Table des matière

Table des figures

Table des tableaux

Introduction générale

Le peer-to-peer a été très répondu dans le milieu des développeurs récents après sa révolution qui a permis de résoudre certains problèmes d'architecture client / serveur. Bien que le P2P peut libérer et distribuer un montant impressionnant de puissance de calcul, de stockage et de bande passante, il reste limité sur la langue Java et ne permet pas l'interopérabilité.

De toute évidence, nous avions besoin d'une technologie plus puissante, la solution a été apportée par Le projet JXTA, une initiative lancée par Sun depuis Avril 2001 pour standardiser les applications P2P qui manquent de normes et de soutien industriels.

JXTA est une plate-forme informatique réseau ouverte conçue pour l'architecture Peer-to-Peer (P2P) en fournissant les éléments de base et les services nécessaires pour permettre tout ce qui concerne la connectivité de l'application. Le JXTA fournit un ensemble commun de protocoles ouverts avec des implémentations de référence open source pour développer des applications Peer-to-Peer.

Cependant Le besoin de dissimuler les informations préoccupe l'homme depuis le début de la civilisation, parmi les problèmes que l'on s'attaque à résoudre de nos jours, on retrouve bien entendu celui du chiffrement des messages et des données transitant entre deux interlocuteurs ou plusieurs, afin d'en garantir la confidentialité mais aussi garantir leur intégrité et leur authenticité, ceci par le biais de la cryptographie.

L'objectif principal de notre projet de fin d'étude est la sécurisation des échanges des données dans un réseau JXTA, il consiste à mettre en œuvre une application P2P sous la plateforme JXTA qui va par la suite chiffrer les messages et assurer la confidentialité grâce à la cryptographie symétrique.

Ce rapport est organisé en quatre chapitres. Chaque chapitre aborde des points spécifiques. Il est structuré comme suit :

Le premier chapitre présente une introduction aux réseaux peer-to-peer, et les différents concepts liés à ces réseaux ainsi que les différents algorithmes mis en œuvre dans différents types

architectures. Le deuxième chapitre expose les concepts fondamentaux de la cryptographie moderne comme ils sont définis par les standards internationaux de la sécurité. Le troisième chapitre comporte une description de la plateforme JXTA et ses objectifs, ses différents concepts, son architecture, les couches de communication, et les six protocoles spécifique à cette plateforme. Dans le dernier chapitre, on présente notre application avec ses différents étapes de fonctionnement et les outils utilisés pour la réalisation. Enfin, nous terminons le mémoire par une conclusion générale dans laquelle nous résumons l'essentiel de notre travail.

Chapitre1
Introduction au réseau P2P

Introduction

L'internet joue un rôle fondamental à la fois dans le domaine économique et social, il est toujours ouvert au progrès afin d'exploiter ses ressources au maximum. Parmi celles-ci, qui sont souvent utilisable mais mal-exploitées, on distingue trois types : bande passante, espace de stockage et capacité de traitement. Contrairement au modèle pair-a-pair, le modèle client-serveur classique ne parvient pas à tirer pleinement profit des ressources du réseau.

Les machines composent une partie essentielle du réseau internet, appelées pairs (peers), ils forment en quelque sorte la périphérie d'internet, ils sont dotés de diverses facultés non utilisées ; puissance de calcul, espace disque… etc. ce qui a induit l'évolution du modèle peer-to-peer à tirer profit de ces capacités gâchés par le modèle précédent d'internet. Parmi les systèmes apparus on cite le tout premier Napster, qui permet le partage des données entre les utilisateurs.

Pour mieux expliquer tous cela nous aborderons dans ce chapitre une comparaison entre le client/serveur et le peer-to-peer tout en définissant chacun des deux modèles, on verra par la suite les différents algorithmes mis en œuvre dans les différents type de réseau peer-to-peer ainsi que la distinction de trois sous-modèles, appelés sous-modèles pur, hybride et centralisé.

1. L'architecture client-serveur

Dans le modèle client-serveur, toutes communications entre clients se doit de passer par le serveur. En d'autres termes, les seules ressources publiées et consommées sont celles situées sur le serveur. On cite l'exemple du service de messagerie mail.

Dans ce modèle le fonctionnement se fait de la manière suivante :[1]

1. Le client émet une requête vers le serveur grâce à son adresse IP et son Port.

2. Le serveur à son tours reçoit la demande et renvoi une réponse au client à l'aide de son adresse IP et son Port.

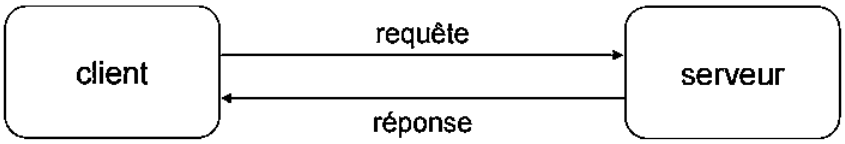


Figure 1. Arhitecture client / serveur

1.1. Avantages de l'architecture client-serveur

Le modèle client-serveur est notamment plus adapté aux réseaux sur le plan de la fiabilité, ses principaux atouts sont :

- **Des ressources centralisées** : sachant que le serveur est au centre du réseau, la gestion des ressources communes entre utilisateurs ne pose aucun souci, tel que la redondance et la contradiction, exemple une base de donnée centralisé.

- **Une meilleure sécurité** : simplement parce que les points d'entrées permettant l'accès aux données ne sont pas nombreux.

- **Une administration au niveau serveur** : vu que les clients ont moins d'importance dans ce modèle, leur administration n'est pas une nécessité.

- **Un réseau évolutif** : il est possible de mettre à jour les clients sans affecter le fonctionnement du réseau et sans modification majeure grâce à cette architecture.

1.2. Inconvénients de l'architecture client-serveur

L'architecture client-serveur a tout de même quelques lacunes parmi lesquelles :

- **Un coût élevé** dû à la technicité du serveur

- **Un maillon faible** : étant donné que tout le réseau est architecturé autour du serveur il est considéré comme le maillon faible du réseau (non tolérant au panne).

2. L'architecture Peer-to-Peer

Parmi les recherches réalisées pour entamer ce mémoire, différentes approches ont été trouvées pour définir un réseau P2P. Voici quelques définitions parmi beaucoup d'autres :

« *Le système P2P ou "égal à égal" est un système dans lequel les nœuds du réseau ont les mêmes rôles. Chaque nœud est à la fois client et serveur. Les systèmes P2P permettent la décentralisation, le partage de l'ensemble des ressources du réseau P2P, la communication et collaboration des nœuds de manière directe. Dit autrement, les systèmes P2P permettent l'exploitation par un nœud de l'ensemble des ressources du réseau .* »[2]

« *Le terme "Peer-to-Peer", "poste à poste" ou "pair à pair" en français (ou plus couramment P2P), désigne un modèle de réseau informatique dont les éléments (les nœuds ou "peer") sont à la fois clients et serveurs lors des échanges. Grâce à ce modèle on a pu décentraliser les réseaux, en opposition aux architectures traditionnelles client/serveur. Ce modèle a pu offrir la possibilité aux pairs d'avoir des communications directes, d'égal à égal comme son nom indique, entre les différents nœuds du réseau, qui peuvent alors échanger différents types d'informations sans passer par un serveur central. »*.[3]

Peer-to-Peer Networking est surtout connu sous la marque de Napster. Dans cette application, le concept de réseau Peer-to-Peer est utilisé pour partager des fichiers, à savoir l'échange de fichiers audio compressés MPEG Layer3 (mp3). Cependant, Peer-to-Peer ne concerne pas seulement le partage de fichiers, mais aussi l'établissement de réseaux de communication multimédia basés sur des concepts Peer-to-Peer ou le partage de ressources.[4]

2.1. Les avantages du P2P

- Optimisation de l'utilisation de la bande passante du réseau (équilibrage de la charge du réseau)
- Maintenance et coûts réduits (ressources réparties)
- Tolérance aux pannes (réplication des ressources) : Si une machine tombe en panne, cela ne remet pas en cause l'ensemble du système.
- Extensibilité

- Les communications sont directes : un nœud peut accéder directement à un ou plusieurs nœuds.
- Décentralisation
- Passage à l'échelle
- La réplication, redondance des données

2.2. Les inconvénients du P2P

- problèmes de sécurité
  - Virus, backdoors (spyware)
  - Distributed Denial of Service (DDoS)
  - Confidentialité, Authentification
- Les temps de localisation sont plus long
- Recherche peut être Non Déterministe
- Le freeloading
- Atteinte à la vie privée

3. Comparaison entre Client /serveur et P2P

Client/Serveur	***P2P***
Non Tolérant aux pannes	Tolérant aux pannes
Maintenance a Coût élevé	Maintenance a Coût réduit
Management centralisé	Auto-organisé
Configuration statique	Evolution dynamique
Consultation de tables	Découverte des peers
Flux centralisé	Flux distribué
Asymétrie du réseau	Symétrie du réseau
Adressage statique	Adressage dynamique
Nœuds dépendants	Nœuds autonomes
Attaques plus simples	Attaques difficiles

Tableau 1 : Comparaison entre le modèle Client-Serveur et le modèle Peer-to-Peer

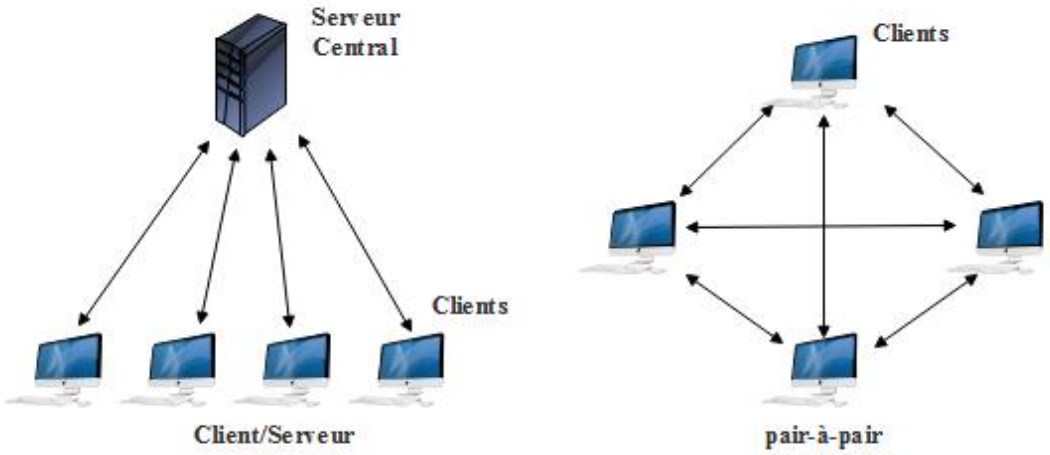


Figure 2. P2P vs Client/serveur

4. Aperçu historique

en 1969, le réseau ARPANET (**Advanced Research Projects Agency**) met en œuvre un réseau d'interconnexion non centralisé qui s'appuie sur la commutation de paquet et les protocoles TCP/IP publier en 1974 [5], sont devenues les bases techniques de l'internet et la norme de tous les réseaux informatique à partir des années 80.[6]

Nul protocole ni application P2P n'ont vu le jour, jusqu'à 1999, ou Napster a été la première application peer-to-peer pour les usagers conçue et développée par Shawn Fanning, dans un premier temps il avait pour but de seulement échanger et chercher des fichiers entre internautes sans avoir à passer par les moteurs de recherche.

Des faits de piratage entraineront la fermeture de Napster durant l'été 2001, soit deux ans après sa création. Cette disparition donnera naissance a des applications peer-to-peer nouvelles: AudioGalaxy, eDonkey, Gnutella, iMesh ou encore Kazaa. Qui connaissent par la suite un succès immédiat (plus de 3 milliards de fichiers téléchargés durant l'été 2001).

L'année 2001 verra également l'apparition des protocoles BitTorrent et WinMX. L'apport principal de ces nouvelles applications réside dans leur mode 100% Peer-to-Peer, aucun serveur central n'est utilisé pour l'établissement des connexions entre utilisateurs. Seuls quelques serveurs dits annuaires sont utilisés pour initialiser les applications avec une première liste d'ordinateurs peers.

C'est ainsi que le Peer-to-Peer fut connu et émergé dans l'univers du partage de fichier .[4]

5. Type de réseaux Peer-to-Peer

Les approches dans le domaine P2P reposent sur la construction d'un réseau virtuel sur le réseau physique (typiquement Internet). Il existe deux catégories principales de réseaux virtuels: les non-structurés, les structurés.

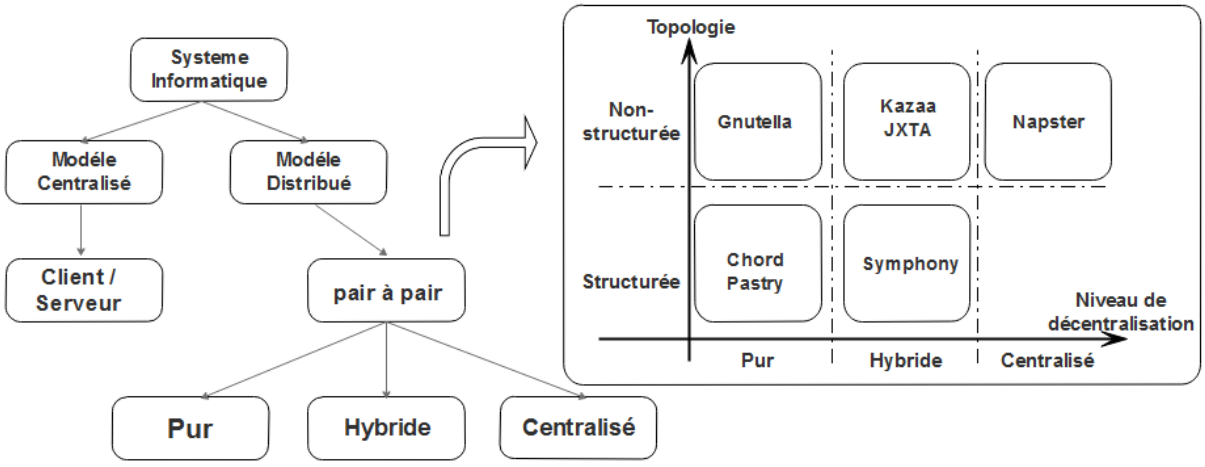


Figure 3. Classification des systèmes informatiques et des applications P2P

5.1. Peer-to-Peer non structurés

Un réseau peer-to-peer est non structuré quand les liens entre les nœuds sont établis de façon aléatoire. La communication entre les nœuds est donc plus difficile à gérer.

Parmi ces peer-to-peer, on cite:

- Gnutella
- Fastrack dont l'application la plus connue est Kazaa
- BitTorrent

- eDonkey utilisé notamment par le logiciel eMule

Les réseaux peer-to-peer se répartissent en plusieurs grandes catégories, selon leur architecture, relative à leur topologie (centralisé, semi-centralisé, décentralisé).

5.1.1. Le modèle centralisé

Cette architecture est basée sur un serveur central qui gère l'identité des utilisateurs, l'indexation des fichiers, les recherches et la liaison des peers. Le serveur attribue une adresse à chaque peer (client) possédant le fichier recherché, les échanges de données entre peer se font de manière explicite, sans passer par le serveur, contrairement à l'indexation qui est centralisée, ce genre d'approche présente une vulnérabilité majeure, une fois le serveur atteint le réseau s'écroule.

Exemples : Napster, Audiogalaxy, eDonkey2000

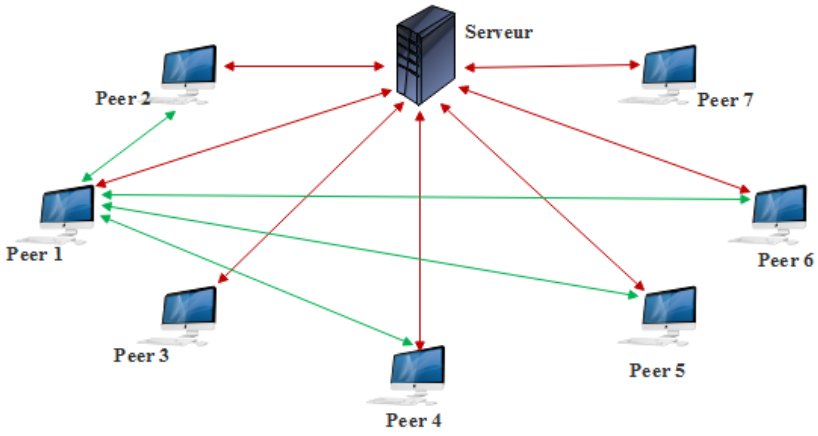


Figure 4. Exemple d'architecture centralisée

Aperçu sur Napster :

Napster a été conçu par Shawn Fawning en 1998 pour le partage et le téléchargement de fichiers en Peer-to-Peer, spécialisé dans les fichiers audio, musicaux généralement sous format MP3 ou WMA. Il est basé sur un système semi-centralisé, la listes des musiques partagées par les peers connectés est mise à jour grâce à un serveur central ceci en temps réel. Lorsqu'un serveur reçoit une requête de recherche de la part d'un client il lui transmet la liste des peers (utilisateurs) possédant ce

titre. Lors d'une requête de téléchargement, le serveur met directement en relation les deux machines concernées par ce téléchargement. [7]

Avantage:

- Performance
- Contrôle d'accès
- Le confort d'utilisation est grand, puisqu'il n'y a pas de soucis de connexion au bon serveur.
- La recherche de document est facilitée. En effet, le serveur est omniscient : si un fichier est disponible sur le réseau, on est en mesure de le savoir systématiquement.

