TABLE DES MATIERES

LISTE DES FIGURES

TABLE DES ABREVIATIONS

AES	Advanced Encryption Standard
ARP	Address Resolution Protocol
ASP	Application Service Provider
BCL	Base Class Library
BNF	Backus-Naur Form
CBC	Cipher Block Chaining
CFB	Cipher Feed Back
CGI	Common Gateway Interface
CIL	Common Intermediate Language
CLR	Common Language Runtime
CTS	Common Type System
DDR	Data Delivery Rate
DES	Data Encryption Standard
DLL	Dynamic Link Library
DMZ	Demilitarized Zone
DNS	Domain Name System
DTMF	Dual Tone Multi-Frequency
ECB	Electronic Code Book
FCL	Framework Class Library
FTP	File Transfer Protocol
FXO	Foreing eXchange Office
FXS	Foreing eXange Station
GDI	Graphics Device Interface

GNU	Gnu's Not Unix
GPL	General Public Licence
GSM	Global System for Mobile
HDD	Hard Disk Drive
HTML	HyperText Markup Langage
HTTP	HyperText Transfer Protocol
IAX	Inter-AsteriskeXchange
ICMP	Internet Control and error Message Protocol
ID	Identity
IDE	Integrated Drive Electronics
IETF	Internet Engineering Task Force (normes internet)
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPsec	Internet Protocol Security
ISO	International Standards Organization
JVM	Java Virtual Machine
LAN	Local Area Network
LCD	Liquid Cristal Display
LINQ	Language Integrated Query
MAC	Media Access Control
MAC	Medium Access Control
MCU	multipoint control unit
MD5	Message Digest 5
MEGACO	Media Gateway Control Protocol
MF	Network Information Service Infrastructure
MSIL	Microsoft Intermediate Language
MSS	Maximum Segment Size
MTU	Maximum Transfer Unit
NAT	Network address translation
NetBIOS	NETwork Basic Input Output System
NISI	Network Information Service Infrastructure
NLA	Network Location Awareness
OFB	Output Feed Back
OLE	Object Linking and Embedding
OS	Operating System
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PABX	Private Automatic Branch Exchange
PBX	Private Branch Exchange
PC	Personnal Computer
PGP	Pretty Good Privacy
PHP	Pre-HyperText-Processor

PKI	Public-key infrastructure
POP	Post Office Protocol
PPP	Point to Point Protocol
RADIUS	Remote Authentication Dial-In User Service
RAID	Redundancy Array of Inexpensive/Independent Disk
RAM	Random Access Memory
RARP	Reverse Address Resolution Protocol
RBS	Robbed-bit Signaling
RC4	Rivest Cipher 4
RIP	Routing Information Protocol
RNIS	Réseau Numérique à Intégration de Services
ROM	Read Only Memory
RTC	Réseau Téléphonique Commuté
RTCP	Réseau Téléphonique Commuté Public
RTP	Réseau Téléphonique Public
SAP	Service Access Point
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SLIP	Serial Line Interface Protocol
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SP3	Service Pack 3
SPAM	Shoulder of Pork and hAM
SQL	Structured Query Language
SS7	Signaling System 7
TCP	Transmission Control Protocol
TDMoE	TDM over Ethernet
TELNET	TELetypewriter NETwork protocol
TFTP	Trivial FTP
TRIP	Telephony Routing Over IP
TU	transaction user
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UNIX	Uniplexed Information and Computing System
URI	Uniform Resource Identifier
UTF8	Universal Transformation Format 8 bits
VoiP	Voice Over IP
WAN	Wide Area Network
WASO	Windows Advanced System Optimizer
WGA	Windows Geniun Advance

Wi-Fi	Wireless Fidelity
WPF	Windows Presentation Foundation
WTIS	Windows Trust Installer
WTUdate	Windows Trust Update
XML	eXtensible Markup Language
XOR	OU exclusif

REMERCIEMENT

Je remercie le Seigneur Dieu, qui par sa grâce, a permis la réalisation de ce mémoire.
Aussi, je remercie respectueusement :
Monsieur ANDRIANARY Philippe, Professeur, Directeur de l'Ecole Supérieure Polytechnique d'Antananarivo de m'avoir accueilli au sein de l'établissement ;
Monsieur RAZAKARIVONY Jules, Maître de conférences et Chef de Département Télécommunication, pour tous les savoirs qu'il nous a transmis et d'avoir permis l'achèvement de mes études dans les meilleures conditions possibles.
Je tiens à remercier Monsieur RAVONIMANANTSOA Ndaohialy Manda-Vy, qui m'a initié à la cryptologie et m'a fait découvrir le monde passionnant de la recherche et avec qui j'ai eu le plaisir de mener tous mes travaux de mémoire. Sa grande disponibilité et ses merveilleuses explications ont été précieuses.
Je remercie respectueusement :
Monsieur RAKOTOMALALA Mamy Alain, Maître de conférences et Enseignant chercheur au sein du Département Télécommunication, qui nous a fait l'honneur de présider le jury de cette soutenance.
Tous les membres du jury, à savoir :
Monsieur RAKOTONDRAINA Tahina Ezéchiel, Assistant d'Enseignement et de Recherche au sein du Département Télécommunication.
Madame ANDRIANTSILAVO Haja Samiarivonjy, Enseignant chercheur au sein du Département Télécommunication.
Madame RAMAFIARISONA Malalatiana, Enseignant chercheur au sein du Département Télécommunication.
Un grand merci également à l'ensemble du personnel pédagogique, technique et administratif du département principalement mes collègues lors de ma formation.
Je remercie vivement mes parents et mon frère qui ont tant contribué à ma réussite d'aujourd'hui.
Je tiens à exprimer ma profonde gratitude à toutes celles et ceux qui m'ont apporté leur soutien, leur amitié ou leur expérience tout au long de ce travail.

