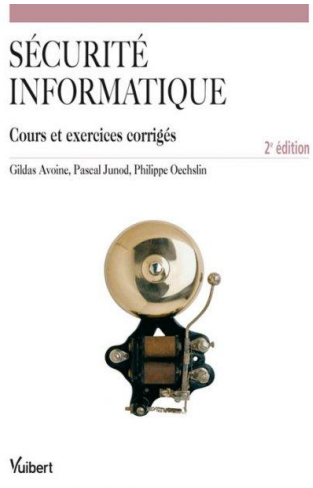
D'une part, dans la gestion d'un projet, d'ailleurs le début du projet est la partie la plus difficile à réaliser, car à ce stade, l'on n'a aucune idée de ce que l'on doit faire. La période de création du phasing nous permet de découper notre projet en partie facilement réalisable et le tout sur une période. En effet, il faut apprendre à organiser les connaissances et le travail. Ainsi nous avons pu gérer sur le temps et sur le travail, notre projet et avec même de l'avance à la fin.

Dans un plan technique, en mathématique, nous avons fait l'étude d'un algorithme, en quoi celui-ci peut-il sécuriser une communication … Nous avons énormément progressé dans la programmation en Java, grâce et à cause des difficultés rencontrés dans notre développement.

VIII] Sources

Livre :

Sécurité informatique – Cours et exercices corrigés – Edition Vuibert



Site Web :

http://cryptage.sectionpc.info/crypto/4.htm
http://fr.wikipedia.org/wiki/Rivest_Shamir_Adleman

http://my.lovelycharts.com/ pour la création des diagrammes

X] Annexes

Code source : RSA.java

```
import java.math.BigInteger;
import java.util.Random;

public class RSA {

   private BigInteger n, e, d; // (e,n) Clée Public, (d,n) Clée Privée.

   public RSA() {
      int tailleCle = 256;
      BigInteger p = BigInteger.probablePrime(tailleCle, new Random());//Nombre Premier de taille 512 p ( Attention ils sont PROBLABLEMENT
premiers ! )
      BigInteger q = BigInteger.probablePrime(tailleCle, new Random());//Nombre Premier de taille 512 p
      n = p.multiply(q); //n = p*q

      BigInteger pbis = p.subtract(new BigInteger("1")); //p-1
      BigInteger qbis = q.subtract(new BigInteger("1")); //q-1
      BigInteger w = pbis.multiply(qbis); //(p-1).(q-1)

      this.e = new BigInteger(tailleCle + 1, new Random()); //e>p,q
      // A essayer les premiers entre eux, pas pour le moment disponible
      while ((!this.e.gcd(w).toString().equals("1")) || (this.e.compareTo(w) != -1)) { // e premier avec w et e < w
         e = new BigInteger(tailleCle + 1, new Random()); //e>p,q ; Possibilité de prendre la valeur de 2^16 + 1 (35537)  qui est une valeur commune
voir ligne suivante
         //e =  new BigInteger("65537");
      }

      d = e.modInverse(w); //Calcul du modulo inverse pour avoir la Clée Pivée.

      // On affiche les infos de cryptage  :
      //System.out.println(" P : " + p);
      //System.out.println(" Q : " + q);
   }

   BigInteger cryptage(BigInteger message) {
      return message.modPow(e, n); // On crypte notre message : message(Puissance : clé public) modulo(n = p*q)
   }
   public static BigInteger cryptage2(BigInteger message, BigInteger e2, BigInteger n2) {
      return message.modPow(e2, n2); // On crypte notre message : message(Puissance : clé public) modulo(n = p*q)
   }
   BigInteger decryptage(BigInteger messageCrypté) {
      return messageCrypté.modPow(d, n); // décrypte notre message : crypté(puissance: clé privé, modulo (n=p*q))
   }
   public static BigInteger decryptage2(BigInteger messageCrypté, BigInteger d2, BigInteger n2) {
      return messageCrypté.modPow(d2, n2); // décrypte notre message : crypté(puissance: clé privé, modulo (n=p*q))
   }

   public BigInteger getN() {
      return n;
   }

   public void setN(BigInteger n) {
      this.n = n;
   }

   public BigInteger getE() {
      return e;
   }

   public void setE(BigInteger e) {
      this.e = e;
   }

   public BigInteger getD() {
      return d;
   }

   public void setD(BigInteger d) {
      this.d = d;
   }
}
```