Inconvénient*:*

- Single point of failure: il suffit de supprimer le serveur pour que l'intégralité du réseau soit inactif.
- Passage à l'échelle
- Ne gère pas l'anonymat

5.1.2. Le modèle Semi-décentralisé (Hybride)

Dans cette architecture, le contenu est distribué mais les fonctions de recherche, localisation, indexation et ainsi que la publication sont centralisées. Le modèle hybride a pour but de profiter des avantages des deux types de réseaux (centralisé et décentralisé). En effet sa structure permet de diminuer le nombre de connexions sur chaque serveur, et ainsi d'éviter les problèmes de bandes passantes. Ce type d'architecture est utilisé dans les réseaux FastTrack tel que Kazaa.

Ces réseaux utilisent des nœuds spéciaux appelés "super-peer" réalisant les indexations des fichiers partagés et servant d'intermédiaire pour les nœuds qui leur sont rattachés. Ils sont choisis en fonction de leur puissance de calcul, leur bande passante.

Les serveurs utilisés dans eDonkey2000 peuvent eux aussi être vu comme des super-peers (Très nombreux serveurs spécialisés communiquent entre eux et servent d'intermédiaire pour les clients).

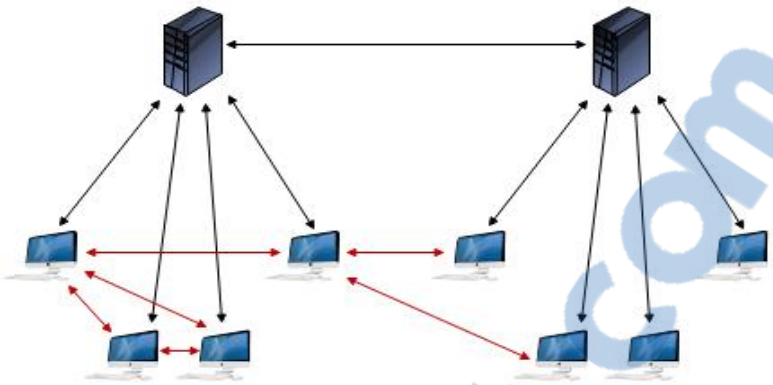


Figure 5. Exemple du modèle super-peer

Aperçu sur FastTrack : kazaa [8]

Les protocoles FastTrack et Kazaa ont été créé et développés par des programmeurs estoniens de BlueMoon Interactive dirigés par Jaan Tallinn. FastTrack est un protocole peer-to-peer utilisé par les programmes de partage de fichiers Kazaa, Grokster, iMesh, et Morpheus. FastTrack, Classifié parmi le type de peer-to-peer hybride, il dispose de plusieurs serveurs pouvant être appelés des super-peers, Ils permettent de gérer l'indexation des données, les recherches et la mise en relation des utilisateurs pour commencer les échanges.

Ces super-peers sont spécialement désignés avec une bande passante élevée, un espace disque et une puissance de traitement. Lorsqu'un peer rejoint le réseau, il est affecté à un super-peer. Le peer informe ensuite son super-peer du contenu qu'il va partager. le super-peer joue le rôle d'un serveur d'annuaire en maintenant une base de données qui mappe la liste des fichiers partagés de ses peers assignés vers les autres peers. Les super-peers créent ensemble une superposition structurée de super-peers, ce qui rend la recherche de contenu plus efficace.

Une requête de recherche dans ce réseau est acheminée vers un super-peer puis inondée dans le réseau superposé. Une requête de réponse sera envoyée par le super-peer contenant la liste des peers ayant le contenu.

Avantage :

Sa force réside dans la facilité d'utilisation puisque son interface est très intuitive et comporte un lecteur multimédia et une bibliothèque pour organiser ses fichiers.

Inconvénient :

Les réseaux hybrides ne sont plus dédiés à un seul serveur, car la base de données est distribuée parmi les super-peers. De plus, la taille de la base de données est relativement limitée, car chaque super-peer ne suit que le contenu de ses peers assignés. Cependant, l'inconvénient de cette approche est qu'elle est considérablement compliquée et nécessite une maintenance non triviale du réseau de recouvrement. En outre, le fait que les super-peers peuvent avoir plus de responsabilités que les peers ordinaires peut entraîner un goulot d'étranglement.

5.1.3. Le modèle décentralisé (purs)

Les réseaux totalement décentralisés sont dits "pure". Dans ce type d'architecture,l'ensemble des nœuds sont égaux et jouent le même rôle. Chaque peer gère donc les recherches et le partage, sans passer par un serveur central ou des super-peer, en utilisant en général des techniques de "flood" ou inondation. [3]

Aperçu sur Gnutella [7]

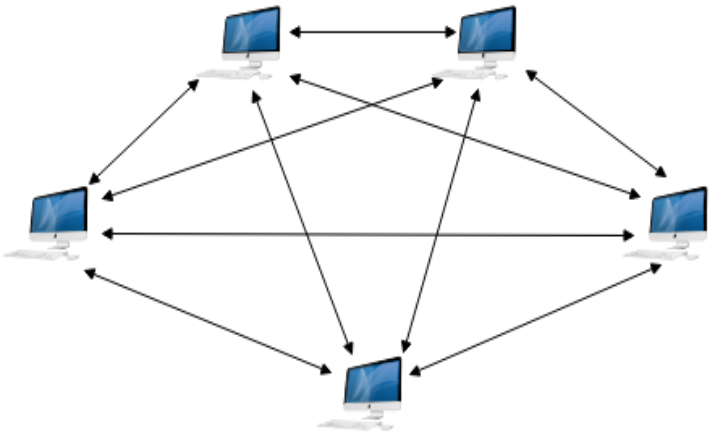


Figure 6. Exemple du modèle décentralisé

Gnutella est le premier réseau peer-to-peer décentralisé non structuré par inondation (flooding), crée en mars 2000 par Justin Frankel et Tom Pepper.

Pour rejoindre le réseau Gnutella, le client commence par envoyé un message à ses voisins disponible (le ping), qui l'envoient a leur tour a leur voisins et ainsi de suite. Les nœuds qui ont reçu le message de connexion répondent en fournissant leur adresse IP, le numéro de port ainsi que la liste et la taille des données qu'ils partagent (le pong).

Pour lancé une recherche le client envoie une requête (Query) a tous ses voisins qui vont eu même la transmettre à leurs voisins et ainsi de suite jusqu'à TTL=0. Tous les nœuds qui sont concerné par la requête vont répondre à l'émetteur du signal (Queryhit). La réponse remonte de proche en proche jusqu'au client initial qui va par la suite envoyer une requête de téléchargement directement au client qui possède le fichier. Si les données sont derrière un firewall un message Push est utilisé.

Les requête sont stoppée par l'expiration de TTL qui a une valeur fixé au début ceci pour éviter de parcourir la totalité du réseau et retourner une réponse négative.

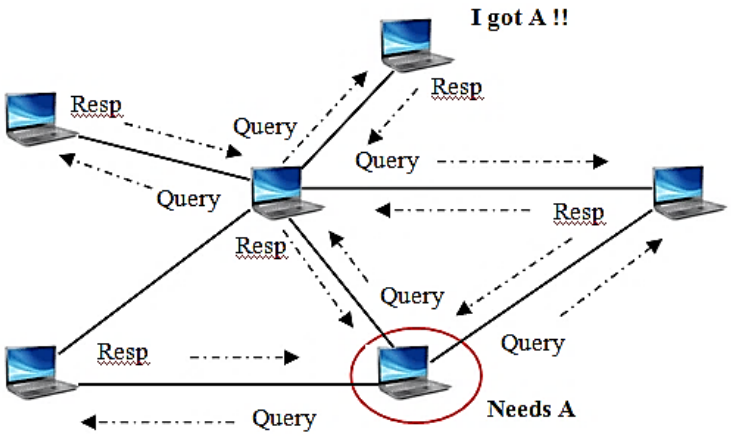


Figure 7. La recherche et téléchargement avec Gnutella

Dans Gnutella, chaque peer dispose de 6 types de messages principaux:

- **Ping** : Utilisé pour découvrir les autres peers sur le réseau. Un peer qui reçoit un Ping doit répondre avec un ou plusieurs Pong.
- **Pong** : C'est la réponse à un Ping; ce message contient l'adresse IP et le port du peer qui répond, ainsi qu'une information sur les fichiers et leurs tailles partagés par celui-ci.

- **Query** : Message pour la recherche de fichier sur le réseau. Un peer qui reçoit un Query répondra par un Query Hit s'il trouve une correspondance avec un ou plusieurs de ses fichiers partagés et les critères de recherche.

- **Query Hit** : C'est la réponse à un Query.

- **Get**: téléchargement des données. • Push : Permet de télécharger des données depuis un peer qui se trouve derrière un Firewall.

Avantages et inconvénients:

On peut distinguer clairement les avantages de gnutella vu que c'est un réseau purement peer-to-peer chose qui le rend tolèrent au panne

Mais pourtant ce réseau ne peut échapper des inconvénients, citons une dégradation de performance sur des grands réseaux, et une faiblesse en domaine de sécurité, disponibilité, fiabilité et scalabilité.

5.2. Peer-to-Peer structurés

Les peer-to-peer structurés sont basés sur l'élaboration d'une DHT (Distributed Hash Table) permettant de structuré "placer" les nouveaux nœuds dans le réseau. Chaque nœud recueille une liste de voisins avec lesquels il pourra communiquer. Il s'agit ici de voisins "logiques", ceux-ci pouvant se trouver à l'autre bout du monde. De plus, chaque peer gère une partie distincte du contenu du réseau qui sont identifiés par un couple clé/valeur.

Parmi les algorithmes structurés, on peut citer: Chord - CAN - Tapestry - Pastry - Kademlia utilisé par exemple par Overnet - Viceroy – Freenet.

5.2.1. Le protocole Chord

Chord est l'un des premiers réseaux P2P de recherche basé sur les tables de hachage distribuées (DHT) qui repose sur une structure en anneau, développé par un groupe de chercheur dans MIT,

Une fonction de hachage SHA-1 génère un identifiant à « m » bits pour chaque peer à partir de son adresse IP. Ensuite, chaque peer est placé dans l'anneau de manière à ordonner les identifiants

par ordre croissant. Ainsi, chaque peer d'identifiant n est responsable de l'intervalle de clés]précèdent(n), n] . un peer Chord a la connaissance de son prédécesseur et de son suivant.

Une table de hachage distribuée stocke des peers clé-valeur en attribuant des clés pour différents ordinateurs (appelés «nœuds»); un nœud va stocker les valeurs de toutes les clés pour lesquelles il est responsable

Il a pour particularité de disposer d'algorithmes d'une complexité d'au plus O(logN) requêtes pour trouver une information dans un anneau de N éléments en utilisant le principe des tables de hachage distribuées. Afin d'obtenir cette complexité, chaque peer maintient une liste de successeurs. Cette liste correspond aux peers successeurs en puissance de 2 (comme illustré sur la Figure 8).[9]

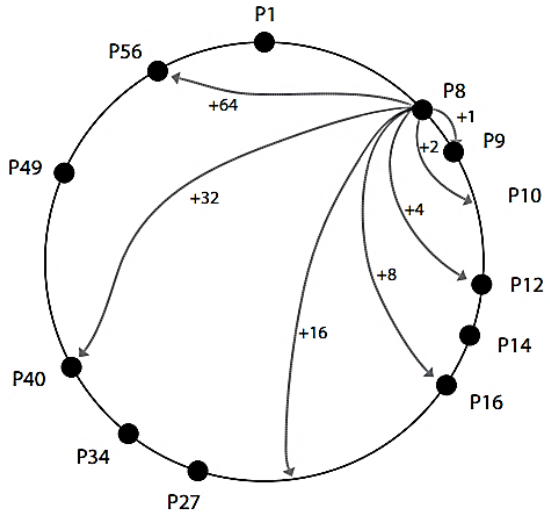


TABLE DES SUCCESSEURS DE P8

P8 + 1	P9
P8 + 2	P9
P8 + 4	P12
P8 + 8	P16
P8 + 16	P16
P8 + 32	P40
P8 + 64	P56

Figure 8. Exemple de réseau peer-to-peer Chord

En bref nous pouvons tirer à propos de Chord les conclusions suivantes :

- Les nœuds et les données ont des identifiants obtenus par hachage.
- Chaque nœud gère les objets de clés proches de son nodeID.
- Chaque nœud maintient une table des nœuds des voisins connus sous le nom de table des fingers

• Quand un nœud veut faire une recherche il consulte cette table pour savoir à quel nœud il va envoyer la requête.

• Cette table a un cout de l'ordre de O(log(N)) avec N le nombre total de nœuds du système, donc chaque nœud ne connait que O(log(N)) nœuds.

5.2.2. Le protocole CAN :

CAN ou Content Addressable Network, est un autre protocole P2P basé sur les tables de hachage distribuées ayant une infrastructure décentralisée et distribuée s'appuyant sur une topologie à N dimensions.

Prenons l'exemple d'une partition dans un espace à deux dimensions, Les ressources sont réparties dans une zone délimité par des coordonnées comprises entre 0 et 1 sur chaque côté, l'ensemble de l'espace est divisé entre l'ensemble des peers inscrits de sorte que chaque peer soit responsable d'une zone, en maintenant à jour la liste de ses voisins (nord, est, sud, ouest) , de plus que l'ensemble de l'espace est constamment alloué. [10]

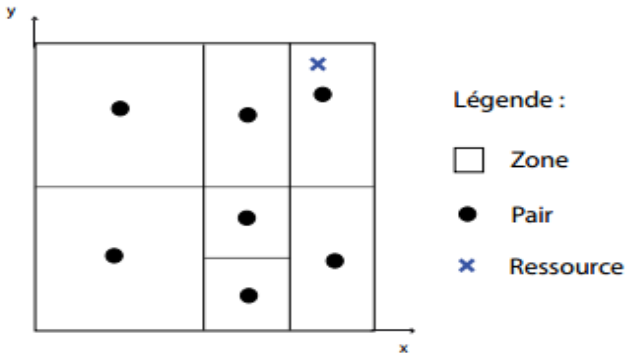


Figure 9. Exemple de réseaux peer-to-peer CAN en 2 dimensions

Conclusion

Ce chapitre montre à travers les différents points développés l'intérêt particulier porté aux architecture peer-to-peer qui représentent un domaine de recherche très actif grâce à leurs actuelles utilisations et leurs applications en comparaison avec l'architecture client-serveur.

Ainsi il détaille et explique les différentes catégories de l'architecture peer-to-peer qui sont classées en trois grandes familles: les réseaux centralisés, les réseaux purs et les réseaux hybrides, chacun de ces réseaux possède des caractéristiques bien différentes, ils offrent beaucoup d'avantages indéniables tels que: la répartition de la charge et la tolérance aux pannes, mais aussi quelques inconvénients liés surtout à l'aspect sécuritaire des données circulant dans le système. Aussi nous avons vu un exemple de protocole avec chaque type comme Napster, fastTrack :Kazaa, Gnutella, chord, kademlia…

Le tableau ci-dessous résume les différents types de systèmes P2P, décrits précédemment, selon les critères suivants : autonomie, efficacité, expressivité de la requête, tolérance aux pannes, sécurité et qualité de service.

	Non structure			**Structuré**
	Centralisé	**Hybride**	**Décentralisé**	
Autonomie	Faible	Moyenne	Forte	Forte
Exhaustivité	Forte	Moyenne	Faible	Forte
Expressivité de la requête	Forte	Forte	Forte	Faible
Tolérance au panne	Faible	Moyenne	Forte	Forte
Sécurité	Moyenne	Moyenne	Moyenne	Moyenne

Tableau 2: Comparaison des systèmes P2P

Chapitre2

Cryptographies

Introduction :

La cryptographie est la science qui étudie les principes et méthodes mathématiques appliqués à la sécurité de l'information pour des buts tels que la confidentialité, l'intégrité des données, les entités de validation pour offrir plus de sécurité.

La cryptographie a tendance à développer des techniques pour sécuriser des informations sensibles transférées via des réseaux non sécurisés (comme l'Internet) afin que les données ne puissent pas être lu ou modifié par des personnes non autorisées ou inconnues.

Ce chapitre présentera quelques types de la cryptographie moderne qui sont utilisés dans ce projet, et qui peuvent aider le lecteur à comprendre certains aspects de la cryptographie moderne.