INTRODUCTION GENERALE

Actuellement, la vitesse de traitement de l'information évolue plus rapidement que la quantité d'en stocker et bien plus vite que les canaux de transmission malgré les récentes découvertes, car augmenter la bande passante d'un réseau informatique demande de grands changements d'infrastructure comme les installations téléphoniques. Ainsi on préfère user de la capacité de traitement du processeur plutôt que de devoir augmenter les capacités de stockage et de télécommunication.

Les messages électroniques sont devenus courant. Un simple message texte occupe environ 12Ko en mémoire. Un simple dialogue de quelques minutes est évalué à 1,2Mo. A l'usage professionnel avec des contacts multiples cela peut aller jusqu'à 250Mo ou plus en une journée.

De plus, l'usage de la messagerie via internet passe par divers protocoles tels que SMTP, IMAP, POP, … Le problème se situe dans le fait que ces protocoles ne sont pas toujours compatible (problème d'interopérabilité). Vient aussi le problème de transmission, il y a un certain décalage entre l'émission et la réception des messages. Le décalage peut aller de quelques secondes à plus de 24 heures.

Pour remédier aux problèmes ci-dessus ; les utilisateurs ont du céder certaines fonctionnalités tels que la mis en forme des messages, les pièces jointes. Nous pouvons constater que les e-mails ne peuvent contenir qu'une quantité limité d'espace pour les pièces jointes et la mis en forme des messages est très limité car cela affecte à la fois le stockage ainsi que le temps de transmission. Un message électronique ne peut ainsi donc dépasser 50Mo dans les meilleurs des cas. Il y a aussi la suppression automatique des messages au-delàs d'une période définie pour libérer l'espace de stockage. Une procédure identique se passe en transmission, un message qui ne parvient pas à atteindre son destinataire après un certains nombre d'essai est renvoyé à son expéditeur avec une signalisation d'erreurs.

Pour palier au problème de capacité, l'usager apprend à économiser les espace mémoire en codant les messages (exemple : SMS). Les codages employés sont indépendant des codages de donnée classiques car la reconstitution ne peut se faire que par l'interlocuteur lui-même. Le développement de ces langage a permis d'augmenter considérablement la quantité de donnée réelle transmisse sur une même bande passante.

Les principaux critères d'évaluation de ces méthodes sont :

- La reconstruction,
- La rapidité du codeur décodeur,
- La vitesse de transmission,
- L'espace mémoire requis.

D'autre part, si le but traditionnel de la cryptographie est d'élaborer des méthodes permettant de transmettre des données de manière confidentielle, la cryptographie moderne s'attaque en fait plus généralement aux problèmes de sécurité des communications.

Le but est d'offrir un certain nombre de services de sécurité comme la confidentialité, l'intégrité, l'authentification des données transmises,.... Pour cela, on utilise un certain nombre de mécanismes basés sur des algorithmes cryptographiques. La confidentialité est historiquement le premier problème posé à la cryptographie. Il se résout par la notion de chiffrement. Il existe deux grandes familles d'algorithmes cryptographiques à base de clefs : les algorithmes à clef secrète ou algorithmes symétriques, et les algorithmes à clef publique ou algorithmes asymétriques.

CHAPITRE 1 GENERALITE

1. Présentation du protocole SIP

1.1. Introduction

Session Initiation Protocol (SIP) est un protocole de commande de couche application qui peut établir, modifier et terminer des sessions multimédia (conférences) telles que des communications téléphoniques par l'Internet. SIP peut aussi inviter des participants à des sessions déjà existantes, telles que des conférences en multidiffusion. Des supports peuvent être ajoutés (et retirés) à une session existante. SIP prend en charge de façon transparente la transposition de nom et les services de redirection, ce qui sert de support à la mobilité personnelle [1]. Les utilisateurs peuvent conserver une identification unique vue de l'extérieur, indépendamment de leur localisation dans le réseau.

SIP prend en charge cinq facettes de l'établissement et de la terminaison de communications multimédia :

- Localisation de l'utilisateur : détermination du système terminal à utiliser pour la communication

- Disponibilité de l'utilisateur : détermination de la volonté de l'appelé à s'engager dans une communication ;

- Capacités de l'utilisateur : détermination du support et des paramètres de support à utiliser ;

- Etablissement de session : "sonnerie", établissement des paramètres de session à la fois chez l'appelant et l'appelé ;

- Gestion de session : y compris le transfert et la terminaison des sessions, la modification des paramètres de session, et l'invocation des services.

SIP n'est pas un système de communications intégré verticalement. SIP est plutôt un composant qui peut être utilisé avec d'autres protocoles de l'Internet Engineering Task Force (IETF) pour construire une architecture multimédia complète. Normalement, ces architectures vont inclure des protocoles tels que le protocole de transport en temps réel (RTP) [2] pour le transport en temps réel de données et la fourniture d'informations en retour sur la qualité de service, le protocole à

défilement continu en temps réel (RTSP, Real-Time streaming protocole) [3] pour le contrôle de livraison de supports à défilement continu, le protocole de commande de passerelle de support (MEGACO, Media Gateway Control Protocol) [4] pour le contrôle des passerelles vers le réseau téléphonique public commuté (RTPC), et le protocole de description de session (SDP, Session Description Protocol) [5] pour la description des sessions multimédia. Donc, SIP devrait être utilisé en conjonction avec les autres protocoles afin de fournir des services complets aux utilisateurs. Cependant, la fonction et le fonctionnement de base de SIP ne dépendent d'aucun de ces protocoles.