Code source : ASCII.java

```
import java.math.BigInteger;

public class ASCII {
        public static int charToInt(char c){

                //Convertion d'un caractere en Entiers
            //Bidouillage de la table ascii pour avoir chaque caracteres sur 2 chiffres
        int m=0;
    if((c >= '%' && c <= '_')){
      m=c; //Ici on convertie direct avec la methode int=char qui donne le code Ascii.
    }else if(c >= 'a' && c <= 'z'){
      m=c-87; //On decale les minuscules car elle sont sur 3 chiffres pour convertir.
    }//Si un caractere autre que a-z, A-Z, 0-9, m=0 est sera remplacer par un espace dans intToChar
    return m;
  }
        //Convertion d'un numero en Char
  public static char intToChar(int n){
    if((n>=37 && n<=95)){
      return (char)n;
    }else if(n>=10 && n<=35){
      n=n+87; //On redecale pour refaire correspondre les minuscules.
      return (char)(n);
    }else{
      return ' ';//Si c'est un caracteres Special on met un Espace. Sache que l'espace est un Caractere Special.
    }

  }
  //Convertion d'une chaine de caracterre en BigInteger grace a charToInt
  public static BigInteger stringToInt(String ch){
    String sh="1";
    for(int i=0; i<ch.length(); i++){
      int p = charToInt(ch.charAt(i));
      if(p<10){
        sh=sh+"0"+p;
      }else{
        sh=sh+p;
      }
    }
    return new BigInteger(sh);
  }
  //Converstion d'un Biginteger en String grace a intToChar
  public static String intToString(BigInteger bg){
    String ch1 = bg.toString();
    String ch = "";
    for(int i=1; i<ch1.length(); i++){
      ch=ch+ch1.charAt(i);
    }
    String sh="";
    for(int i=0; i<ch.length(); i=i+2){
      char p = ch.charAt(i);
      char p2 = ch.charAt(i+1);
      String tmp = ""+p+p2;
      int m = Integer.parseInt(tmp);
      char t = intToChar(m);
      sh=sh+t;
    }
    return sh;
  }

}
```