1. Histoire de la cryptographie :

Les ordinateurs et le réseau Internet d'aujourd'hui font entrer la cryptologie dans son ère moderne. La grande invention de ces dernières décennies fut la cryptographie à clefs publiques et l'utilisation des empreintes digitales. Les années ci-dessus présentent un petit aperçu historique de l'évolution de la cryptographie moderne :

1970 : Au début des années 1970, **Horst Feistel** a mené un projet de recherche à l'**IBMWatson Research Lab** qui a développé le chiffre Lucifer, qui inspira plus tard le chiffre **DES** et d'autres chiffres.

1976 : **WhitfieldDiffie** et **MartinHellman** publient **New Directionsin cryptography,** introduisent l'idée d'un système à clef publique. Ils donnent une solution entièrement nouvelle au problème de l'échange de clés.

Novembre 1976 : **DES** est un algorithme très répandu à clef privée dérivée du chiffre **Lucifer** de **Feistel** (de chez **IBM**) dans sa version à 64 bits. Il sert à la cryptographie et l'authentification de données.

Avril 1977 : **RSA** signifie **Rivest-Shamir-Adleman**, en l'honneur de ses trois inventeurs : **Ron Rivest**, **Adi Shamir** et **Leonard Adleman** qui l'ont inventé en 1977. Le brevet de cet algorithme appartient à la société américaine **RSA Data Security**, **RSA** est un algorithme à clé publique qui sert aussi bien à la cryptographie de documents, qu'à l'authentification.

1978 : L'algorithme **RSA** est publié dans les Communications de l'**ACM**.

1984 : L'algorithme **El Gamal** basé sur le problème du logarithme publié par **Tahar El Gamal**.

1985 : Victor Miller et Neal Koblitz utilisent les courbes elliptiques pour la cryptographie.

1987 : Le **RC4** est développé par **Ronald L. Rivest** pour la **RSASecurity** et sera gardé secret jusqu'en 1994, où l'algorithme est rendu public anonymement dans une liste de distribution de **Cypherpunks**.

1990 : **Xuejia Lai** et **James Massey** publient **A Proposal for a New Block EncryptionStandard**, un algorithme de cryptage des données International l'**IDEA** pour remplacer le **DES**. L'**IDEA** emploie une clef de 128 bits et utilise des opérations convenantes à tout type d'ordinateurs, permettant donc une programmation plus efficace.

1990 : Première publication des résultats expérimentaux de la cryptographie quantique par **Charles H. Bennett** et **Gilles**.

1991 : **Phil Zimmermann** rend disponible version de **PGP**.

1992 : **MD5** est développé par **Ronald L. Rivest**.

1993 : **Bruce Schneier** conçoit **Blowfish** et**Don Coppersmith**a créer**SEAL**.

1994 : **Ron Rivest**, déjà auteur de **RC2** et **RC4**, publie **RC5**.Un standard régissant les signatures numériques voit le jour : le **DSA** (**DSS**).

1995 : **Nicolas Gisin** et son équipe distribuent des clefs secrètes à l'aide d'un câble optique de 25 kilomètres sous le lac Léman en codant les q-bits par la polarisation de photons (cryptographie quantique).

1998 : L'algorithme **Rijndael** est final et soumis au **NIST** pour devenir le nouveauchiffrement avancé : l'**AES**. Quinze autres algorithmes font partie du groupe : **MARS**, **RC6**, **Serpent** et **Twofish**.

Août 1999 : 11 sites répartis dans 6 pays factorisent le premier nombre ordinaire de 155 chiffres décimaux (512 bits). Un tel nombre aurait pu servir de clef dans un système de chiffrement moderne de type **RSA**, qui est utilisé dans le commerce électronique.

2000 : **Rijndael** devient **l'AES**, le standard du chiffrement avancé.[11]

2. La terminologie de la cryptographie :

- La cryptologie : est la science des messages secrets. Elle englobe la cryptographie et la cryptanalyse. Le mot cryptologie est utilisé comme un synonyme de cryptographie.[12]
- Cryptographie : Cette section regroupe l'ensemble des méthodes qui permettent de chiffrer et de déchiffrer un texte en clair afin de le rendre incompréhensible à tous ceux qui ne sont pas en possession de la clé de déchiffrement.
- La cryptanalyse : est la science complémentaire qui consiste à déterminer certaines propriétés de ces systèmes dans le but de reconstituer le texte en clair, souvent en absence des paramètres qui sont nécessaires pour le déchiffrement. [13]

Cryptologie = Cryptographie + Cryptanalyse

- Protocole : description de l'ensemble des données nécessaires pour mettre en place le mécanisme de cryptographie : ensemble des messages clairs, des messages cryptés, des clés possibles, des transformations.
- Signature : Chaine de caractères associées à un message donné (et éventuellement à une entité) et le caractérisant.[14]
- Texte en clair : c'est le message à protéger.
- Texte chiffré : c'est le résultat du chiffrement du texte en clair.

- Chiffrement : c'est la méthode ou l'algorithme utilisé pour transformer un texte en clair en texte chiffré.
- Déchiffrement : c'est la méthode ou l'algorithme utilisé pour transformer un texte chiffré en texte en clair.
- Clé : information (secrète) utilisée pour chiffrer un texte ou déchiffrer un texte chiffré.

3. Définition de la cryptographie :

Le mot cryptographie représente l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire de les rendre inintelligibles sans actions spécifiques. [15]

La cryptographie est l'art d'écrire des messages cryptés via l'utilisation de codes secrets ou un ensemble des principes, méthodes et techniques dont l'application assure le « crypter » et le « décrypter » des données (Figure 10). La signification de « crypter » et « décrypter » est donnée successivement comme suit :

- Crypter : rendre le message incompréhensible, c'est-à-dire transformé un texte en clair en texte chiffré.
- Décrypter : c'est l'action normale pour récupérer le texte en clair correspondant à un texte chiffré sans posséder la clé qui a servi au chiffrement.

Figure 10. Chiffrement / Déchiffrement

4. Objectif de cryptographie :

Le but fondamental de la cryptographie est de respecter adéquatement les quatre objectifs majeurs de la sécurité, en théorie et en pratique : [16]

La confidentialité : La confidentialité consiste à garder les informations secrètes à tout le monde sauf les personnes autorisées à les voir.

Intégrité des données :L'intégrité est de faire en sorte que l'information n'a pas été modifiée par des personnes non autorisées ou inconnues.

Authentification : personne ou entité d'authentification est d'assurer l'identité de cette personne ou entité.

La non-répudiation : La non-répudiation de l'information est la garantie qu'un message a bien été émis par son initiateur. La personne l'ayant émis ne peut pas renier l'avoir émis.

5. Technique de cryptographie moderne

Il existe deux principales catégories de cryptographie, classifié selon le type de clés de sécurité utilisées pour Crypter et décrypter les données. Ces deux catégories sont :

Techniques de cryptage asymétrique et symétrique.

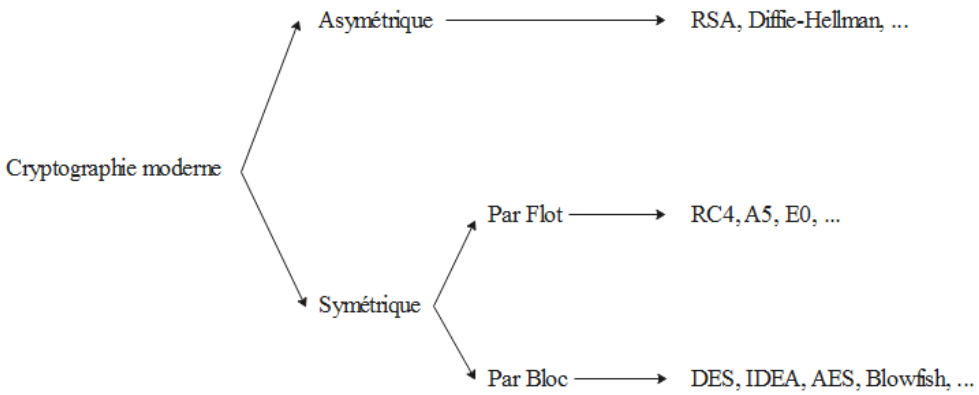


Figure 11. Classification de quelque algorithme de cryptographie

6. Cryptographie Asymétrique

Inventé en 1977 par Diffie et Hellman [17], le chiffrement asymétrique, aussi connu sous le nom de chiffrement à clé publique utilisant deux types de clé

- une clé publique connue de tous.

- une clé privée connue seulement du destinataire du cryptogramme.

Les données sont cryptées avec la clé publique et ne peuvent être décryptées qu'avec la clé privée. Signifiant que même si la clé publique est révélée il n'est pas possible de déchiffrer les informations.

Ce type de crypto-système permet de résoudre le problème de communication des clés du chiffrement symétrique, et permet aussi l'authentification, en se servant des fonctions de hachage.via la signature électronique qui est basée sur l'utilisation conjointe d'une fonction de hachage et de la cryptographie asymétrique

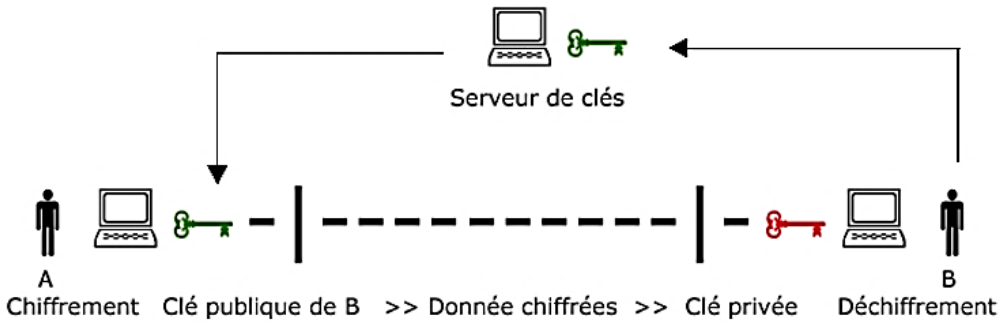


Figure 12. Chiffrement Asymétrique

6.1. Avantage et inconvénients du chiffrement Asymétrique

Avantage:

- Gestion sécurisée des clés : adapté aux environnements anonymes
- La clé publique peut être diffusée et utilisée sans risque.
- La clé secrète ne quitte jamais son possesseur (le chiffreur).

Inconvénient

- Algorithmes très lents
- Authentification de la clé publique.

6.2. Algorithme de chiffrement à clé publique

L'algorithme RSA

Le nom RSA représente les initiales des noms de ses inventeurs : Ron **R**ivest, Adi **S**hamir et Leonard **A**dleman qui l'ont inventé en 1977, c'est l'algorithme de cryptage asymétrique le plus connu et le plus courant parmi les algorithmes à clé public vu que c'est le premier système mise au point dans ce genre, et aussi grâce à la sureté qu'il offre pour le transfert de clé. Le brevet de cet algorithme appartient à la société américaine RSA Data Security, qui fait maintenant partie de Security Dynamics et aux Public Key Parteners, (PKP à Sunnyvale, Californie, Etats-Unis) qui possèdent les droits en général sur les algorithmes à clé publique.

Fondée sur la difficulté de factoriser des grands nombres qui est le produit de deux très grands nombres premiers, elle est considérée comme une fonction à sens unique, utilisant une clé de longueur importante et variable 512, 640, 1024, 2048 bits.

Remarque : nous avons utilisé dans notre application l'algorithme RSA au début de l'échange pour échanger les clés symétriques de Blowfish et de RC4 qu'on détaillera par la suite.

7. Cryptographie Symétrique

Penchons-nous maintenant sur le chiffrement symétrique qui est le plus ancien des chiffrements, citant le chiffrement de César ou le chiffrement de Vigenère.

Le chiffrement symétrique aussi appelé chiffrement à clé privée ou chiffrement à clé secrète, c'est un chiffrement dans lequel la clé de chiffrement est la même que la clé de déchiffrement.

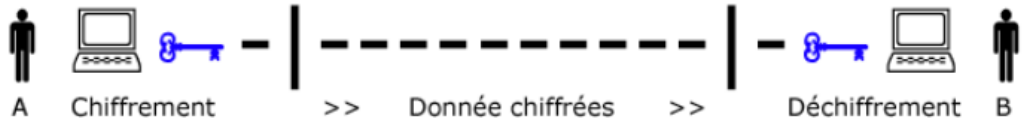


Figure 13. Chiffrement Symétrique

7.1. Avantage et inconvénient du chiffrement symétrique

Avantage

Le chiffrement/déchiffrement est très rapide grâce à la simplicité des algorithmes de chiffrement symétrique et au peu de ressources système qu'il utilise, contrairement aux algorithmes de chiffrement asymétrique qui sont beaucoup plus complexes.

Inconvénients

Le principal inconvénient d'un chiffrement symétrique repose sur l'échange de la clé secrète. D'autre part, une communication entre plusieurs personnes nécessite un grand nombre de clé selon le nombre d'interlocuteur. Ainsi, se pose le problème de distribution des clés. Il est nécessaire de garantir la confidentialité de cette clé. Les échanges qui suivront reposent sur celle-ci.

Le chiffrement symétrique n'assure pas l'authentification des données, contrairement au chiffrement asymétrique que nous aborderons ultérieurement et qui permet d'assurer des principes de sécurité supplémentaire.

7.2. Algorithme de chiffrement à clé secrète

7.2.1. Chiffrements symétrique par flux

Connu aussi sous le nom de « stream cipher », est une technique de chiffrement à clef symétrique qui opère sur chaque bit séparément de façon continue sur les données

La structure d'un chiffrement par flux se base sur un générateur de clés qui produit une séquence de clés k 1 ,k 2 ,...,k i .

L'algorithme RC4

Conçu en 1987 par Ronald Rivest, l'un des inventeurs du RSA il fut adopté par plusieurs protocoles de sécurisation tels SSL ou encore WEP pour les réseaux sans fils. Initialement gardé secret par la RSA, cet algorithme a été publié anonymement en 1994 sur Internet. Le RC4 reste la propriété de RSA Labs.[18]

RC4 est la méthode la plus simple de cryptage des données. Il est également plus rapide et plus adapté à l'application en continu. RC4 utilise la méthode de chiffrement de flux. Il acquiert moins de temps CPU, moins d'utilisation des ressources et aussi facile à mettre en œuvre par rapport au chiffrement des blocs.[19]

Le fonctionnement de RC4 :[18]

Le fonctionnement de RC4 commence par l'initialisation d'un tableau de 256 octets en répétant la clé tant que nécessaire jusqu'à le remplir. Suite à cela, des opérations très simples sont effectuées : les octets sont déplacés dans le tableau, des additions sont effectuées, etc. pour but de mélanger au maximum le tableau. Le résultat sera une suite de bits pseudo-aléatoires utilisés pour chiffrer les données via XOR.

Cet algorithme fonctionne sur les octets. Ainsi, la clé, de longueur variable, peut avoir une taille comprise entre 1 et 256 octets (de 8 à 2048 bits). Elle est utilisée pour initialiser un vecteur S de 256 octets. A tout moment, S contient une permutation de toutes les cellules le composant.

Initialisation:

L'initialisation d'un tableau [S] de 256 octets, les entrées de S sont égales aux valeurs de 0 à 255 par ordre croissant; C'est; S [0] = 0, S [1] = 1, ..., S [255] = 255.

Figure 14. Initialisation (1)

Un vecteur temporaire de longueur T (de longueur égale à S) est également créé et destiné à recevoir la clé

- Si la Longueur de la clé k est égale à 256, K est simplement transféré dans T.
- Si k est inférieur à 256 octets, il est recopié dans T jusqu'à atteindre la taille de T.

Avec [K] tableau d'octets de Secret Key.

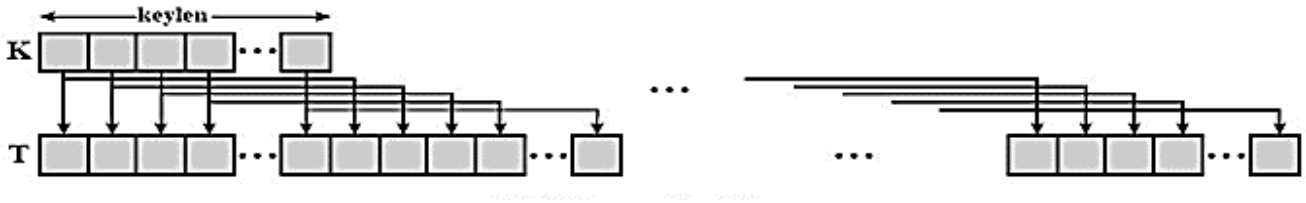


Figure 15. Initialisation (2)

Algorithme de planification de clé (Key Schedule Algorithm):

Ensuite, nous utilisons T pour produire la permutation initiale de (S). Pour chaque cellule S[i] de S, celle-ci sera échangée avec une autre cellule de S selon un calcul basé sur la valeur comprise dans la cellule T[i] correspondante, on peut représenter la permutation initiale de S comme suit :

```
    int j=0;

    for ( int i ; i<256 ; i++ )

    {

    j = ( j + S[i] + K[i] ) % 256;

    Swap( S[i], S[j] ); // On échange les valeurs de S[i] et
S[j]}
```

Ces deux dernières lignes de code portent le nom de KSA (Key Schedule Algorithm).