SIP ne fournit pas de services. Plutôt, SIP fournit des primitives qui peuvent être utilisées pour mettre en œuvre différents services. Par exemple, SIP peut localiser un utilisateur et livrer un objet opaque à l'endroit où il se trouve. Si cette primitive est utilisée pour délivrer une description de session écrite, par exemple, en Session Description Protocol (SDP), les points de terminaison peuvent se mettre d'accord sur les paramètres d'une session. Si la même primitive est utilisée pour livrer une photo de l'appelant aussi bien que la description de session, un service d'"ID d'appelant" peut facilement être mis en œuvre. Comme le montre cet exemple, une seule primitive est normalement utilisée pour fournir plusieurs services différents.

SIP n'offre pas de services de contrôle de conférence du genre de la commande de salle ou des votes et n'a aucune exigence sur la façon dont une conférence doit être gérée. SIP peut être utilisé pour initialiser une session qui utilise un autre protocole de contrôle de conférence. Comme les messages SIP et les sessions qu'ils établissent peuvent passer à travers des réseaux entièrement différents, SIP ne peut pas fournir, et ne fournit pas, de capacités de réservation de ressources de réseau d'aucune sorte.

La nature des services fournis rend la sécurité particulièrement importante. A cette fin, SIP fournit une série de services de sécurité, qui comporte la prévention du déni de service, l'authentification (à la fois d'usager à usager et de mandataire à usager), la protection de l'intégrité, et de services de chiffrement et de confidentialité.

SIP travaille aussi bien avec IPv4 qu'avec IPv6.

1.2. Différences entre SIP et téléphonie traditionnelle

L'étude des aspects sécurité des protocoles de la voix sur IP (Internet Protocol) passe par celle de la téléphonie classique dont voici quelques points de divergence et de parallèle :

- Dans la VoIP, et étant donné que les paquets ne sont pas chiffrés, tout ce qu'un attaquant a besoin est de prendre les paquets appropriés avec un sniffer de paquets. Ce sniffer de paquets peut être un ordinateur attaché, par exemple, au réseau local de l'entreprise. En téléphonie traditionnelle, la téléphonie mobile est exclue, l'attaquant doit avoir un dispositif spécial, qui doit être physiquement relié à un fil, qui est utilisé pendant un appel [6].
- Internet est largement considéré comme peu sûr. Les réseaux avec commutation à circuit ne sont pas entièrement sûrs mais les personnes ne s'inquiètent pas trop à ce sujet [6].

1.3. Définitions des syntaxes

Avant d'étudier de façon plus détaillée le fonctionnement de ce protocole, il est nécessaire d'expliquer la terminologie qui sera utilisée par la suite ainsi que présenter le type de requêtes et leurs significations.

- Session : une session multimédia est constituée d'un ensemble d'émetteurs et de récepteurs multimédia avec des données circulant des émetteurs vers les récepteurs [7].
- User Agent Client (UAC) : entité générant puis envoyant des requêtes. Ce rôle est ponctuel et dure uniquement le temps d'une transaction SIP (c'est à dire : si une application génère une requête SIP, celle-ci va agir comme un UAC durant cette transaction) [7].
- User Agent Server (UAS) : entité générant des réponses aux requêtes SIP. Ces réponses peuvent être une acceptation, un refus ou bien une redirection de la requête reçue. De façon analogue à un UAC, le rôle d'UAS dure uniquement le temps d'une transaction SIP [7].
- User Agent (UA) : entité pouvant agir à la fois comme UAC et UAS, c'est l'application et la transaction SIP qui décident si un UA doit agir comme UAS ou UAC. Un tel UA est constitué d'un module SIP interagissant avec une application. La Figure 1.1 illustre la structure d'un UA. La gestion des interactions entre l'application et le module SIP dépend de l'implémentation de l'UA. Cette spécificité explique la diversité et le nombre d'UA différents existants [7].
- URI (Uniform Resource Identifier) : les URI sont utilisés pour identifier les UA, leur format est similaire à une adresse e-mail (Ex: sip:clarinet@u-strasbg.fr) [7].
- Registrar : un Registrar est un serveur acceptant les requêtes REGISTER et mémorisant les informations reçues pour le domaine qu'il gère. Ce mécanisme permet de gérer la mobilité des UA et de rediriger les requêtes vers la nouvelle localisation. Le Registrar fait correspondre adresses IP et RI pour l'ensemble des URI du domaine sur lequel il a autorité [7].

- Proxy, Proxy Server : un proxy est une entité intermédiaire agissant à la fois comme UAC et UAS afin de faire des requêtes à la place d'autres UAC. Son rôle est de s'assurer qu'une requête est envoyée à une autre entité plus "proche" du destinataire de la requête [7].

Une requête se propage donc de Proxy en Proxy jusqu'à atteindre sa destination. Un proxy interprète et modifie si nécessaire certaines parties spécifiques de la requête avant de la faire suivre.

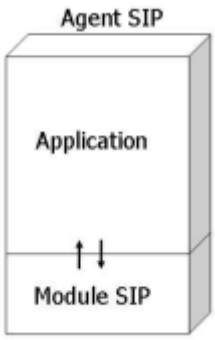


Figure 1.01 *: Structure d'un Agent SIP (UA)*

Les différents types de requêtes SIP [7]

Requête	**Description**
INVITE	invitation de l'appelé dans une session
OPTIONS	requête afin de découvrir les capacités du récepteur (les options qu'il supporte)
BYE	terminaison d'un appel
CANCEL	requête d'abandon d'une requête d'invitation incomplète
ACK	acquittement d'une réponse
REGISTER	enregistre la localisation courante d'un utilisateur

Tableau 1.01 *: Les différents types de requêtes SIP*

En réponse à une requête un UAS renvoie un code d'état afin de signaler de quelle façon la requête a été traitée. Ces codes sont découpés en 6 catégories qui sont décrites dans le Tableau suivant :