Code source : TCPChat.java

```
/*
 * Le chat crée un serveur TCP
 * OU
 * les clients se connectent au serveur
* Les phrases de plus de 50 caractères sont coupées
 * Ne gère pas les pseudos
 *
 *
 *
 * Les adresses IP MARCHENT pour joindre le serveur
 * IUT de b��thune
 */
package rsa;

/**
 *
 * @author jake
 */
import java.lang.*;
import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.math.BigInteger;
import javax.swing.*;
import java.net.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


////////////////////////////////////////////////////////////////
// Action adapter for easy event-listener coding
public class TCPChat implements Runnable {
    // Statue de Connexion

    public final static int NULL = 0;
    public final static int DISCONNECTED = 1;
    public final static int DISCONNECTING = 2;
    public final static int BEGIN_CONNECT = 3;
    public final static int CONNECTED = 4;
    //RSA
    // Notre RSA
    public static RSA test;
    public static BigInteger eRSA;
    public static BigInteger nRSA;
    public static BigInteger dRSA;
    // RSA contact
    public static BigInteger eRSA2;
    public static BigInteger nRSA2;
    public static BigInteger dRSA2;
    // Autre constante
    public final static String statusMessages[] = {
        " Impossible de se connecter", " Deconnection",
        " Deconnection...", " Connection...", " Connectée"
    };
    public final static TCPChat tcpObj = new TCPChat();
    public final static String END_CHAT_SESSION =
            new Character((char) 0).toString(); // Indique la fin d'une connection
    // Info de connection
    public static String hostIP = "localhost";
    public static int port = 1234;
    public static int connectionStatus = DISCONNECTED;
    public static boolean isHost = true;
    public static String statusString = statusMessages[connectionStatus];
    public static StringBuffer toAppend = new StringBuffer("");
```

```
    public static StringBuffer toSend = new StringBuffer("");
    // Various GUI components and info
    public static JFrame mainFrame = null;
    public static JTextArea chatText = null;
    public static JTextField chatLine = null;
    public static JPanel statusBar = null;
    public static JLabel statusField = null;
    public static JTextField statusColor = null;
    public static JTextField ipField = null;
    public static JTextField portField = null;
    public static JRadioButton hostOption = null;
    public static JRadioButton guestOption = null;
    public static JButton connectButton = null;
    public static JButton disconnectButton = null;
    //Info TCP
    public static ServerSocket hostServer = null;
    public static Socket socket = null;
    public static BufferedReader in = null;
    public static PrintWriter out = null;

    ////////////////////////////////////////////////////////////////
    private static JPanel initOptionsPane() {
        JPanel pane = null;
        ActionAdapter buttonListener = null;

        // Create an options pane
        JPanel optionsPane = new JPanel(new GridLayout(4, 1));
        // IP address input
        pane = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        pane.add(new JLabel("IP ou nom du serveur:"));
        ipField = new JTextField(10);
        ipField.setText(hostIP);
        ipField.setEnabled(false);
        ipField.addFocusListener(new FocusAdapter() {

            public void focusLost(FocusEvent e) {
                ipField.selectAll();
                // Should be editable only when disconnected
                if (connectionStatus != DISCONNECTED) {
                    changeStatusNTS(NULL, true);
                } else {
                    hostIP = ipField.getText();
                }
            }
            });
        pane.add(ipField);

        optionsPane.add(pane);

        // Port input
        pane = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        pane.add(new JLabel("Port:"));
        portField = new JTextField(10);
        portField.setEditable(true);
        portField.setText((new Integer(port)).toString());
        portField.addFocusListener(new FocusAdapter() {

            public void focusLost(FocusEvent e) {
                // should be editable only when disconnected
                if (connectionStatus != DISCONNECTED) {
                    changeStatusNTS(NULL, true);
                } else {
                    int temp;
                    try {
                        temp = Integer.parseInt(portField.getText());
                        port = temp;
                    } catch (NumberFormatException nfe) {
```

```
                    portField.setText((new Integer(port)).toString());
                    mainFrame.repaint();
                }
            }
        }
    });
    pane.add(portField);
    optionsPane.add(pane);

    // Option pour savoir si on est host => Serveur ou guest => Client
    buttonListener = new ActionAdapter() {

        public void actionPerformed(ActionEvent e) {
            if (connectionStatus != DISCONNECTED) {
                changeStatusNTS(NULL, true);
            } else {
                isHost = e.getActionCommand().equals("host");

                // Impossible d'��tablir la conecion ip ou non ?
                if (isHost) {
                    ipField.setEnabled(false);
                    ipField.setText("localhost");
                    hostIP = "localhost";
                } else {
                    ipField.setEnabled(true);
                }
            }
             }
    };
    ButtonGroup bg = new ButtonGroup();
    hostOption = new JRadioButton("Serveur", true);
    hostOption.setMnemonic(KeyEvent.VK_H);
    hostOption.setActionCommand("host");
    hostOption.addActionListener(buttonListener);
    guestOption = new JRadioButton("Client", false);
    guestOption.setMnemonic(KeyEvent.VK_G);
    guestOption.setActionCommand("guest");