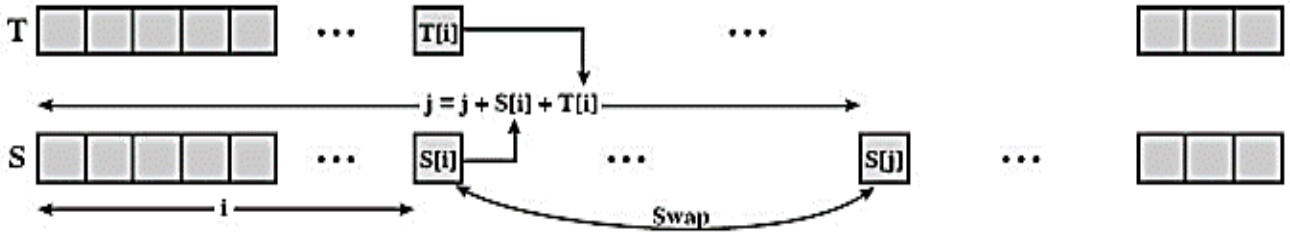


Figure 16. Initialisation (3)

Génération du flux pseudo aléatoire:

Il s'agit de la Génération du flux pseudo aléatoire. A partir de cet instant, la clé d'entrée n'est plus utilisée. Pour chaque S[i], on procèdera à un échange avec un autre octet de S, selon un schéma basé sur la configuration courante de S. Une fois arrivé à S [255], le processus redémarre à la cellule S [0]. On peut représenter la permutation initiale de S comme suit :

```
I = j = 0;

While (output_bytes)

{ I = (I + 1)% 256;

J = (j + S [i])% 256;

Swap (S [i], S [j]);

Output = S [(S [i] + S [j])% 256] }
```

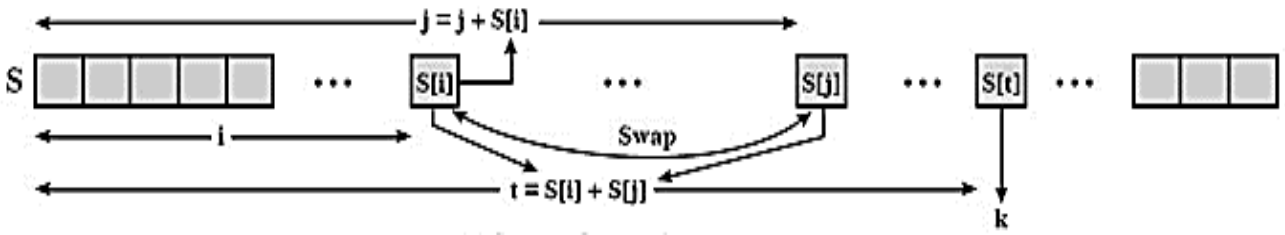


Figure 17. Génération de Flux RC4

La valeur de k est alors utilisée pour le chiffrement (⊕ avec le prochain octet de texte clair), ou pour le déchiffrement (⊕ avec le prochain octet de texte chiffré).

L'algorithme A5 [20]

C'est un algorithme à flot utilisé pour protéger les liaisons GSM, en chiffrant la communication entre le mobile et la borne reliée au réseau. Il existe deux algorithmes utilisés

dans GSM : A5/1 et A5/2. La version A5 / 1 est utilisée par environ 130 millions de clients en Europe, tandis que la version A5 / 2 est utilisée par 100 millions de clients sur d'autres marchés

L'algorithme A5/1 a comme paramètre une clé de taille 64bits et un vecteur d'initialisation de 22bits correspondant au numéro de trame, ainsi qu'un générateur pseudo-aléatoire composé de trois LFSR de longueurs respectives 19, 22 et 33.

Les spécifications d'A5/1 étaient initialement secrètes, mais elles ont été précisément révélées par Briceno, Goldberg et Wagner en 1999, suite à un travail de reverse-engineering.

7.2.2. Chiffrement symétrique par bloc

Définition

Un algorithme de chiffrement par bloc (Block Cipher en anglais) transforme des blocs de données de taille fixe en blocs de données chiffrées de la même taille. Les blocs font généralement 128 bits, mais ils peuvent aller de 32 à 256 bits selon l'algorithme. La transformation reste la même pour chaque bloc.

La plupart des chiffrements symétrique par bloc (DES « Data Encryption Standard », 3DES, TEA « Tiny Encryption Algorithm», IDEA « International Data Encryption Algorithm») sont construits sur l'approche du réseau Feistel.

Le réseau de Feistel [18]

Cette structure a été inventé en 1973 (par Horst Feistel, employé chez IBM) dans le cadre de Lucifer et popularisé par le DES.

Un chiffrement de Feistel est un chiffrement itératif opérant sur des blocs de 2n bits. Transformant toute fonction en permutation. Dans ce système de chiffrement, un bloc de texte en clair est découpé en deux ; la transformation du tour (rond) est appliquée à une des deux moitiés, et le résultat est combiné avec l'autre moitié par ou exclusif. Les deux moitiés sont alors inversées pour l'application du tour suivant.

L'avantage est que la fonction de chiffrement et la fonction de déchiffrement sont similaires.

citons quelques algorithmes qui utilise le réseaux de Feistel: Blowfish , DES, Triple DES, Lucifer, RC5.

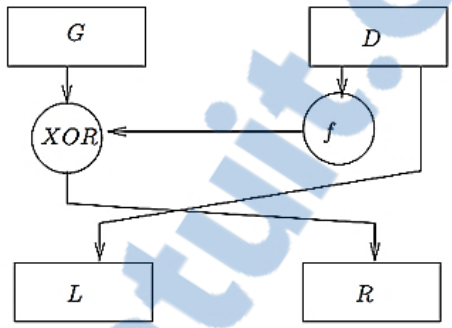


Figure 18. Structure de Feistel

L'algorithme DES :[15]

Créé par IBM dans les années 70 dans le cadre du projet LUCIFER et rendu public par le NBS4 (National Bureau of Standards) en 1981, il a été réécrit environ tous les 5 ans et sa plus récente version date de 1994, considéré de loin le moyen de chiffrement le plus sûr pour les données non militaires jusqu'à 1998.

DES est un algorithme de cryptographie par blocs, basé sur les réseaux de Feistel, il opère sur des blocs de 64 bits avec des clés de 56 bits (56bits pour chiffrer + 8 bits de parité pour vérifier l'intégrité de la clé) transformé en 16 sous-clé de 48 bits chacune.

Le bon fonctionnement de l'algorithme DES consiste à respecter les étapes suivantes :

1. Fractionner le texte en blocs de 64 bits (8 octets).
2. Permuter les blocs avec une permutation initiale IP.
3. Découper les blocs en deux parties de 32 bits chacune : gauche et droite, nommées G et D.
4. Assurer des tours de permutations et de substitutions répétées 16 fois, en faisant intervenir la clé cryptographique.

5. Concaténer les parties G et D et faire une permutation inverse de la permutation initiale.

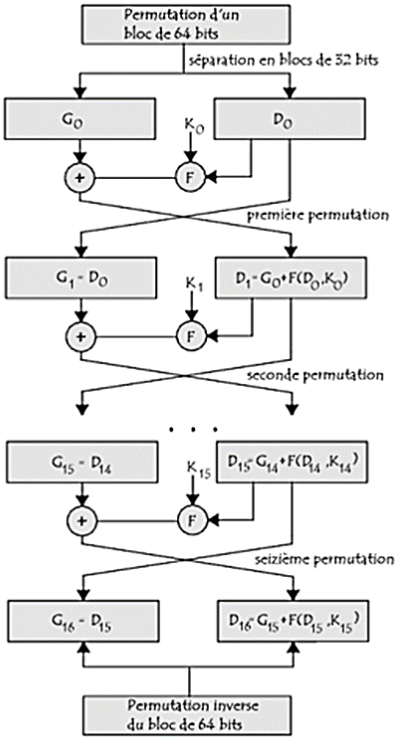


Figure 19. L'algorithme DES

L'algorithme AES

L'Advanced Encryption Standard, ou Rijndael a été officialisé en octobre 2000 en remportant le concoure pour créer un successeur au DES, lancé par NIST (National Institute of Standards and technology) en janvier 1997, cet appel à la candidature reçoit 15 proposition d'algorithme, 5 seulement furent choisis pour accomplir l'évaluation plus poussé en avril 1999 : MARS, RC6, RIJNDAEL, SERPENT et TOWFISH [21]. Au finale la NIST élit RIJNDAEL comme étant le nouveau standard pour joan Daemen et Vincent Rijmen devaient respecter certaines conditions comme la rapidité du chiffrement, simplicité de l'algorithme ou encore une large portabilité.[15]

L'algorithme AES est un algorithme symétrique de chiffrement par blocs de 128 bits (soit 16 octets) avec une clé de 128, 192 ou 256 bits. L'AES applique un certain nombre de rounds

dépendent de la taille de la clé: 10 tours pour une clé de 128 bits, et 14 tours pour une clé de 256 bits, et il travaille sur des blocs rectangulaires de 4 rangées et Nc colonnes, où chaque terme xi,j (appelé octet ou byte) est composé de 8 bits (b = b7b6b5b4b3b2b1b0) et peut être représenté algébriquement en degré polynôme ≤ 7 (b = b7X7 + b6X6 + b5X5 + + b4X4 b3X3 + b2X2 + b1x + B0) avec des coefficients dans {0,1}.le chiffrement et le déchiffrement s'effectuent de la même manière.[15]

L'algorithme de Blowfish

Blowfish a été créé en 1993 par Bruce Schneier comme une alternative rapide et gratuite aux algorithmes de cryptage existants.

Blowfish est un algorithme de chiffrement symétrique qui peut être utilisé efficacement pour le cryptage et la sauvegarde des données. Il faut une clé de longueur variable allant de 32 bits à 448 bits; 128 bits par défaut, ce qui le rend idéal pour sécuriser les données et qui peut être utilisé comme un remplacement pour l'algorithme DES. Blowfish n'est pas breveté, sans licence, et est disponible gratuitement pour tous les utilisateurs. Blowfish est basé sur une structure de type "réseau de Feistel", itérant une fonction de cryptage simple 16 fois. La taille du bloc est de 64 bits, et la clé peut être n'importe quelle longueur jusqu'à 448 bits.

Blowfish est un chiffrement de bloc avec clé de longueur variable. Il convient aux applications où la clé ne change pas souvent, comme un lien de communication ou un chiffreur automatique de fichiers. [22]

Principe général de l'Algorithme Blowfish :[23]

- Manipule les données en gros blocs.
- A une taille de bloc 64 bits.
- Possède une clé évolutive, de 32 bits à au moins 256 morceaux.
- Utilise des opérations simples qui sont efficaces sur les microprocesseurs. « Par exemple, la recherche de table exclusive ou supplémentaire, la multiplication modulaire. »
- Employer des sous-clés pré computables : sur les systèmes à mémoire étendue, ces sous-clés peuvent être précomptées pour un fonctionnement plus rapide.

- Se compose d'un nombre variable d'itérations.
- Utilise des sous-clés qui sont un hash à sens unique de la clé.
- N'ont pas de structures linéaires qui réduisent la complexité de la recherche exhaustive.
- Utilise un design simple à comprendre. Cela facilite l'analyse et augmente la confiance dans l'algorithme. En pratique, cela signifie que l'algorithme sera un chiffrement de bloc itératif Feistel.

Description et détail de l'algorithme blowfish :[24]

Blowfish est un code à bloc variable de 64 bits, et utilise une clé de longueur variable, L'algorithme se compose de deux parties : une partie expansion de la clé et une partie encodage des données. La partied'expansion de clé convertit une clé de plus de 448 bits en plusieurs tableaux de sous-clés totalisant 4168 octets.

Le cryptage des données se fait via un réseau Feistel à 16 tours (rond). Chaque cycle se compose d'une permutation dépendante de la clé, et d'une substitution dépendante de la clé et des données. Toutes les opérations sont des XOR et des ajouts sur des mots de 32 bits.

Sous-clés(partie expansion de la clé) **:**

Blowfish utilise un grand nombre de sous-clés. Ces clés doivent être préalablement calculées avant d'effectuer le codage ou le décodage des données.

1. Le champ P se compose de 18 sous-clés de 32 bits: P1, P2, ..., P18.
2. Il existe quatre S-Boxes de 256 éléments chacune. Les S-Boxes acceptent un mot de 8 bits en entrée et produisent une sortie de 32 bits. Entrées chacune :

S1,0, S1,1, ..., S1,255;
S2,0, S2,1, ... ,, S2,255;
S3,0, S3,1, ..., S3,255;
S4,0, S4,1, ... ,, S4,255.

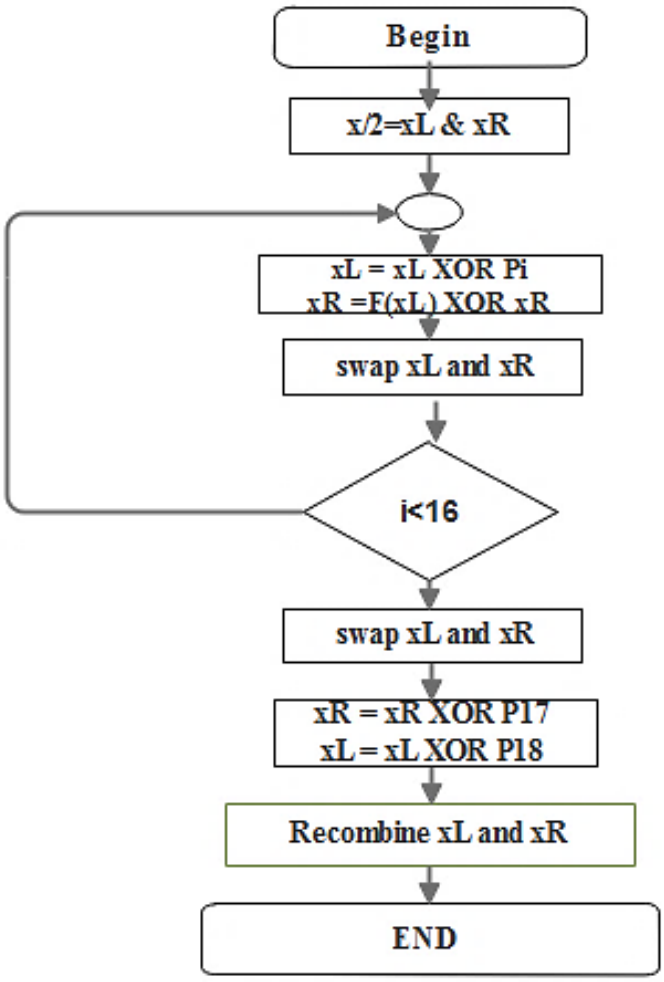
Cryptage (partie encodage des données) :

- Blowfish a 16 rounds.
- L'entrée est un élément de données 64 bits, x

- Début chiffrement :

```
    Diviser x en deux moitiés de 32 bits : xL, xR.
    Ensuite, pour i = 1 à 16 :
    XL = xL XOR P[i]
    XR = F (xL) XOR xR
    Permuter xL et xR
    Fin Pour
    Après le seizième tour, échangez xL et xR À nouveau pour annuler le dernier échange.
    Ensuite, xR = xR XOR P [17]
    xL = xL XOR P [18]
    x = XL XOR P18.
    Fin chiffrement :
    Enfin, pour obtenir le Texte chiffré il suffit de combiner xL et xR.
```

- Le décryptage est exactement le même que le cryptage, sauf que P1, P2, …, P18 sont utilisés dans l'ordre inverse.



-

Figure 20. L'algorithme de Blowfish

La fonction F :

La F-fonction de Blowfish divise une entrée (Lx) de 32 bits en quatre morceaux de 8 bits : a, b, c et d et les utilise comme entrées pour accéder aux S-Boxes. S1/2/3/4

Les sorties sont additionnées avec une somme de modulo 232 et l'algorithme effectue un XOR entre les deux sous-totaux pour produire la sortie finale de 32 bits.

((S1, a + S2, b MOD 232) XOR S3, c) + S4, d MOD 232

On peut représenter la fonction Blowfish comme suit :

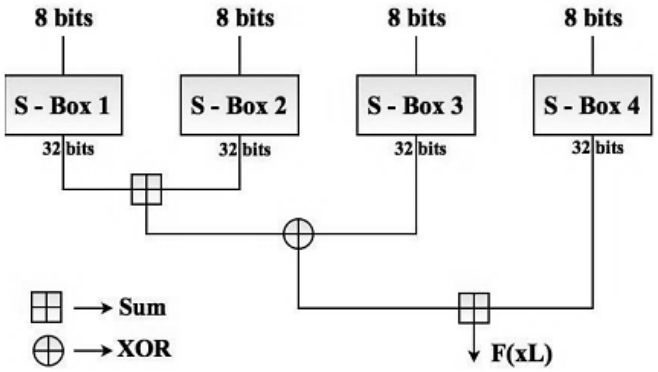


Figure 21. La F-fonction de Blowfish

Génération des sous-clés :[24]

Les sous-clés sont calculées en utilisant l'algorithme Blowfish. La méthode de la construction est la suivante :

1. Initialisez d'abord le tableau P et les quatre boîtes S avec des valeurs dérivées à partir des chiffres hexadécimaux de pi, qui ne contiennent pas de motif évident. Par exemple : P1 =0x243f6a88, P2 = 0x85a308d3, etc.
2. Le OU exclusif P1 avec le premier bloc de 32 bits de la clé, et P2 avec le second 32 bits de la clé. Et ainsi de suite pour tous les bits de la clé (jusqu'à P14).
3. Encoder la chaîne de 0 avec l'algorithme Blowfish, en utilisant les sous-clés décrites aux étapes (1) et (2).
4. Remplacez P1 et P2 par la sortie de l'étape (3).
5. Encoder la sortie de l'étape (3) en utilisant l'algorithme Blowfish avec les nouvelles sous-clés modifiées.
6. Remplacez P3 et P4 par le nouveau texte chiffré (la sortie de l'étape (5)).
7. Continuer le processus, en remplaçant toutes les entrées du tableau P et les quatre S-boxes dans l'ordre.