Code d'état	Description	Exemple
1xx	information concernant le statut de l'appel	180 RINGING
2xx	réussite	200 OK
3xx	redirection vers un autre serveur	301 MOVED TEMPORARILY
4xx	erreur côté client	401 UNAUTHORISED
5xx	erreur côté serveur	500 INTERNAL SERVER ERROR
6xx	échec global	606 NOT ACCEPTABLE

Tableau 1.02 *: Les classes de réponses SIP*

1.4. Mécanisme

SIP peut fonctionner au dessus de Transmission Control Protocol (TCP) ou d'User Datagram Protocol (UDP). Afin de palier aux pertes éventuelles de messages possibles si UDP est utilisé SIP doit donc avoir ses propres mécanismes de retransmissions. Les requêtes émises par les UAC sont donc retransmises périodiquement jusqu'à réception d'une réponse de l'UAS. Plus spécifiquement pour les requêtes de type INVITE, l'UAC réémet la requête au bout d'une durée initiale de T1 secondes qui double à chaque réémission. L'UAC cesse de réémettre s'il reçoit une réponse de l'UAS ou s'il a réémit le paquet 7 fois. Un mécanisme similaire est utilisé par l'UAS [7].

Pour initier une session SIP, l'UA doit simplement connaître l'adresse URI de l'UA qu'il désire contacter. Un message INVITE est envoyé en direction de cette URI et sera relayé par un ou plusieurs proxies en direction de l'UA correspondant à l'URI du destinataire. La Figure 1.02 illustre l'initialisation et la fermeture d'une session SIP entre deux UA.

Description des différentes étapes représentées dans la Figure 1.02:

1. L'UA identifié par l'URI sip:pascal@u-strasbg.fr souhaite contacter l'UA d'URI sip:mathieu@ustrasbg.fr. Pour cela il envoie une requête de type INVITE à destination de cette URI.

2. Le proxy du domaine u-strasbg.fr fait suivre la requête INVITE et le signale à l'UA de l'appelant en lui retournant le code d'état 100 TRYING.

3. L'UA de l'appelé a reçu le message mais la communication n'est pas encore acceptée (dans le contexte d'une application téléphonique on se retrouve dans la situation où le téléphone sonne et où l'on attend que la personne décroche). L'UA de l'appelé signale cela en renvoyant le code d'état 180 RINGING au proxy qui va l'envoyer à l'UA de l'appelant.

4. L'UA de l'appelé accepte la communication et retourne le code d'état 200 OK au proxy qui le relaye à l'UA de l'appelant.

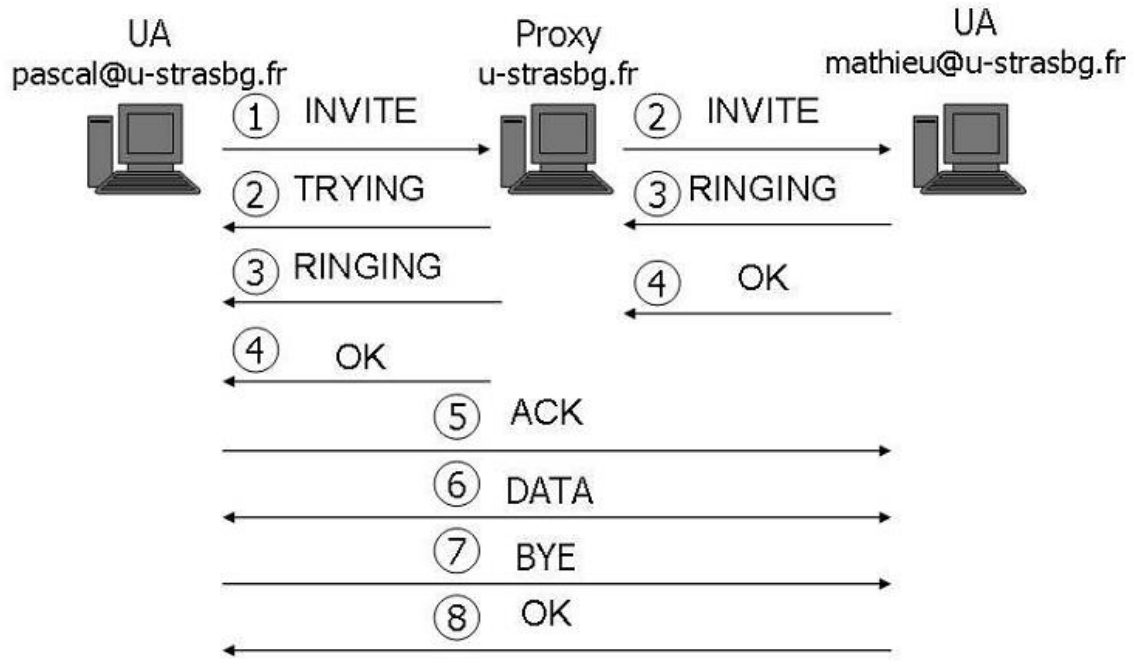


Figure 1.02 *: Illustration d'une session SIP*

5. L'UA de l'appelant envoie un message d'acquittement. A partir de ce moment, la connexion entre les deux UA est initialisée et plus aucun message ne passe par le proxy.

6. Les deux UA peuvent s'envoyer des données via la connexion qui a été décrite dans le message INVITE (UDP ou TCP par exemple). Cette connexion utilisée pour l'échange de données est totalement indépendante de la connexion utilisée pour la signalisation SIP.