    guestOption.addActionListener(buttonListener);
    bg.add(hostOption);
    bg.add(guestOption);
    pane = new JPanel(new GridLayout(1, 2));
    pane.add(hostOption);
    pane.add(guestOption);
    optionsPane.add(pane);

    // Connect/disconnect buttons
    JPanel buttonPane = new JPanel(new GridLayout(1, 2));
    buttonListener = new ActionAdapter() {

        public void actionPerformed(ActionEvent e) {
            // Request a connection initiation
            if (e.getActionCommand().equals("connect")) {
                changeStatusNTS(BEGIN_CONNECT, true);
            } // Disconnect
            else {
                changeStatusNTS(DISCONNECTING, true);
            }
        }
    };
    connectButton = new JButton("Connection");
    connectButton.setMnemonic(KeyEvent.VK_C);
    connectButton.setActionCommand("connect");
    connectButton.addActionListener(buttonListener);
    connectButton.setEnabled(true);
    disconnectButton = new JButton("Deconnection");
    disconnectButton.setMnemonic(KeyEvent.VK_D);
    disconnectButton.setActionCommand("disconnect");
```

```
        disconnectButton.addActionListener(buttonListener);
        disconnectButton.setEnabled(false);
        buttonPane.add(connectButton);
        buttonPane.add(disconnectButton);
        optionsPane.add(buttonPane);

        return optionsPane;
    }

    ////////////////////////////////////////////////////////////////

    private static void initGUI() {
        // Set up the status bar
        statusField = new JLabel();
        statusField.setText(statusMessages[DISCONNECTED]);
        statusColor = new JTextField(1);
        statusColor.setBackground(Color.BLACK);
        statusColor.setEditable(false);
        statusBar = new JPanel(new BorderLayout());
        statusBar.add(statusColor, BorderLayout.WEST);
        statusBar.add(statusField, BorderLayout.CENTER);

        // Set up the options pane
        JPanel optionsPane = initOptionsPane();

        // Set up the chat pane
        JPanel chatPane = new JPanel(new BorderLayout());
        chatText = new JTextArea(10, 20);
        chatText.setLineWrap(true);
        chatText.setEditable(false);
        chatText.setForeground(Color.blue);
        JScrollPane chatTextPane = new JScrollPane(chatText,
                JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
                JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
        chatLine = new JTextField();
        chatLine.setEnabled(false);
        chatLine.addActionListener(new ActionAdapter() {
            // Quand on appuis sur Retour Chariot, on envoie un message

            @Override
            public void actionPerformed(ActionEvent e) {
                String s = chatLine.getText();
                if (!s.equals("")) {
                    if (s.length() <= 50) {
                        appendToChatBox("Moi: " + s + "¥n");
                        chatLine.selectAll();
                        BigInteger mess2 = ASCII.stringToInt(s);
                        BigInteger ma = test.cryptage2(mess2, eRSA2, nRSA2);
                        String msg = "" + ma;

                                                sendString(msg);
                    } else { // ici on tronque les chaines de + de 50 charactéres, à régler pour recup la suite
                        Pattern pattern = Pattern.compile(".{50}");
                        Matcher matcher = pattern.matcher(s);
                        while (matcher.find()) {
                            String texte = matcher.group();
                            appendToChatBox("Moi: " + texte + "¥n");
                            chatLine.selectAll();
                            BigInteger mess2 = ASCII.stringToInt(texte);
                            BigInteger ma = test.cryptage2(mess2, eRSA2, nRSA2);
                            String msg = "" + ma;
                            sendString(msg);
                        }
                    }

                }
            }
```

```
        });
        chatPane.add(chatLine, BorderLayout.SOUTH);
        chatPane.add(chatTextPane, BorderLayout.CENTER);
        chatPane.setPreferredSize(new Dimension(700, 200));

        JPanel mainPane = new JPanel(new BorderLayout());
        mainPane.add(optionsPane, BorderLayout.NORTH);
        mainPane.add(statusBar, BorderLayout.SOUTH);
        //  mainPane.add(optionsPane, BorderLayout.WEST);
        mainPane.add(chatPane, BorderLayout.CENTER);

        mainFrame = new JFrame("TCP Chat avec Crypto RSA by Renaud Berthes & Anthony Monier");
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setContentPane(mainPane);
        mainFrame.setSize(mainFrame.getPreferredSize());
        mainFrame.setLocation(200, 200);
        mainFrame.pack();
        mainFrame.setVisible(true);
    }

    private static void changeStatusTS(int newConnectStatus, boolean noError) {
        // Change state if valid state
        if (newConnectStatus != NULL) {
            connectionStatus = newConnectStatus;
        }

        if (noError) {
            statusString = statusMessages[connectionStatus];
        } // Otherwise, display error message
        else {
            statusString = statusMessages[NULL];
        }

        // Call the run() routine (Runnable interface) on the
        // error-handling and GUI-update thread
        SwingUtilities.invokeLater(tcpObj);
    }

    private static void changeStatusNTS(int newConnectStatus, boolean noError) {
        // Change state if valid state
        if (newConnectStatus != NULL) {
            connectionStatus = newConnectStatus;
        }

        if (noError) {
            statusString = statusMessages[connectionStatus];
        } // Otherwise, display error message
        else {
            statusString = statusMessages[NULL];
        }
        tcpObj.run();
    }

    private static void appendToChatBox(String s) {
        synchronized (toAppend) {