L'algorithme génèrera 4 kilo-octets de données qui sont traitées lors de cette opération, et qui nécessites en tout 521 itérations. Les applications peuvent stocker les sous-clés plutôt que d'exécuter plusieurs fois ce processus de dérivation.

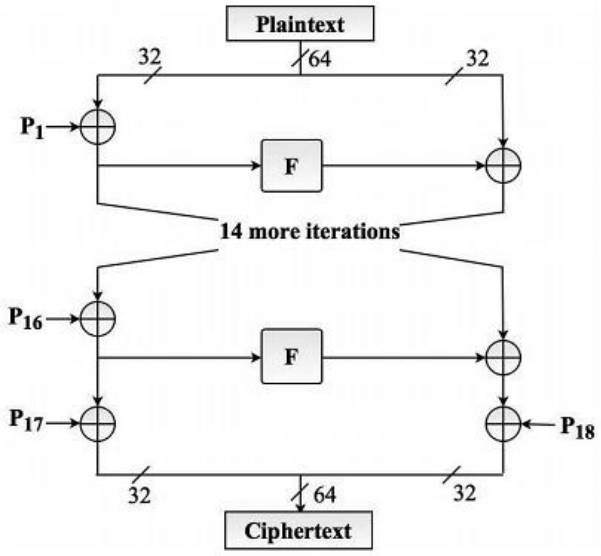


Figure 22. Calcule de sous-clé

Conclusion

Des études ont été faite pour évaluer la performance entre les algorithmes de cryptographie à clé symétrique en comparant la vitesse de l'algorithme pour cryptage et décryptage.[25]

Blowfish est encore nouveau et pas très répandu. Mais cet algorithme à clairement montrer de meilleurs performance en terme de sécurité et de rapidité par rapport aux autres algorithme de cryptage symétrique sachant qu'il n'existe pas à ce jour une attaque complète sur Blowfish, car une attaque ne peut pas être utilisée contre 16 tours Blowfish.

Afin de garantir une meilleure sécurité dans les communications et les transactions utilisant les infrastructures réseaux et internet, la cryptographie assure des services supplémentaire tels que le service d'authentification, d'intégrité et de non-répudiation en plus de la confidentialité.

Chapitre3

Introduction à JXTA

Introduction

La plus part des applications P2P existantes ne permettent pas l'interopérabilité. C'est depuis cette problématique et pour but de satisfaire l'informatique distribuée moderne que SUN Microsystem a créé le projet JXTA

JXTA vient du mot anglais " Juxtapose ". En effet, le but de JXTA est de garantir les échanges entre tous type de plateforme et de protocoles. Il est conçu pour être accepté par tous les développeurs, indépendamment des langages de programmation préférés ou des environnements de développement. Ce projet a été gardé secret mais au finale Sun a décidé de le rendre Open Source dans l'objectif de le populariser en vue d'en assurer le succès auprès des développeurs et utilisateurs.

JXTA est la principale plateforme sur laquelle nous avons mis en œuvre nos travaux de recherche, c'est pourquoi, dans ce chapitre nous détaillons particulièrement les concepts de JXTA et ses protocoles.

1. Le projet JXTA

JXTA a été le fruit d'un projet de Sun Microsystems. Une petite équipe de développement sous la direction de Bill Joy et Mike Clary fondé en avril 2001 pour subvenir au besoin d'une langue P2P commune avec une base solide et bien définie pour permettre d'interconnecter n'importe quel système sur n'importe quel réseau, ceci via des messages XML permettant à n'importe quelle entité sur le réseau (téléphones cellulaires, Serveurs et PCs) de communiquer.

Le projet JXTA est une plateforme qui utilise un ensemble de protocoles P2P ouverts basés sur les technologies Java proposée par Sun Microsystems offrant un ensemble de mécanismes simples et permettant le développement d'applications peer-to-peer pour gérer les fonctions centrales des communications de manière simple, rapide et facile et se focaliser sur les couches hautes des applications.

JXTA est basé sur la Licence Apache Software, sa dernier version est JXTA 2.7 qui date de Mars 2011, les protocoles JXTA peuvent aussi être implémentés en c, c ++, perl ou tout autre langage de programmation et pas seulement en JAVA. [26]

2. La structure de JXTA [26]

JXTA se divise en trois principales couches, chaque couche s'appuie sur les capacités de la couche ci-dessous, ajoutant ses propres fonctionnalités :

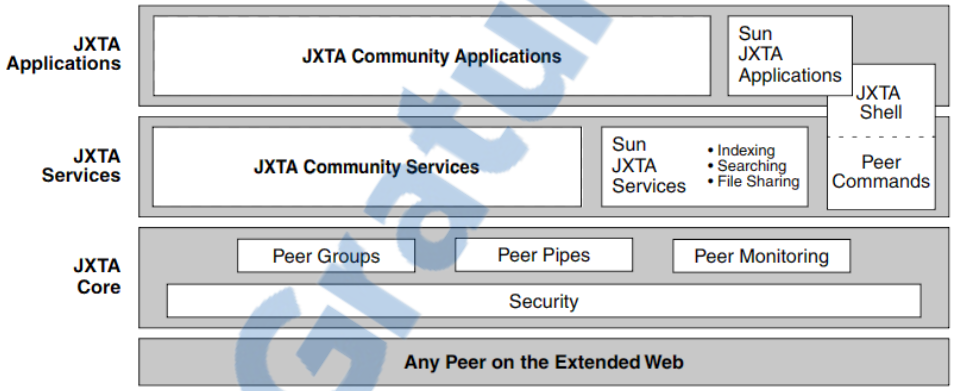


Figure 23. Structure de JXTA

• **Le noyau.** Contient les principales fonctionnalités du système, soit l'établissement des peers, la gestion des canaux de communication et de la sécurité , création et gestion des groupes de peers ou encor les mécanismes basiques de découverte, d'appartenance ou de surveillance (monitoring), L'intégrité, la confidentialité et la sécurité peuvent aussi être garanties par le pipe. Le Noyau est basé sur les Concepts et les Protocoles.

• **La couche des services.** C'est une couche qui élargie les fonctionnalités offertes par le noyau et contient les services de Sun, Services intégrés de la plate-forme JXTA, qui fournissent des mécanismes d'indexation, de recherche et de partage de données, d'authentification et de stockage, comme elle englobe des fonctionnalités supplémentaires de La communauté JXTA (développeurs open source travaillant avec Project JXTA). Ces mécanismes sont requis par différentes applications P2P.

• **La couche des applications.** Tout comme la couche précédente, il existe deux types d'application, celle proposées par Sun et d'autres développées par la communauté JXTA combinées pour former un P2P complet. Cette couche hérite d'avantage des capacités de la couche service telles que la messagerie instantanée, partage de données, la gestion et la livraison de données…etc, tel des services de base pour le développement d'une application accédant aux mécanismes fournis dans la Couche Noyau.

3. Objectifs de JXTA :

Les trois principaux objectifs de JXTA sont :

1. Interopérabilité JXTA est créé pour permettre aux peers interconnectés d'interagir de façon transparente à travers différents systèmes P2P et différentes applications.

2. Indépendance des applications La technologie JXTA est conçue pour être acceptée par tous les développeurs, indépendamment des langages de programmation (tels que C , Java ™), des environnements de développement telles que les systèmes d'exploitation Microsoft Windows et UNIX) ou des plateformes réseau (telles que TCP / IP ou Bluetooth).

3. Ubiquité signifie la possibilité de le mettre en œuvre sur tout type de périphérique tel que les capteurs, les PDA, les routeurs réseau, les PC, les Serveurs, les organiseurs, les Clients GPS,....

4. Sécurité de JXTA [27]

Suivant les primitives de sécurité :

• Une bibliothèque de cryptage simple prenant en charge des fonctions de hachage (par exemple, MD5), des algorithmes de cryptage symétriques (par exemple RC4) et Des algorithmes cryptographiques asymétriques (par exemple, Diffie-Hellman et RSA).

• Un cadre d'authentification modélisé par PAM (module d'authentification enfichable, défini pour la première fois par la Plateforme UNIX puis adopté par l'architecture de sécurité Java).

• Un simple schéma de connexion basé sur un mot de passe qui, comme d'autres modules d'authentification, peut être branché dans le PAM cadre.

• Un simple mécanisme de contrôle d'accès basé sur des groupes de peers, où un membre d'un groupe se voit automatiquement accordé l'accès à toutes les données offertes par un autre membre pour le partage, alors que les non-membres ne peuvent pas accéder à ces données.

• Les services de démonstration appelés InstantP2P et CMS (service de gestion de contenu) utilisent également des services supplémentaires et les fonctionnalités de sécurité fournies par la plateforme Java sous-jacente.

5. Les concepts de base de JXTA

Pour pouvoir mettre en œuvre une plateforme générique, applicable à n'importe quel type de service P2P, les concepteurs de JXTA ont tout d'abord identifié les différents concepts qui sont à la base de JXTA:[26]

Endpoint.

Un endpoint est une source ou une destination logique de toute ressource transmise dans un réseau via un protocole spécifique de communication. Un peer peut avoir de multiples endpoints donc il peut communiquer via différents protocoles.

Identifiants.

Dans JXTA les peers, les peergroups, les pipes, les publicités, les services… etc. sont des ressources et des entités qui doivent être identifié de manière unique dans un environnement d'exécution local. Un identifiant dans jxta est généré aléatoirement en tant que UUID avec une taille de 128 bits.

Une identification de JXTA est une URN (Uniform Ressource Name) standard dans le namespace d'identification de JXTA composé d'une chaine de 17 ou 30 caractères qui commence par « urn:jxta:uuid-78746150325033F3BC76FF13C2414C »« un identifiant de namespace = jxta

; une spécification de format = urn ; un ID = valeur unique ». Chaque identification de JXTA, indépendamment du format ou du type doit être non-ambigu, Unique et Canonique. Il est intéressant de noter qu'il existe trois IDs réservés, il s'agit du NULL ID, du World Peer Group ID et du Net Peer Group ID., les deux derniers étant des groupes par défaut. En effet les identifiants peuvent en fait être considéré comme étant le système d'adressage de JXTA.

Peers :

Englobe tous type de composant qui peuvent faire partie d'un réseau. Afin de communiquer entre eux, les peers publient une ou plusieurs adresses réseau à utiliser avec les protocoles JXTA. Chaque adresse est annoncée par un « Peer endpoint », qui identifie l'adresse réseau. Les peer endpoint sont utilisés pour établir une connexion directe « point à point » entre deux peers, d'autre part les connections indirect entre les peers sont effectivement possible grâce au peers intermédiaires (appelés relay ou rendezvouspeers) Un peer a besoin d'un identifiant unique afin de pouvoir le localiser dans le réseau. Cette identification est notamment nécessaire pour l'acheminement d'un message vers son correspondant.

On déduit par cela qu'il existe trois types principaux de peers:

- **Simple Peer**: C'est le type des peers le plus commun, permettant à un utilisateur de recevoir et d'envoyer des services avec d'autre peer dans le réseau, il peut aussi disposé d'un cache locale de stockage ainsi il répond à une requête Discovery par les informations dans son cache.
- **Rendez-vous Peer**: Gère l'index globale de ressource afin de facilité la recherche. Manipule en outre le broadcasting de message et contrôle la propagation et la maintenance de la table de hachage distribuée grâce à la topologie de l'infrastructure des autres peers qu'il maintient à jour. résolution des requêtes de découvertes (localisation).
- **Relay peer** : Pour stocker et expédier des messages entre deux ensembles de peers séparés physiquement en raison des firewalls ou NAT. En effet, un Relay n'est actif que lorsque des peers ne peuvent pas recevoir des raccordements des autres peers.

PeerGroup :

Un PeerGroup est un ensemble de peers qui partage les mêmes services et intérêts, chaque groupe est identifié par un seul identifiant « Peer group ID ». Un Peer appartient à un ou plusieurs groupes, le groupe racine auquel chaque peer appartient par défaut est le NetPeerGroup qui contient divers PeerGroups qui eux-mêmes peuvent contenir des sous-groupes. Tous les peers appartiennent à ce groupe comme ils peuvent rejoindre un autre Peer group à n'importe quel moment.

La création d'un PeerGroup peut être motivée par plusieurs raisons qui sont : le regroupement des peers autour d'intérêts communs, l'établissement d'un accès protégé vers un environnement sécurisé par un mécanisme d'authentification dont le contenu est crypté.

Pipes :

Représente le canal de communication virtuel qui relie les extrémités d'un réseau communiquant et permet à des peers de communiquer malgré les pare feu ou d'autres obstacles, chaque pipe est un raccordement entre les endpoints dans JXTA, permettent à des applications d'envoyer des messages d'un endpoint à un ou plusieurs endpoints (unicast/multicast).

Il y a trois types de pipes dans JXTA: unicast, unicast-sécurisé et propagate. Les pipes d'Unicast sont unidirectionnelles, non fiables et non sécurisés, ainsi la réception du message n'est pas garantie.Les pipes d'Unicast sécurisé sont semblables à des pipes unicast, mais sont fiables. En conclusion, les pipes de propagation sont des pipes multicast mais incertains et non sécurisés, et sont employés pour les messages de multicast à tous les peers reliés dans le pipe

Messages :

JXTA permet deux représentations possibles des messages : binaire ou XML. Cela dépend du protocole de transport utilisé (respectivement TCP/IP, SOAP). Ce message sera envoyer dans un pipe, pour cela le peer doit établir une connexion entre le endpoint source et le endpoint destination. Le message sera encapsulé et envoyé par un output pipe, de l'autre côté du chemin un peer recevra le message par un input pipe.

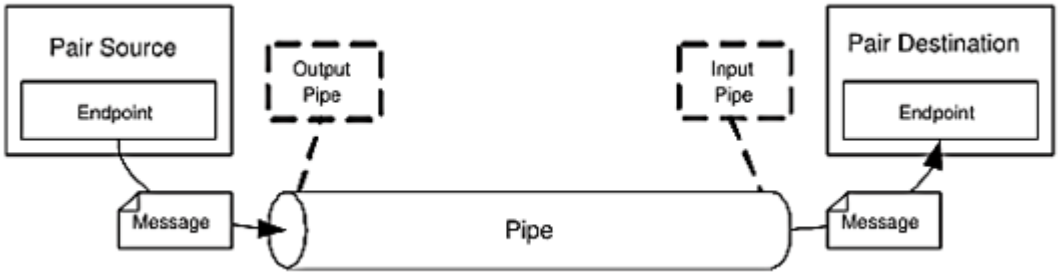


Figure 24. Illustration du modèle de communication JXTA fondé sur des pipes, des endpoints et des messages

Advertissements :

Une annonce (Advertisement) est un document XML qui décrit, nomme et publie un peer, peergroup ou un service du réseau JXTA. Les annonces doivent respecter les normes de codage, les étiquettes et le contenu. Elles sont utilisées pour l'échange d'informations sur ce qui est disponible dans le réseau JXTA.

- ***Peer Advertissement.*** Détient toutes les informations concernant les peers (nom, ID, endpoints..)
- ***PeerGroup Advertissement.*** Détient tous les informations concernant les groupes de peers (nom, l'ID du peer group, la description, la spécification et les paramètres de service)
- ***Pipe Advertissement.*** utilisé par le pipe service pour créer les input pipes et les output pipes Endpoints. contient une symbolique ID, un pipe type (point-to-point, propagate, secure, etc.) et un pipe ID unique.
- ***Endpoint Advertissement***
- ***Module Advertissement***
- ***Content Advertissement***

6. Les Peer Group services :

Les protocoles sont implémentés par des services essentiels de base (*Core services*) définit par JXTA fournit par les peer group

- ***LeDiscovery service:*** recherche/publication d'annonce locale/distante, vidange du cache local des annonces
- ***Le Rendezvous***: placement des annonces.
- ***Le Resolverservice***: formuler requêtes et réponses.
- ***Le Pipeservice***: canal de communication. canal virtuel construit entre peers, unidirectionnel et asynchrone, identifié par un PipeID. Il utilise une série d'endpoints pour construire la route (vérifiée périodiquement)
- ***Le EndPointservice*** : interface réseau. Liaison abstraite non fiable et unidirectionnelle entre deux peers identifiés par des PeerID. Véhicule des messages JXTA.