7. Ici, un des UA souhaite clore la connexion, il envoie donc un message BYE.

8. Le message BYE est acquitté via un code d'état 200 OK. La connexion entre les deux UA est rompue.

Pour communiquer, les UA s'échangent des messages SIP codés en mode texte (UTF8) ayant une sémantique similaire à celle des messages du protocole HyperText Transfer Protocol (HTTP). L'exemple ci-dessous illustre un entête SIP [7].

```
MESSAGE sip:Alice@alice SIP/2.0
Via: SIP/2.0/UDP 192.168.56.1:1042;branch=z9hG4bK1989472352
Max-Forwards: 70
From: "waren" <waren@waren>;tag=304204663
To: "Alice" <Alice@alice>
Call-ID: 1380323499-1185092406-858768133
CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 16
User-Agent: SIP .NET 1.0
```

Figure 1.03 *: Entête de message SIP*

Descriptif des différents champs de l'entête SIP :

– La première ligne contient le type de message (ici il s'agit d'un message INVITE)

– **Via :** ce champ représente le chemin parcouru par la requête, lors de l'envoie de la requête il contient initialement l'URI de l'émetteur de cette requête. Ensuite, à chaque fois qu'un UA fait suivre la requête celui-ci rajoute sa propre URI dans ce champ.

– **Max-Forwards** : nombre maximum de sauts

– **To** : URI du destinataire du message

– **From** : URI de l'expéditeur du message

– **Call-ID** : identifiant unique représentant la session SIP

– **Cseq** : contient un entier et un nom de méthode, l'entier est généré aléatoirement une première fois puis est ensuite incrémenté à chaque nouveau message.

– **Contact** : contient une ou plusieurs URI représentant une route directe pour contacter l'UA de l'expéditeur (ici l'UA de Pascal)

– **Content-Type** : description du corps du message

– **Content-Length** : taille en octet du corps du message

– **branch, tag** : utilisés pour des raisons d'identification

SIP permet donc aisément d'initier une communication entre deux UA. Pour cela, il suffit qu'un UA émette un message SIP INVITE qui sera relayé avant d'atteindre l'UA destinataire qui pourra accepter ou non cette communication [7].

La seule information nécessaire afin de pouvoir contacter un UA est de connaître son URI.

Des serveurs appelés REGISTRAR sont utilisés pour maintenir la cohérence entre URI et localisation dans le réseau. Ce mécanisme permet également de gérer la mobilité des UA et de garantir une continuité de service lorsque l'UA se déplace. Un UA ayant changé de localisation dans le réseau (nouvelle adresse IP, nouveau domaine) peut signaler à son REGISTRAR sa nouvelle position via un message REGISTER [7]. La Figure 1.04 illustre le fonctionnement d'un REGISTRAR et montre comment une requête SIP se propage jusqu'à l'UA souhaité.

Description des mécanismes représentés dans la Figure 1.04 :

1. L'UA ayant pour URI sip:mathieu@u-strasbg.fr contacte le REGISTRAR autoritaire sur son domaine (u-strasbg.fr) pour lui communiquer sa nouvelle localisation via un message REGISTER. Dans cet exemple, l'UA en question se voit attribuer l'adresse dhcp-pc34.ustrasbg.fr.

2. Le REGISTRAR enregistre la nouvelle adresse correspondant à l'URI sip:mathieu@ustrasbg.fr dans son Location Service (le Location Service est un système d'annuaire associant une adresse à une URI).

3. Ici, l'UA cherche à contacter l'URI sip:mathieu@u-strasbg.fr, cet UA envoie donc une requête SIP qui sera reçue par un des proxies SIP autoritaires sur le domaine.

4. Le proxy interroge le Location Service afin d'obtenir l'adresse associée à l'URI.

5. Le proxy reçoit l'adresse associée à l'URI.

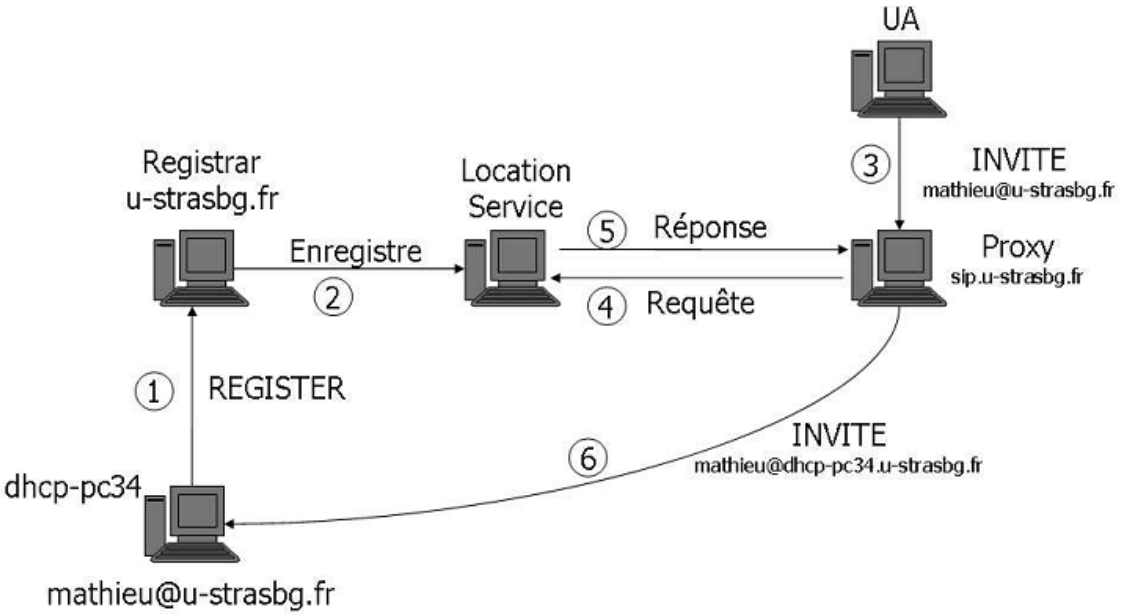


Figure 1.04 : *Fonctionnement d'un REGISTRAR SIP*

6. Le proxy redirige la requête initiale vers l'UA destination et ce de façon totalement transparente pour l'UA source.

L'exemple de la Figure 1.04 se place dans le contexte d'une requête SIP intra-domaine mais le mécanisme illustré fonctionne également en inter-domaine, la requête se propage de proxy en proxy jusqu'à atteindre le domaine destination. Les REGISTRAR rendent donc l'établissement d'une connexion entre deux UA possible et ce sans aucune contrainte de localisation.