            toAppend.append(s);
        }
    }

    private static void sendString(String s) {
        synchronized (toSend) {
            toSend.append(s + "¥n");
        }
    }

    private static void cleanUp() {
        try {
```

```
            if (hostServer != null) {
                hostServer.close();
                hostServer = null;
            }
        } catch (IOException e) {
            hostServer = null;
        }

try {
            if (socket != null) {
                socket.close();
                socket = null;
            }
        } catch (IOException e) {
            socket = null;
        }

        try {
            if (in != null) {
                in.close();
                in = null;
            }
        } catch (IOException e) {
            in = null;
        }

        if (out != null) {
            out.close();
            out = null;
        }
    }
    public void run() {
        switch (connectionStatus) {
            case DISCONNECTED:
                connectButton.setEnabled(true);
                disconnectButton.setEnabled(false);
                ipField.setEnabled(true);
                portField.setEnabled(true);
                hostOption.setEnabled(true);
                guestOption.setEnabled(true);
                chatLine.setText("");
                chatLine.setEnabled(false);
                statusColor.setBackground(Color.black);
                break;

            case DISCONNECTING:
                connectButton.setEnabled(false);
                disconnectButton.setEnabled(false);
                ipField.setEnabled(false);
                portField.setEnabled(false);
                hostOption.setEnabled(false);
                guestOption.setEnabled(false);
                chatLine.setEnabled(false);
                statusColor.setBackground(Color.GRAY);
                break;

            case CONNECTED:
                connectButton.setEnabled(false);
                disconnectButton.setEnabled(true);
                ipField.setEnabled(false);
                portField.setEnabled(false);
                hostOption.setEnabled(false);
                guestOption.setEnabled(false);
                chatLine.setEnabled(true);
                statusColor.setBackground(Color.green);
                break;
```

```
            case BEGIN_CONNECT:
                connectButton.setEnabled(false);
                disconnectButton.setEnabled(false);
                ipField.setEnabled(false);
                portField.setEnabled(false);
                hostOption.setEnabled(false);
                guestOption.setEnabled(false);
                chatLine.setEnabled(false);
                chatLine.grabFocus();
                statusColor.setBackground(Color.blue);
                break;
                }
        ipField.setText(hostIP);
        portField.setText((new Integer(port)).toString());
        hostOption.setSelected(isHost);
        guestOption.setSelected(!isHost);
        statusField.setText(statusString);
        chatText.append(toAppend.toString());
        toAppend.setLength(0);

        mainFrame.repaint();
    }

    ////////////////////////////////////////////////////////////
    // Fonction principale
    public static void main(String args[]) {
        String s;
        test = new RSA();
        eRSA = test.getE();
        nRSA = test.getN();
        dRSA = test.getD();
        System.out.println("Clé publique : " + eRSA + " & " + nRSA);
        System.out.println("Clé privé : " + dRSA + " & " + nRSA);

        initGUI();

        while (true) {
            try { // On lance le thread toute les 5secondes
                Thread.sleep(5);
            } catch (InterruptedException e) {
            }

            switch (connectionStatus) {
                case BEGIN_CONNECT:
                    try {
                        // Si on est l'host
                        if (isHost) {
                            hostServer = new ServerSocket(port);
                            socket = hostServer.accept();
                        } // Si l'on est le client
                        else {
                            socket = new Socket(hostIP, port);
                        }

                        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                        out = new PrintWriter(socket.getOutputStream(), true);
                        out.println(eRSA);
                        String getRSAe = in.readLine();
                        out.println(nRSA);
                        String getRSAn = in.readLine();
                        eRSA2 = new BigInteger(getRSAe);
                        nRSA2 = new BigInteger(getRSAn);
                        System.out.println("Clée E contact= " + getRSAe);
                        System.out.println("Clée N contact= " + getRSAn);
                        changeStatusTS(CONNECTED, true);
                    } // Si erreur
                    catch (IOException e) {
```

```
                    cleanUp();
                    changeStatusTS(DISCONNECTED, false);
                }
                break;

            case CONNECTED:
                try {
                    // Envoi des donn??es
                    if (toSend.length() != 0) {
                        System.out.println(toSend);
                        out.print(toSend);
                        out.flush();
                        toSend.setLength(0);
                        changeStatusTS(NULL, true);
                    }
                    // Reception des donn??es
                    if (in.ready()) {
                        s = in.readLine();
                        if ((s != null) && (s.length() != 0)) {
                            // On test si c'est la  fin de la transmission
                            System.out.println(s);
                            String sh = "";
                            if (!s.equals("
```