Des services supplémentaires peuvent être développés.

7. Les protocoles JXTA:

JXTA propose une base de six protocoles dans lesquels la communication se fait par des messages XML, ces protocoles décrivent Comment signaler l'existence d'une ressource, Comment rechercher de l'information sur ce réseau (type de requête), Comment mettre en relation producteurs et fournisseurs d'information, Comment établir un canal de communication entre plusieurs peers.

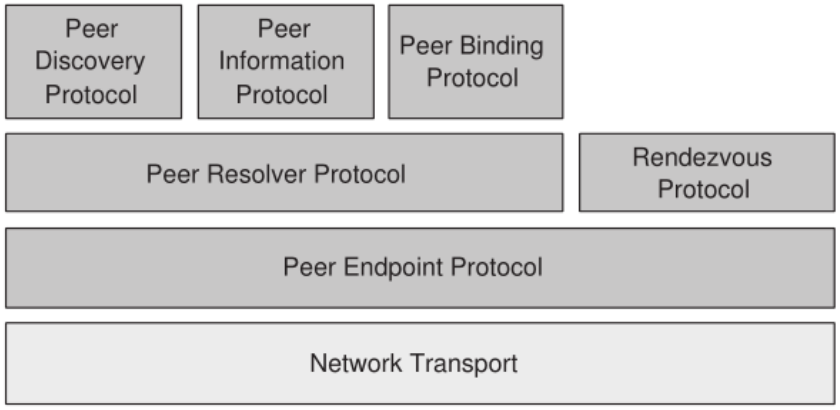


Figure 25. Les protocoles JXTA

- **Endpoint Routing Protocol (ERP):**Ce protocole permet à un peer de trouver et établir un chemin jusqu'à un peer distant lors de l'envoi de messages, sachant que ce chemin peut se

composer de relais intermédiaires. comme il fait appel au RelayService pour les peers inatteignables (par les firewall)

- **Rendez Vous Protocol (RVP):**Ce protocole gère les relations entre les peers simples et les peers de rendez-vous. Afin de propager les requêtes dans un PeerGroupe ou à tous les peers simples connectés à des peers de rendez-vous qui sont eux-mêmes connectés à d'autres peers de rendez-vous. Cette interconnexion des peers de rendez-vous forme une table de hachage distribuée appelée SRDI

  – utilisé par le Peer Resolver Protocol pour propager les messages

- **Peer Resolver Protocol (PRP):**C'est un protocole à travers lequel les peers peuvent émettre des requêtes et recevoir des réponses. Les requêtes sont génériques et peuvent être utilisés par différents protocoles de niveau supérieur tel que PDP et PIP. La portée des recherches est limitée au groupe auquel le peer appartient.

- **Peer Discovery Protocol (PDP):** Il offre la possibilité à des peers de publier et rechercher des ressources chez d'autres peers. Par le biais de ce protocole, les peers peuvent découvrir et localiser les ressources et les éléments présents dans une communauté (groupe), Il permet la propagation par IP multicast dans un LAN (implémentation Java) et utilise rendezvous/relay dans un WAN (RVP) ainsi que le TTL pour prévenir le flooding

- **Peer Information Protocol (PIP) :** Ce protocole, mène vers la gestion, permet à un peer d'obtenir des informations sur d'autres peers de la communauté

- **Pipe Binding Protocol (PBP) :**Ce dernier protocole est utilisé dans le cadre de la création des pipes pour effectuer le lien entre deux ou plusieurs Endpoint qui sont les extrémités (input/output(s)) de la connexion. Il utilise le PRP

Conclusion

JXTA met à disposition un ensemble de protocoles au près des développeurs des systèmes peer-to-peer, ces protocoles offrent des mécanismes nécessaires dans le but de réaliser la découverte, l'organisation, la surveillance et les communications entre les peers.

En conclusion, JXTA est une technologie encore jeune en constante amélioration et de nombreuses applications sont encore en cours de développement. Il existe des services qui sont en cours de progression comme le shell et la sécurité.

Chapitre 4 Conception et réalisation de l'application

Introduction :

Dans ce chapitre, nous allons présenter notre application qui est fondée sur le cryptage d'un contenu partagé dans un réseau P2P JXTA, nous parlerons de l'objectif de cette application et ses caractéristiques, nous présenterons par la suite les outils que nous avons utilisés pour commencer la réalisation de notre projet. Enfin nous nous intéresserons au fonctionnement de l'application par des diagrammes, des captures d'écran, des classes utilisés et des interfaces graphiques.

1. Problématique

Dans un réseau JXTA la sécurité joue un rôle primordial dans l'efficacité d'une application, plus précisément lors des échanges d'information qui ce fait à travers les pipes.

Dans notre application, la communication est effectuer par une diffusion multicast entre les peers connectés dans le réseau, les messages sont transférés à travers les pipes de propagation, ceux-ci sont incertains et non sécurisés chose qui met en danger la confidentialité des données échangées.

2. Objectifs

Notre application a pour objectif de maintenir des communications privées dans un réseau P2P JXTA, c'est-à-dire de pouvoir cacher les informations lors de la transaction des messages. Raison pour laquelle on a fait appel à la cryptographie qui a des objectifs communs tels que la protection des communications pour but d'assurer la sécurité.

3. Caractéristiques de l'application :

- ✓ Tout peer peut voir les autres peers connectés.
- ✓ La messagerie est destinée à tous les membres connectés dans le réseau JXTA local.
- ✓ La communication est sécurisée par trois algorithmes (RC4, RSA et BlowFish).
- ✓ La discussion et l'envoie des fichiers (.txt) est une sorte de forum, ou chaque Peer doit être admis par l'un des membres du groupe.
- ✓ Utilisation de plateforme JXTA

✓ Utilisation de **JXTA Security Toolkit** pour crypter et décrypter le contenu (RC4, RSA)

✓ Utilisation de **L'APIJCE (Java Cryptography Extension)** sert de base pour l'implémentation de certaines fonctionnalités de cryptage et de décryptage (Blow Fish).

4. Outils de travail :

JAVA :

Est un langage de programmation orienté objet, développé au milieu des années 1990 par James A.Gosling, un ancien informaticien avec Sun Microsystems qui produit des logiciels pour plusieurs plates-formes. Java fonctionne sur une machine virtuelle, il permet donc facilement de développer les applications et de la déployer. Java présente un ensemble de classes standards (bibliothèque standard) pour tous les domaines d'application informatique existants.

NetBeans :

Est un environnement de développement intégré (IDE) créé par Sun Microsystems. NetBeans est sous licence OpenSource, il permet de développer en Java, Ruby, C/C++ ou même PHP et déployer rapidement les applications Web, applications graphiques Swing, des Applets, etc... Netbeans est une plateforme facile et simple à utiliser. Il permet de créer et modifier vos propres applications Java(IDE 7.3.1 / IDE 8.1).

Le framework JCE (Java Cryptography Extension) :

Java Cryptography Extension (JCE) est une interface de programmation Applicative (API) qui fournit un fort cryptage pour la mise en œuvre des fonctions de sécurité dans Java. Le framework JCE (Java Cryptography Extension) sert de base pour l'implémentation de certaines fonctionnalités telles que :

- ✓ chiffrements symétrique.et asymetrique
- ✓ chiffrements à flux et à bloc
- ✓ génération, stockage et recherche des clés
- ✓ Signatures numériques

JCE est intégré à Java SE à partir de la version 1.4. Pour utiliser un service cryptographique de JCE, il faut obligatoirement fournir une implémentation de référence nommée SunJCE. Les

classes de l'API JCE sont contenues dans le package **javax.crypto** qui contient plusieurs classes et interfaces. [28]

Plateforme JXTA :

Est une plateforme réseau Peer to Peer open-source développée par Sun qui spécifie un ensemble de protocoles de collaboration et de partage de contenu et qui permet la communication avec d'autres plateformes JXTA dans le même réseau, ainsi que la création et le déploiement d'applications peer-to-peer (P2P). La communauté JXTA a développé différentes APIs propres à différents langages qui seront ajoutés lors de la création du nouveau projet. Parmi les APIs utilisé dans notre projet il y a jxta-jxse-2.5, L'implémentation est divisée en deux parties : JSE Standard JXTA API et JSE JXTA.

JXTA Security Toolkit :

La JXTA Security Toolkit (jxta.security) fournit des mécanismes pour la cryptographie comme : définir des clés secrètes RSA, effectuer des chiffrements RC4, créer des messages hachés au format SHA-1 et MD5, offre la possibilité de signer numériquement des données. Dans le reste de ce chapitre, nous décrierons les principaux composants de JXTA Security Toolkit que nous avons utilisé.

Adobe Photoshop :

Photoshop est un logiciel de retouche, de traitement et de dessin assisté par ordinateur, écrit en langage C++ sous les environnements Windows et Mac OS X, existe en multiples langues. Édité par Adobe, il est principalement utilisé pour le traitement de photographies numériques et il travaille essentiellement sur les images matricielles.

En addition d'un ***point d'accès mobile*** et ***trois ordinateur portable (i3 et i5 sous windows 10, RAM 6G …)***

5. Fonctionnement de notre application :

5.1. Vue générale de la démarche de l'application :

a. *Le cas de deux Peer :*

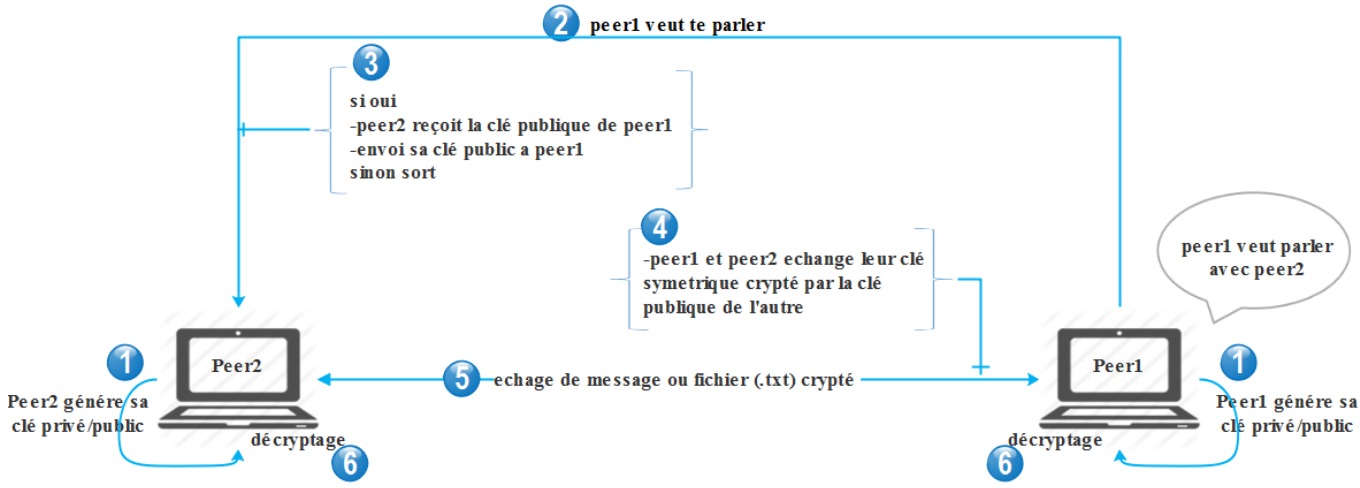


Figure 26. Fonctionnement de l'application entre 2 Peer

b. *Le cas de trois Peer :*

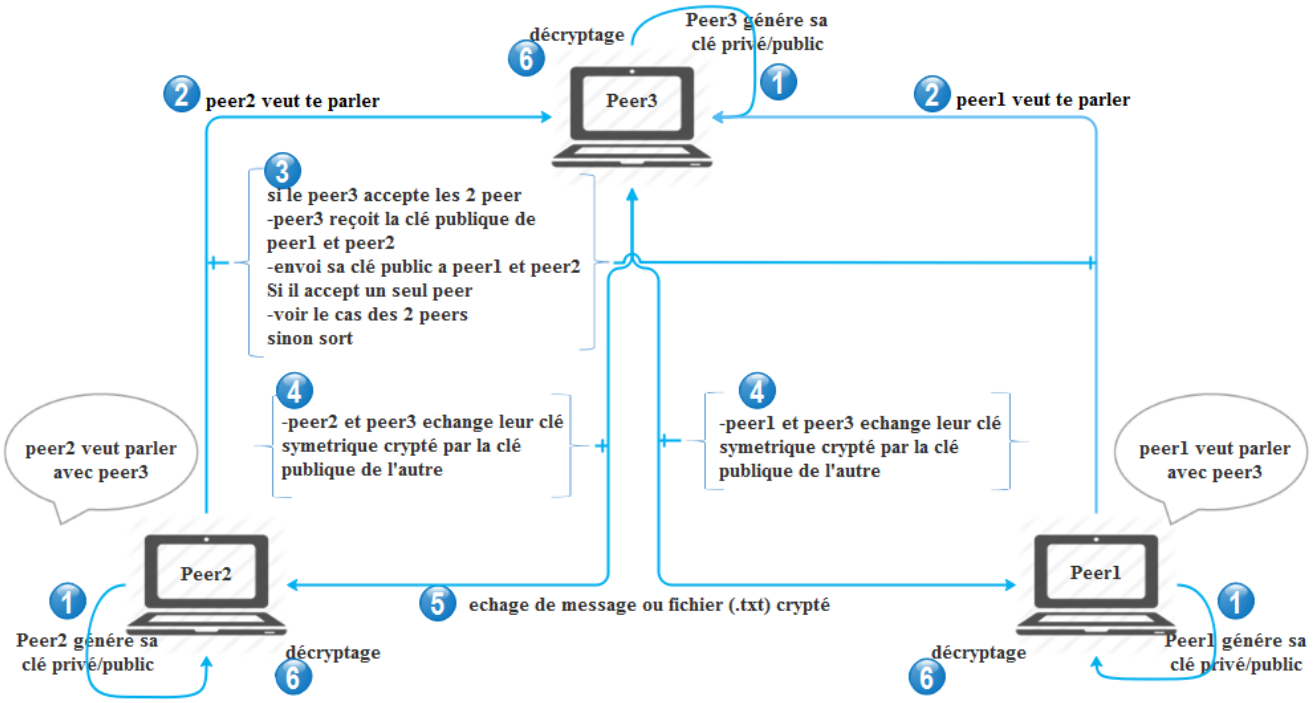


Figure 27. Fonctionnement de l'application entre 3 Peer

5.2. Les Diagrammes :

Pour bien comprendre le fonctionnement de l'application et les objectifs correspondant nous avons utilisé un langage **UML (Unified Modeling Language)**. UML est une notation qui permet de décrire une représentation graphique, on l'utilise pour la modélisation et la définition d'un ensemble d'aspects fonctionnels et relationnels de l'application, en utilisant des éléments appelés diagrammes UML. Voici quelques diagrammes utilisés :

- Dans les deux diagrammes on suppose que le Peer1 veut parler avec le Peer2

a. Diagramme de cas d'utilisation :