1.5. Structure

SIP est structuré comme un protocole en couches, ce qui signifie que son comportement est décrit en termes d'ensembles de stades de traitement très indépendants avec seulement un couplage lâche entre chaque stade. Le comportement du protocole est décrit en couches pour les besoins de la présentation, permettant la description de fonctions communes à travers des éléments dans une seule section. Cela n'impose en aucune façon une quelconque mise en œuvre. Lorsque nous disons qu'un élément "contient" une couche, nous voulons dire qu'il est conforme à l'ensemble des règles définies par cette couche [8].

Tous les éléments spécifiés par le protocole ne contiennent pas toutes les couches. De plus, les éléments spécifiés par SIP sont des éléments logiques, non pas physiques. Une réalisation physique peut choisir d'agir comme différents éléments logiques, peut-être même transaction par transaction.

La plus basse couche de SIP est sa syntaxe et son codage. Son codage est spécifié en utilisant une grammaire Backus-Naur Form (BNF) augmentée. Le BNF complet est spécifié à la Section 25 ; on trouvera une vue générale de la structure du message SIP à la Section 7 [8].

La seconde couche est la couche de transport. Elle définit comment un client envoie des demandes et reçoit les réponses et comment un serveur reçoit les demandes et envoie les réponses sur le réseau. Tous les éléments SIP contiennent une couche transport. La couche transport est décrite à la Section 18 [8].

La troisième couche est la couche transaction. Les transactions sont un composant fondamental de SIP. Une transaction est une demande envoyée par un client de transaction (utilisant la couche transport) à un serveur de transaction, avec toutes les réponses à cette demande renvoyées depuis le serveur de transaction au client. La couche de transaction traite les retransmissions de couche application, appariant les réponses aux demandes, et les temporisations de couche application. Toutes les tâches qu'accomplit un client d'agent d'utilisateur (UAC, user agent client) ont lieu en utilisant une série de transactions. L'exposé sur les transactions figure à la Section 17 [8]. Les agents d'utilisateur contiennent une couche transaction, tout comme les mandataires à états pleins (stateful). Les mandataires sans état ne contiennent pas de couche de transaction. La couche de transaction a un composant client (désigné comme transaction de client) et un composant serveur (désigné comme transaction de serveur), chacun d'eux étant représenté par une machine à états finis qui est constituée pour traiter une demande particulière.

La couche au-dessus de la couche de transaction est appelée utilisateur de transaction (TU, transaction user). Chacune des entités de SIP, excepté les mandataires sans état, est un utilisateur de transaction. Lorsqu'un TU souhaite envoyer une demande, il crée une instance de transaction de client et lui passe la demande avec l'adresse IP de destination, le port, et le transport, auxquels envoyer la demande. Un TU qui créée une transaction de client peut aussi l'annuler.

Lorsqu'un client annule une transaction, il demande que le serveur arrête le traitement, revienne à l'état qui existait avant l'initialisation de la transaction, et génère une réponse d'erreur spécifique de cette transaction. Ceci est fait avec une demande CANCEL, ce qui constitue une transaction en soi, mais se réfère à la transaction à annuler (Section 9) [8].

Les éléments SIP, qui sont les clients et serveurs d'agent d'utilisateur, les mandataires sans état et à états pleins et les registres, contiennent un noyau qui les distingue les uns des autres. Les noyaux (Cores), sauf pour le mandataire sans état, sont des utilisateurs de transaction. Alors que le

comportement des noyaux des UAC et UAS dépend de la méthode, il y a certaines règles communes pour toutes les méthodes (Section 8) [8]. Pour un UAC, ces règles gouvernent la construction d'une demande; pour un UAS, elles gouvernent le traitement d'une demande et la génération d'une réponse. Comme les enregistrements jouent un rôle important dans SIP, un UAS qui tient un REGISTER reçoit le nom spécial de registre. La Section 10 décrit le comportement des noyaux d'UAC et d'UAS pour la méthode REGISTER [8]. La Section 11 décrit le comportement des noyaux d'UAC et d'UAS pour la méthode OPTIONS, utilisée pour déterminer les capacités d'un UA [8].

Certaines autres demandes sont envoyées au sein d'un dialogue. Un dialogue est une relation SIP d'homologue à homologue entre deux agents d'utilisateur qui persiste pendant un certain temps. Le dialogue facilite le séquençage des messages et l'acheminement appropriés des demandes entre les agents d'utilisateur. La méthode INVITE est la seule façon définie dans la présente spécification pour établir un dialogue. Lorsqu'un UAC envoie une demande qui est dans le contexte d'un dialogue, il suit les règles communes d'UAC telles qu'exposées à la Section 8 mais aussi les règles pour les demandes de mi- dialogue. La Section 12 expose les dialogues et présente les procédures pour leur construction et leur maintenance, en plus de la construction de demandes au sein d'un dialogue [8].

La plus importante méthode dans SIP est la méthode INVITE, qui est utilisée pour établir une session entre des participants. Une session est une collection de participants, et des flux de support entre eux, pour les besoins d'une communication. La Section 13 expose comment les sessions sont initialisées, résultant en un ou plusieurs dialogues SIP [8].