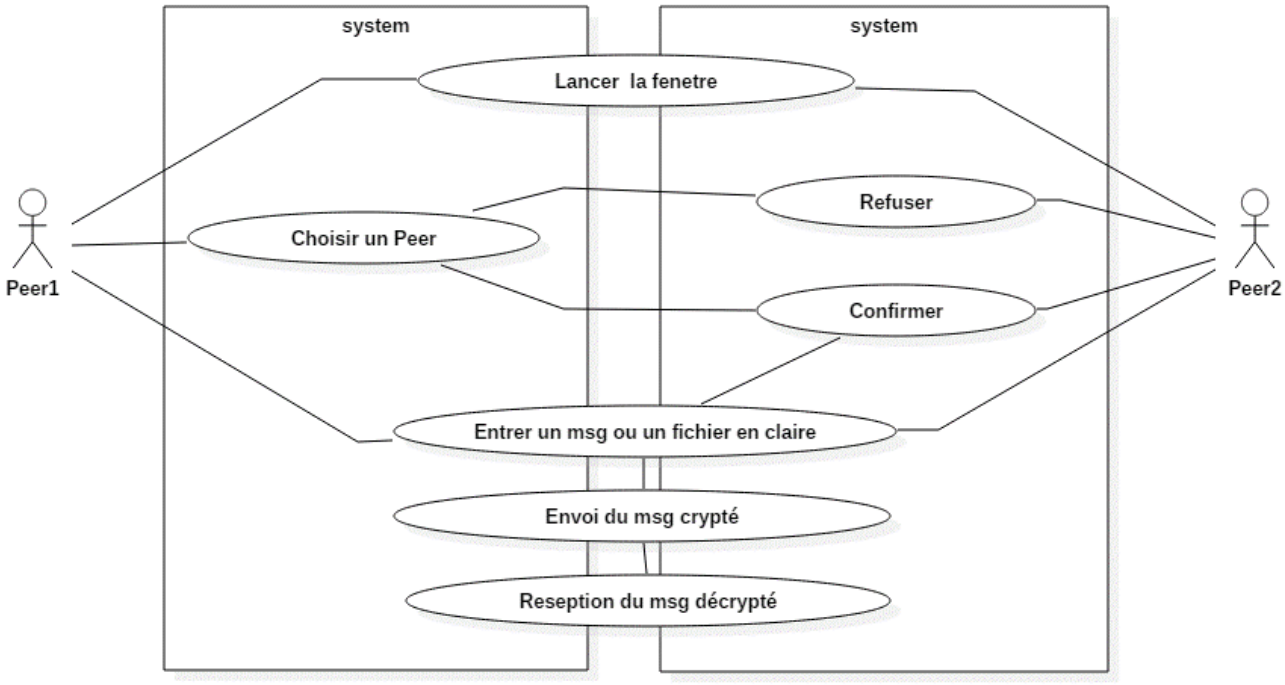


Figure 28. Diagramme de cas d'utilisation

b. Diagramme de classe

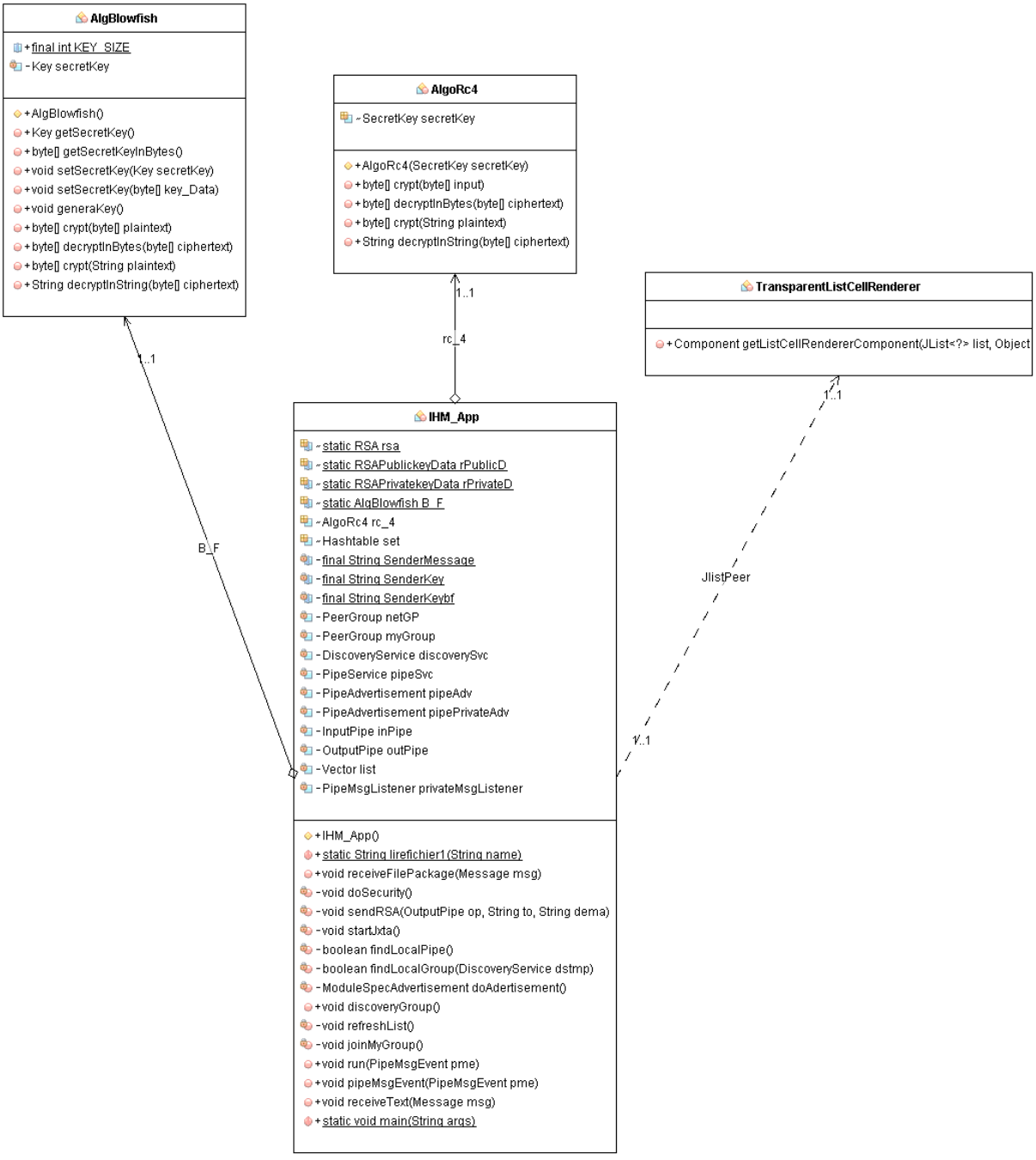


Figure 29. Diagramme de class

c. Diagramme de séquence du cas d'utilisation :

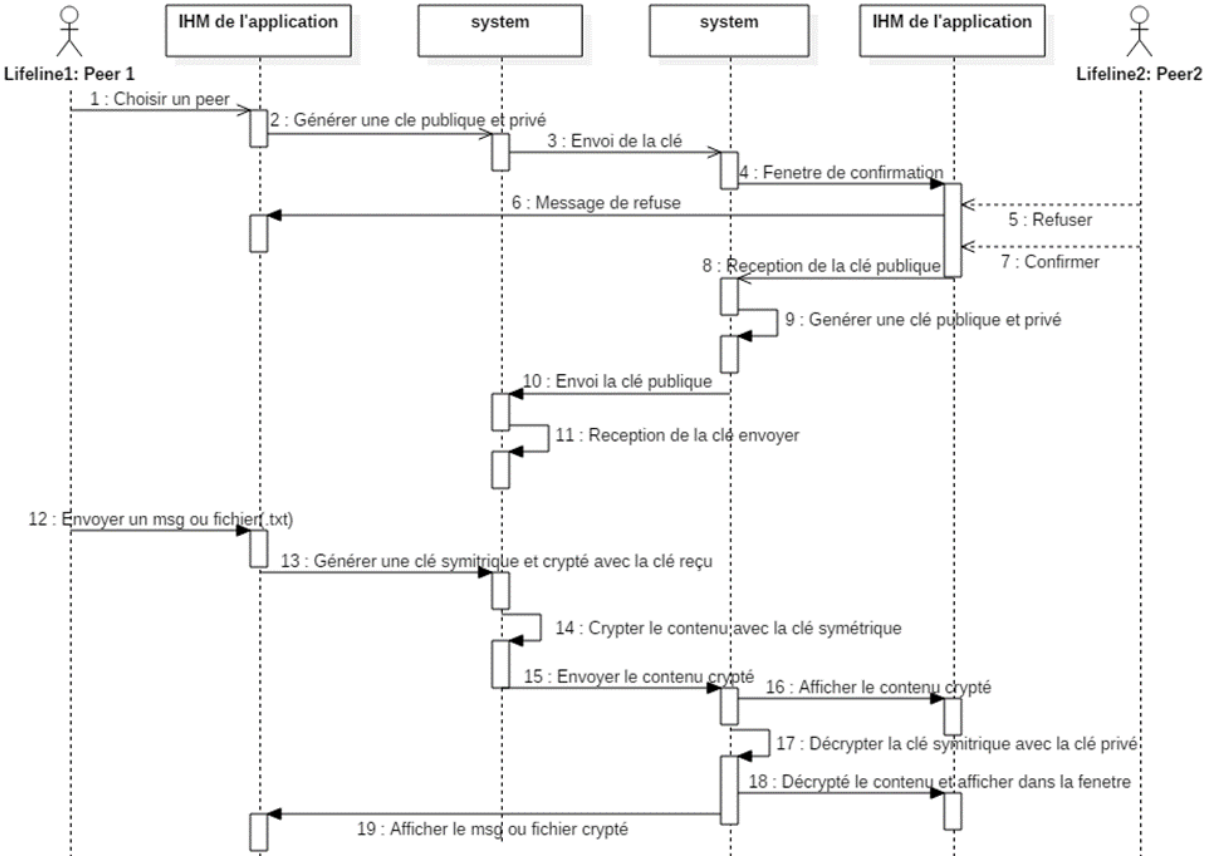


Figure 30. Diagramme de séquence

5.3. Fonctionnement :

Une fois la fenêtre ouverte :

Etape 1 :

La plateforme JXTA démarre selon une configuration LAN, Pendant le démarrage tous les peers créent, lancent et rejoignent le groupe par défaut dans JXTA à l'aide de la fonction « **startJXTA ()** ».

Etape 2 :

Rejoindre un groupe de peers est habituellement la prochaine étape après le démarrage, il est exécuté en chargeant une annonce du groupe de peers, si le groupe de peer n'existe pas donc il faut créer un nouveau groupe à l'aide de « **ModuleImplAdvertisement »**, celui-ci contient tous les services de base que chaque groupe se doit d'implémenter.

Si un peer rejoint le groupe pour sa première fois, il doit découvrir l'annonce du groupe des peers du réseau à l'aide de « **Discovery service »**, ces annonces sont sous forme d'un document XML appelés « **Adverstisement ».**