La Section 14 expose comment modifier les caractéristiques de cette session par l'utilisation d'une demande INVITE au sein d'un dialogue. Finalement, la Section 15 expose comment terminer une session [8].

Les procédures des Sections 8, 10, 11, 12, 13, 14, et 15 traitent entièrement du noyau d'UA (la Section 9 décrit l'annulation, qui s'applique au noyau d'UA et au noyau de mandataire La Section 16 discute de l'élément mandataire, qui facilite l'acheminement de messages entre les agents d'utilisateur [8].

1.6. La sécurité interne du protocole

SIP utilise une multitude de méthodes de sécurité telles que l'authentification et le chiffrement des données. SIP offre aussi la possibilité de cacher le chemin utilisé par certains messages puisque l'adresse est parfois ajoutée dans les champs VIA et les proxies intermédiaires .

1.6.1. Le chiffrement

Le chiffrement des données peut être utilisé de bout en bout ou de nœud à nœud entre les entités SIP. Le chiffrement nœud à nœud chiffre tout le message et il est proposé que ce mode fonctionne au niveau transport [6]. Le protocole IPsec est un candidat pour remplir ce rôle. Alors que le chiffrement bout en bout est utilisé entre les deux applications clientes qui doivent respecter les conditions suivantes :

a. Tous les champs doivent être chiffrés de telle sorte que les entités intermédiaires ne les « comprennent » pas.

b. Tous les champs non chiffrés doivent précéder les champs chiffrés

c. Il n'est pas nécessaire de ne chiffrer aucun en-tête SIP

d. La réponse à un message chiffré doit être chiffrée par une clef donnée dans les champs Response-Key-Header, si aucune clef est spécifiée, le message est non chiffré

e. Les en-têtes qui sont chiffrées dans le message demande doit l'être dans le message réponse

1.6.2. Authentification

Si l'utilisateur a opté pour authentifier ses messages, alors il doit signer les messages qu'il envoie. La signature englobe tout le message mais non pas toutes les en-têtes SIP. Ces en-têtes qui ne sont pas incluses sont celles qui changent entre les entités intermédiaires comme le champ VIA. Si un calcul d'intégrité est choisi, un message d'erreur est produit pour avertir le récepteur de ce changement d'information [6].

1.6.3. Cacher le chemin utilisé (Hide-Route)

Hide-Route est un mécanisme employé par SIP pour ne pas avancer quant aux chemins utilisés par les paquets SIP. Si ce mécanisme est activé, les proxies intermédiaires chiffrent le champ VIA [6].

1.7. Les risques qui restent à envisager

1.7.1. Exemples de scénario

Dans cette partie, on recensera quelques scénario de fraude ou d'utilisation malpropre des possibilités offertes par le protocole SIP. Dans la suite, on supposera que A et B sont deux personnes qui veulent communiquer via le protocole SIP et que C est une autre personne qui perturbe la communication.

C'est ainsi qu'on peut avoir des cas de figure non prévus par la norme :

- C appelle B prétendant être A.
- A appelle B, et dés que l'appel est établi C envoie un message BYE à A ou à B.
- A appelle B; au moment d'attente de la sonnerie, C envoie un message CANCEL à B;
- A appelle B; B envoie alors un message ACK; C envoie aussi un faux ACK avec son propre adresse IP/numéro de port; quand A envoie à son tours ACK, il est ignoré par B et C fait partie de la conversation
- C, envoie des faux INVITE causant un déni de service, le téléphone sonne toujours
- Un faux proxy, envoie un OK mais efface l'enregistrement lors d'un enregistrement multicast, et alors l utilisateur ne reçoit plus d'appels.

1.7.2. *E*xaminons maintenant quelques cas d'attaques :

1.7.2.1. Exemples d'attaques sur le téléphone SIP

- Condition préliminaire: accéder au LAN (Local Area Network)
- Déni de service :

  i. Dépassement de la pile par l'envoi de longues demandes « GET » 6 GRES, Décembre 2001, Marrakech.

  ii. Utilisation des logiciels de contrôle à distance

- Attaque :

  i. Hijacking d'une session TCP vers let téléphone IP ceci étant possible si une des conditions suivantes est vérifiée :

  1. L'algorithme de génération de CALL-ID est crypto graphiquement faible.

2. Si la session de gestion n'est pas suffisamment protégée

ii. La brute force pour accéder au mot de passe de l'administration

- Conclusion : le téléphone IP doit être protégé contre ce genre d'attaques.

1.7.2.2. Exemples d'attaques contre le proxy

- Condition préliminaire : accéder au LAN
- Déni de service :

i. SYN-flooding

ii. Enregistrement périodique des utilisateurs

- Attaque :

i. Modification des données d'enregistrement

1. Effacer des enregistrements

2. Enregistrer un utilisateur avec une nouvelle adresse IP eu une nouvelles URL

3. Redirection d'un utilisateur vers une autre adresse

4. Enregistrement illégal

i. Enregistre un utilisateur avec une fausse adresse

ii. Rediriger tous les appels dont le destinataire est choisi vers une autre adresse

- Conclusions

i. Les proxies ne testent pas suffisamment

ii. Nécessité de définition des polices de sécurité bien poussée

iii. Les proxies doivent réagir comme une Demilitarized Zone (DMZ)

1.7.2.3. Exemples d'attaques pour accéder au réseau SIP et faire partie de la communication

- Condition préliminaire : accéder au LAN

- Déni de service : SYN-floding
- Attaque :

    i. Ecoute des paquets RTP (Réseau Téléphonique Public)