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xml:space="preserve"
xmlns:jxta="http://jxta.org">
<Id>
urn:jxta:uuid-04B8173E42A445F0AF6BBB486CEA53D1AAC4571E4ED94C16B6
17A66648B798CC04</Id><Type>JxtaUnicast</Type><Name>
Group_SecurisÃ©:PipePrivateAdv:GroupSecurisÃ©</Name><Desc>
peer1</Desc></jxta:PipeAdvertisement>
```

Figure 31. Pipe ADV

Ensuite le peer publie cet « **Adverstisement »** afin que d'autre peers puissent créer un Pipe en sortie permettant d'envoyer des messages au tube correspondant en entrée, cette publication se fait pour deux types local et à distance.

Etape 3 :

Création d'un Pipe d'entré qui permet d'établir une communication entre peers et d'attendre l'arrivée des messages, le Pipe est créé grâce à la méthode « **createInputPipe () »** du Pipe Service. Celle-ci prend deux paramètres en entrée, il s'agit de l'**advertisement** du Pipe et du « **PipeMsgListener »** associé qui recevra les évènements liés au Pipe à l'aide de la méthode « **pipeMsgEvent() »** qui sera automatiquement invoquée pour chaque évènement de type « **pipeMsgEvent »** correspondant à la réception d'un message

Etape 4 :

Pour que le peer puisse communiquer ses informations, ses messages et ses requêtes il a besoins d'effectuer une recherche de la publicité sur les groupe de peer à l'aide de la méthode

« **discoveryGroup() »** une fois la publicité est trouvée (localement ou à distance) ,le peer crée un Pipe en sortie et envoie ses requêtes. La méthode de création s'utilise de la même façon que pour un Pipe en entrée, il s'agit de « **createOutputPipe() »** du Pipe Service .

Etape 5 :

Chaque peer crée une paire de clé RSA grâce à la méthode « **doSecurity() »** qui permet de créer la clé RSA à l'aide de la classe « **KeyBuilder »** qui existe dans le package « **JXTA Secyrity Toolkit »** et qui spécifie la taille de la clé ,ensuite l'appel de la méthode « **setPublicKey() »** pour calculer la clé publique en fonction de la taille spécifiée par le « **KeyBuilder »** et l'appel de la méthode « **setPrivateKey() »** qui génère la clé privée en fonction de la clé publique.

Etape 6 :

Après avoir cliqué dans « la liste des peers connecté » un message contenant une clé est envoyé au peer sélectionné (étape 4) et la méthode « **sendRSA(**OutputPipe op, String to, String dema**(avec dema='req')) »** invoque automatiquement un « **pipeMsgEvent »** qui va afficher une fenêtre de confirmation. Si le peer sélectionné accepte par « oui » il passe à l'étape 7 sinon il sort. La méthode « **sendRSA »** envoi la clé publique générée par l'étape 5 .

Etape 7 :

Après avoir sélectionné et accepté le peer choisi, ce dernier va recevoir une clé publique envoyer par le peer qui a effectué le choix à l'aide de la méthode « **run**() », et va lui envoyer sa clé publique avec « **dema= 'rep' »** .

Etape8 :

Echange de contenu :

Si le type du contenu est un message, alors le peer va créer une clé RC_4 à l'aide de la classe « **KeyBuilder »** et l'initialiser**.** Le peer va crypter cette clé avec la clé publique de RSA qui est envoyée par le peer sélectionné à l'aide de la méthode « **Algorithm »** de RSA, ensuite le peer va crypter le message avec la clé **RC_4** grâce à la méthode « **crypt(byte[] input)** »de la classe **AlgoRC_4** ». Après avoir cliqué sur le bouton « MSG » un couple de message (la clé RC_4 crypté, le message crypté) va être envoyé grâce à l'étape 4.

Si le type de contenu est un fichier texte alors le peer va générer une clé Blowfish à l'aide de la méthode « **generaKey**() » de la classe **AlgoBlowfish**. Le peer va crypter cette clé avec la clé publique de RSA qui est envoyée par le peer sélectionné à l'aide de la méthode « **Algorithm »** de RSA, ensuite le peer va crypter le fichier texte avec la clé Blowfish grâce à la méthode « **crypt(byte[] input**)» de la classe **AlgoBlowfish** Après avoir cliqué sur le bouton « File » un couple de message (la clé Blowfish crypté, le fichier texte crypté) va être envoyé grâce l'étape 4 .

Etape 9 :

A l'aide d'étape 3 le peer va recevoir un contenu et vérifier si le contenu lui est destiné :

Si le contenu est un message il va le recevoir à l'aide de fonction « **receiveText(msg) »** qui va décrypter la clé RC_4 avec sa clé privé de RSA à l'aide de la méthode « **Algorithm »** de RSA, et ensuite décrypter le message avec la clé RC_4 décryptée avec la méthode « **decryptInString(byte[] ciphertext**) » de la classe AlgoRC_4 .

Si le contenu est un fichier texte, il va le recevoir à l'aide de la fonction « **receiveFilePackage(msg) »** qui va décrypter la clé Blowfish avec sa clé privé de RSA à l'aide de la méthode « **Algorithm »** de RSA, et ensuite décrypter le fichier texte avec la clé Blowfish décryptée grâce à la méthode « **decryptInBytes(byte[] ciphertext) »** de la classe **AlgoBlowfish**.

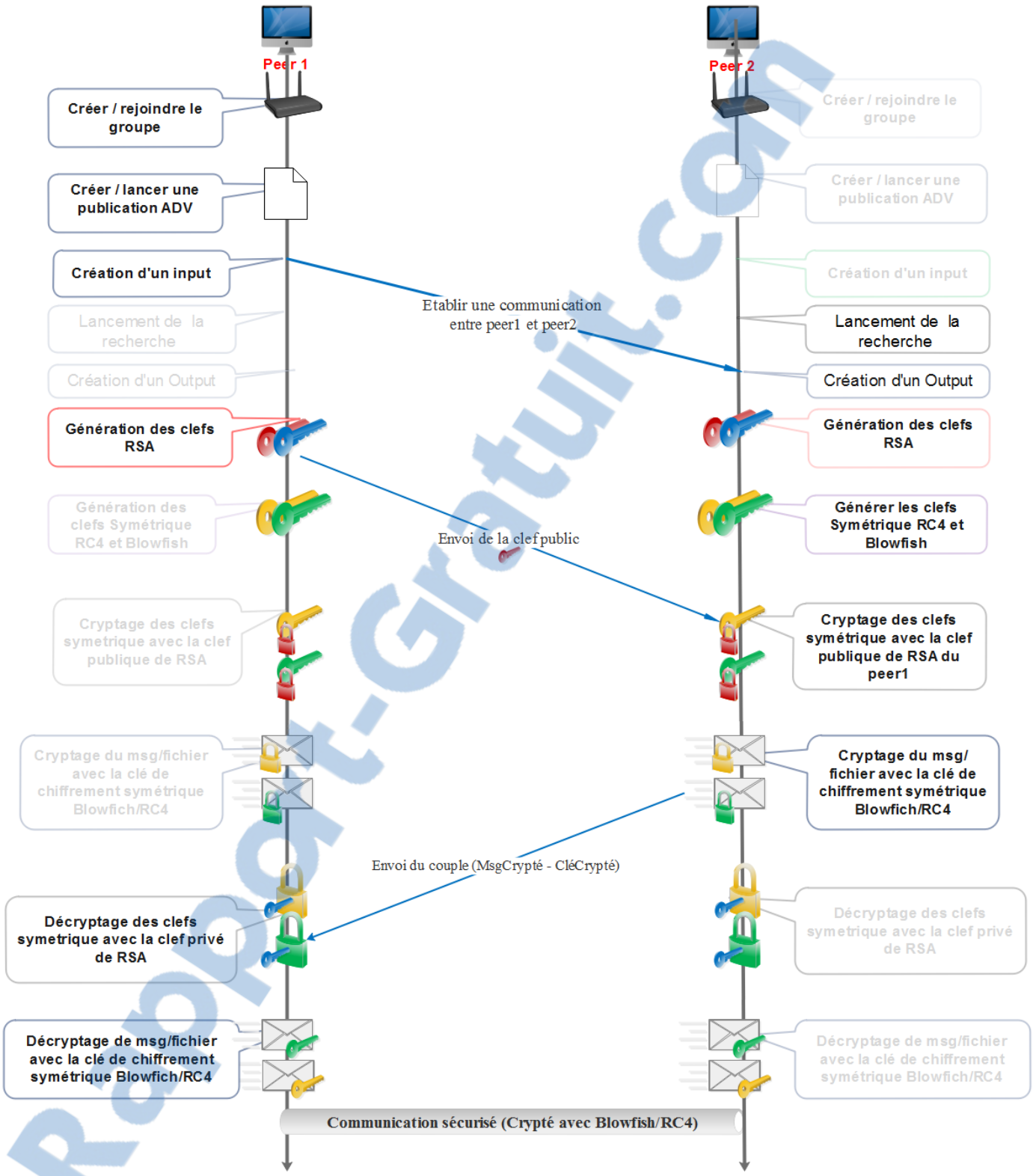


Figure 32. Fonctionnement détaillé de l'application

6. Interface graphique

- Fenêtre 1er :

Cette première capture présente la fenêtre initiale qui s'ouvre lors du lancement de l'application, elle comporte le logo et le concepte de notre projet .

Figure 33. Interface initiale

- Interface principale :

C’est la fenêtre la plus essentielle sur la quelle est fondée l’application, qui inclut tous les composants fonctionnels nécessaires. Elle est utilisée pour pouvoir établir une discution avec un autre peer ou dans un groupe.

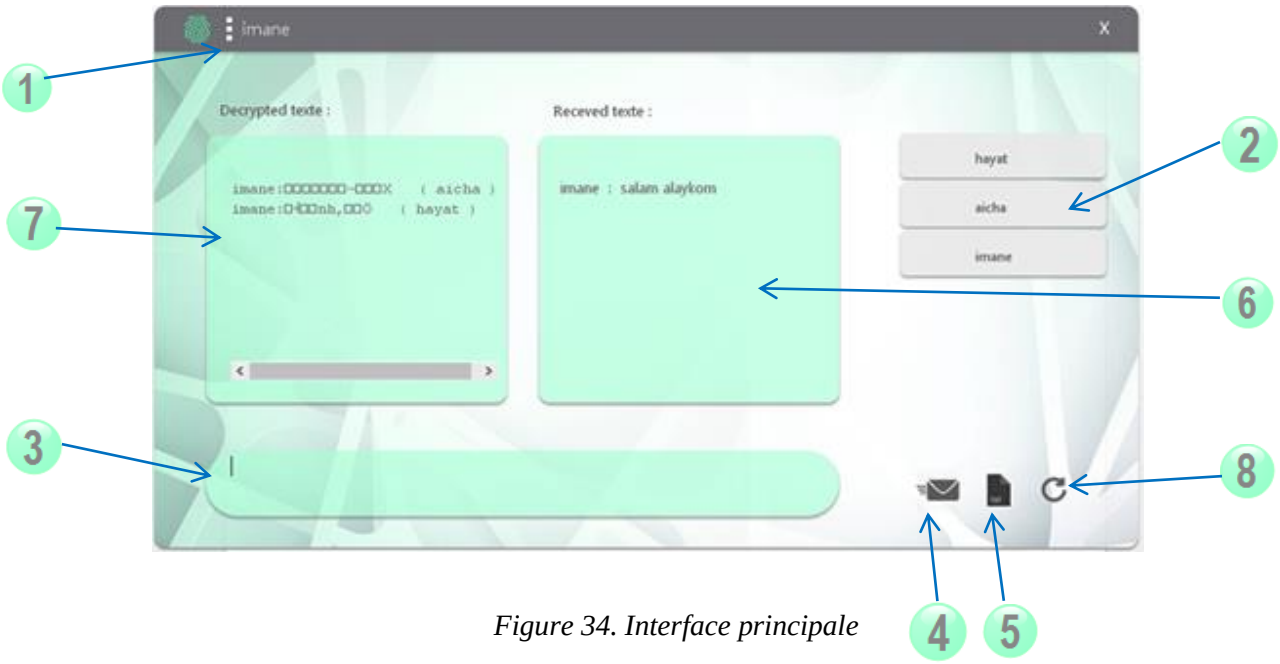


Figure 34. Interface principale

Déscription des composants :

1- le nom du peer connecté
2- la liste des peers connectés
3- zone d'écriture de message
4- bouton d'envoi des messages
5- bouton d'envoi des fichiers txt
6- zone de message en clair
7- zone de message crypté
8- bouton d'actualisation des peers connecté

- Fenêtre de confirmation

Cette fenêtre vous s'affiche lorsqu'un peer veut communiquer avec vous. C'est-à-dire lorsque vous recevez une invitation pour rejoindre un groupe de discussion.

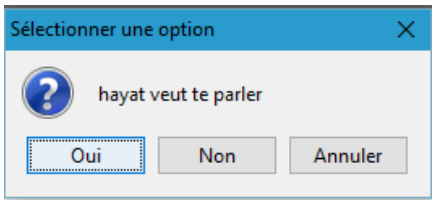


Figure 35. Fenêtre de confirmation

Cette fenêtre s'affichera après avoir cliqué sur le bouton « file » vu précédemment dans la « Fenêtre principal », celle-ci vous permettra de sélectionné le fichier texte que vous voulez envoyer.

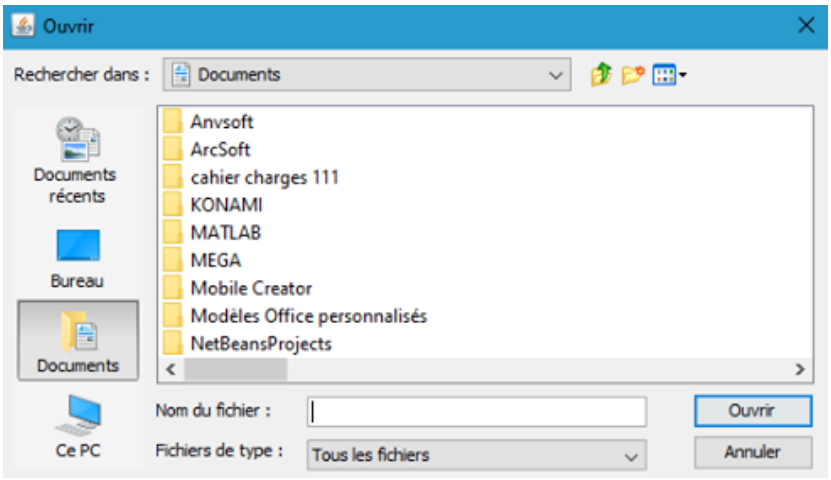


Figure 36. Fenêtre pour la selection de fichier

Conclusion et perspective :

Dans ce chapitre, nous avons présenté notre application et aborder les différentes étapes du fonctionnement de notre projet de fin d'étude. Notre travail s'est appuyé essentiellement sur la combinaison entre les algorithmes de chiffrement à clefs publiques-privées et les algorithmes à clefs secrètes dans la plateforme JXTA, l'application que nous avons développée répond nettement à l'objectif initialement posé, nous souhaitons par la suite l'améliorée en rendons possible le partage de tous type de fichiers ainsi que le partage de vidéos et d'image. Comme nous aimerions régler l'inconvénient que pose la configuration de JXTA qui ne permet pas de changer le nom du Peer après la première exécution, et acquérir par cela la satisfaction d'un grand nombre d'utilisateur.

Conclusion général

L'aspect principal de ce projet a été la sécurisation des messages par la cryptographie symétrique dans une application P2P décentralisé en utilisant la technologie JXTA. De ce fait pour atteindre cet objectif posé il a fallu combiner à la fois le système pair à pair (P2P), la plateforme JXTA ainsi que la cryptographie symétrique et asymétrique, et la programmation en langage java.

Nous avons donc vu dans ce projet les notions de base du Peer-to-Peer et les types de cette architecture et nous avons compris son utilité pour la transmission des messages instantanée et le partage de fichiers. Nous nous somme introduit à la cryptographie moderne en se focalisant sur les algorithmes de cryptage adoptés dans notre application, sans oublier de présenter les concepts de la plateforme JXTA et ses protocoles, sachant qu'il existe dans le noyau de JXTA une bibliothèque de cryptage simple, prédéfinie et prenant en charge des algorithmes de cryptage symétriques (RC4) et Des algorithmes cryptographiques asymétriques (RSA).

Notre travail peut aider les nouveaux développeurs des applications P2P qui s'appuient sur JXTA à se familiariser avec ce nouveau concept.

Le projet a abouti au but fixé initialement, malgré les difficultés qu'on a eu pour se documenter sur JXTA qui sont rares et nous a fait perdre du temps, sans parler des restrictions à respecter pendant la programmation pour se limiter au protocoles et service propre à JXTA. Toutefois, on aimerait intégrer plus d'option de partage et ajouter d'autres sous services afin d'étendre cette infrastructure au sein d'une large communauté.

Acronymes

Acronymes	Définitions
ACM	Association for Computing Machinery
AES	Advanced Encryption Standard
CMS	content management service
CPU	central processing unit
DES	Data Encryption Standard
DHT	Distributed Hash Table
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
GSM	Global System for Mobile Communications
IBM	International Business Machines Corporation
IDEA	International Data Encryption Algorithm
KSA	Key Schedule Algorithm
LAN	Local Area Network
LFSR	Linear Feedback Shift Register
MD5	Message Digest 5
MOD	modulo
NBS4	National Bureau of Standards
NIST	National Institute of Standards and Technology
PAM	module d'authentification enfichable
PDA	personal digital assistant
PGP	Pretty Good Protection
PKP	Public Key Partners
RC2	Ron's Code 2
RC4	Ron's Code 4
RC5	Ron's Code 5

RSA	Rivest, Shamir, & Adleman
S-box	Substitution box
SEAL	Software-Optimized Encryption Algorithm
SOAP	Simple Object Access Protocol
SRDI	Shared Resource Distributed Index
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TEA	Tiny Encryption Algorithm
TLS	Transport Layer Security
TTL	Time to Laugh
UUID	universally unique identifier
WAN	Wide Area Network
XML	Extensible Markup Language
XOR	Exclusive or (exclusive disjunction)

Références

[1] “NGNEPI NKANYOU GIDE RENAUD, diplôme de fin d’étude, Réalisation d’un client de chat de 3ème génération 2008/2009.”

[2] Hameau Gaétan, Miclard Yann les protocoles peer to peer 2006.

[3] Patrick Marlier, « Securité du Peer to Peer», 2007.

[4] “Nathalie BUDAN Benoit TEDESCHI Stéphane VAUBOURG Les réseaux peer-to-peer 2003,” .

[5] Vinton G. Cerf et Robert E. Kahn, un protocole pour le réseau de paquets intercommunication , 5 (1974).

[6] Ronda Hauben. « De l’ARPANET à l’Internet ». TCP Digest (UUCP) . Récupérée 2007-07-05.

[7] “THUAUX Anthony SOUSA LOPES Eric Travail d’Etude 2004 - Licence Informatique Université Nice Sophia-Antipolis Sujet : L’échange de fichiers dans les réseaux Pair-à-Pair. 16 Juin 2004.”

[8] Hyunggon Park , RafitIzhakRatzin Peer-to-Peer Networks – Protocols, Cooperation and Competition.

[9] Stoica I; Morris R; Karger D; Kaashoek M.F; Balakrishnan H. (2001). (PDF). ACM SIGCOMM Computer Communication Review 31 “Chord: A scalable peer-to-peer lookup service”. .

[10] “Guillaume Doyen. Supervision des réseaux et services pair à pair. Réseaux et télécommunications [cs.NI]. Université Henri Poincaré- Nancy I, 2005. Français.” .

[11] “http://www.apprendre-en-ligne.net/crypto/menu/index.html.” . [Accessed: 20-Apr-2017]

[12] “https://www.apprendre-en-ligne.net/crypto/lexique.html.” [Accessed: 20-febr-2017]

[13] Richard Brisson and François Théberge, “UN APERÇU DE l’histoire de la cryptologie.” .

[14] “http://math.univ-lyon1.fr/~roblot/resources/masterpro_chapitre_1.pdf.” [Accessed: 25-march-2017]

[15] Jonathan BLANC and Adrien DE GEORGES, “TECHNIQUES DE CRYPTOGRAPHIE.” Licence Informatique, 2004-2003.

[16] “http://ram-0000.developpez.com/tutoriels/cryptographie/.”[Accessed: 28-marsh-2017]

[17] W. Diffie et M. E. Hellman, “New Directions in Cryptography.” IEEE Transactions on Information Theory, Nov-1976.

[18] Renaud Dumont, “Cryptographie et Sécurité informatique.” Université de Liège Faculté des Sciences Appliquées, 2010-2009.

[19] Kavita Tewari and Anushree Ashok Shenoy, “Novel Method to Strengthen RC4 Algorithm.” ISSN, 2013.

[20] B. Schneier, Ed., Fast software encryption: 7th international workshop, FSE 2000, New York, NY, USA, April 10-12, 2000: proceedings. Berlin ; New York: Springer, 2001.

[21] Jean philippe gaulier, “analyse des algorithmes finalistes concurant pour le futur standard AES.” conservatoire national des arts et métiers centre régional associé de limoge, france.

[22] “https://www.schneier.com/academic/blowfish/.”[Accessed: 28-apr-2017]

[23] Saikumar Manku and K. Vasanth, “BLOWFISH ENCRYPTION ALGORITHM FOR INFORMATION SECURITY.” ARPN Journal of Engineering and Applied Sciences, 2015-2006.

[24]“https://www.schneier.com/academic/archives/1994/09/description_of_a_new.html.” [Accessed: 05-Apr-2017]

[25] Jawahar Thakur and Nagesh Kumar, “, DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis.” International Journal of Emerging Technology and Advanced Engineering, Dec-2011.

[26] B. J. Wilson, JXTA, 1st ed. Indianapolis, Ind: New Riders, 2002.

[27] “Project JXTA: A Technology Overview Li Gong Sun Microsystems, Inc. 901 San Antonio Road Palo Alto, CA 94303 USA, April 25, 2001.”

[28] “http://jmdoudoux.developpez.com/cours/developpons/java/chap-jce.php.” [Accessed: 07-Apr-2017]

Résumé

Le projet JXTA est une infrastructure du développement d'application peer-to-peer, JXTA est composé de 6 protocoles standards et de réalisations multi-langage. Un réseau des peers JXTA est un recouvrement complexe, construit sur le réseau physique, avec son propre arrangement, d'identification et de cheminement.

Le but de notre étude est d'assurer la sécurisation des messages par cryptographie symétrique dans une application P2P décentralisé utilisant la technologie JXTA et programmer en langage java. Ceci pour répondre au besoin fondamentale des réseaux peer-to-peer concernant la sécurité de transmission des données entre les peers.

Notre effort se concentre sur la plateforme JXTA, pour laquelle nous avons utilisé trois entité principal pour l'émission des données : Endpoint , InputPipe ,OutputPipe, et se concentre principalement sur la sécurisation des données circulant en clair dans les tubes de transmission. Pour cela nous avons combiné plusieurs algorithmes de chiffrement afin d'assurer une sécurité et une protection de données qui circulent dans notre réseau.

MOTS-CLÉS : réseau, P2P, sécurité, JXTA, cryptographie

Abstract

The JXTA project is an infrastructure for peer-to-peer application development, JXTA is composed of 6 standard protocols and multi-language implementations. A peer network JXTA is a complex overlay, built on the physical network, with its own arrangement, identification and routing.

The purpose of our study is to ensure the securing of messages by symmetric cryptography in a decentralized P2P application using JXTA technology, programed in Java language. This is as respond to the fundamental need of peer-to-peer networks for the security of data transmission between peers.

Our effort focuses on the JXTA platform, for which we have used three main entity for data transmission: Endpoint, InputPipe, OutputPipe, and focuses mainly on securing the data flow in clear in the transmission tubes, for which we have combined several encryption algorithms to ensure data security and protection in our network.

KEYWORDS: network, P2P, security, JXTA, cryptograph

ملخص

المشروع JXTA هو مشروع لتطويرالبنية التحتية لتطبيقات "peer-to-peer" من قرين الى قرين. يحتوي JXTA على 6 بروتوكولات والمشاريع متعددة اللغات. شبكة الأقران ترتكز على الشبكات المادية مع الترتيب لخاص بها وكذا نظام للتعريف والتتبع.

والهدف من دراستنا هو ضمان آمن الرسائل بواسطة التشفير المتماثل في تطبيق لامركزي P2P باستخدام تكنولوجيا JXTA والبرمجة بلغة java. هذا لتلبية الاحتياجات الأساسية لشبكات "peer-to-peer" لنقل آمن للبيانات بين أقرانه.

يركز جهودنا على منصةJXTA ، والتي استخدمنا منها ثلاثة عناصررئيسية لنقل البيانات: OutputPipe ,InputPipe , Endpoint، ويركز في المقام الأول على تأمين البيانات الواضحة المتدفقة في أنابيب نقل. لهذا جمعنا بين عدة خوارزميات التشفير لضمان أمن وحماية البيانات المتداولة في شبكتنا.

الكلمات الرئيسية: الشبكة، peer-to-peer ، الأمن، JXTA، الترميز