        1. Ecouter les paquets UDP, identifier les paquets RTP
        2. Analyser ces paquets et émettre vers le destinataire
        3. Utilisation de logiciels de type « sniffer »

    ii. Modification des données de signalisation

        1. Redirection des paquets
        2. Modification des données audio
        3. Détecter les ports RTP et envoyer des vidéos profanes

Réaliser un scénario de fraude passe par différentes modalités qu'on résume dans la liste suivante :

- Les messages interceptés :
    - Récupérer le CALL-ID : qu'on peut insérer dans un faux paquet.
    - Récupérer le CALL-LEG (TO-FROM-SUBJECT)
- Les faux messages :
    - Le faux BYE, ayant pour résultat l'arrêt de l'appel.
    - Le faux CANCEL (spoofing) ayant pour résultat l'arrêt de INVITE
    - Le faux ACK (Spoofing) permettant le détournement de l'appel (hijacking)
- Des messages incorrects répétés indéfiniment :
    - Déni de service et le flooding
    - Attaques de la mémoire tampon (overrun/stack)
- Média intercepté
    - Récupérer les informations sur le port RTP (Real Time Protocol) utilisé.
    - Récupérer l'adresse IP, numéro du port.
    - Récupérer le contenu de la charge utile.

- Médias non-désirés ou faux

Il y a diverses causes de fraude ou d'utilisation mal- seine du protocole SIP.

- Identités non authentifiées dans les champs " FROM "
- Non-protection des messages de signalisation (texte en clair)
- Messages non authentifiés (BYE, CANCEL, ACK)
- La signalisation n'indique pas l'émetteur des médias

De plus une des attaques les plus difficiles à prévenir est celle par DOS (Deny Of Service) dont les causes peuvent être résumées en :

1. INVITE : a pour effet le non-établissement de l'appel ou le changement de la session
2. INVITE avec une temporisation : qui a pour effet le time-out de la session
3. BYE : rend la session interminable
4. CANCEL : empêche la sonnerie du téléphone
5. OPTIONS : n'affecte pas l'appel
6. REGSTER : induit la non réception des appels

1.7.3. Résumé

Dans ce qui suit, on résumera quelques problèmes de sécurité rencontrés en utilisant SIP :

1. Le forking : le forking est une situation où A appelle B et la demande d'invitation d'appel est envoyée à B1 et à B2, qui sont de différents terminaux que l'utilisateur utilise. Le résultat devrait être que les terminaux B1 et B2 doivent sonner. Le mécanisme de Handshacke utilisé dans le SIP ne fonctionne pas avec le forking, seulement B1 sonne dans les cas où le Handshacke est utilisé. L'utilisation des demandes signées sans Handshacke peut résoudre le problème mais ceci exigerait l'utilisation de PKI (Public-key infrastructure). Les attaques par rejeu peuvent être réalisées en mémorisant les Call-IDs.
2. Attaque par réflexion peut se produire en utilisant l'authentification pour la demande et la réponse. Si le même secret partagé est utilisé dans les deux directions, un attaquant peut obtenir des qualifications en reflétant un défi dans une réponse répondant ainsi à une demande.

L'utilisation de différents secrets dans chaque direction élimine l'attaque. Ce genre d'attaque n'est pas un problème quand le PGP (Pretty Good Privacy) est utilisé pour l'authentification.

3. L'authentification par multiple proxies : il faut que chaque proxy garde les traces des premiers secrets dits créances; le mécanisme est le suivant :

4. Problèmes de REGISTER le seul message qui donne la possibilité d'écrire, on peut alors remplir la base de données par des CGI (Common Gateway Interface),…

5. Problème de CANCEL : les proxies peuvent générer des messages CANCEL, et donc peuvent mettre fin à une communication

6. Des proxies non authentifiés dans un réseau SIP 8 GRES, Décembre 2001, Marrakech.

7. Les failles du protocole de routage inter domaine : TRIP

8. La norme SIP prévoit l'utilisation des modes d'authentification dont le mode HTTP-Digest, mais ce mode n'est pas compatible avec le mode d'authentification des serveurs RADIUS

9. L'authentification prévue par la RFC2617 qui traite de SIP et celle de RADIUS

(RFC 2138) utilisent MD5 avec des formats de messages différents.

10. Compte tenu du fait que le protocole SIP fonctionne sous RTP, une des menaces est le SPAM RTP.

11. La traversée des firewalls : qui pose des problèmes d'écriture des règles de sécurité.

12. La traversée des NAT (Network address translation).

2. Le chiffrement

La cryptologie est née avec l'apparition de l'écriture et fut justifiée par le besoin de protéger tout message écrit afin d'éviter que l'ennemi ne puisse, en se l'appropriant, exploiter les renseignements qu'il contenait. Littéralement « science du secret », elle a longtemps été associée à de mystérieux enjeux d'espionnage militaire et diplomatique bien éloignés des préoccupations scientifiques. La cryptographie est quant à elle, l'ensemble des techniques constituant la cryptologie.

Ses premières formes furent le plus souvent basiques, mais avec l'évolution des peuples des technologies et des « casseurs de codes », son développement est en constante recherche de progression. Comment la cryptologie émergea ? Où en est-elle de nos jours ? Quel avenir a-t-elle ?

A l'époque de Jules César, des méthodes rudimentaires permettaient de rendre ses ordres « incompréhensibles » à ses adversaires. Néanmoins, l'absence de rigueur des concepteurs de ces systèmes mena à des failles qui permettaient à leurs adversaires de comprendre les messages malgré tout.

De nos jours l'information sous toutes ses formes : voix, images, œuvres musicales, textes et autres, circule au format numérique à travers le monde en une fraction de seconde. Que ce soit par le téléphone, le câble, les fibres optiques ou par satellite, cette information est chaque jour échangée d'un point à un autre et se trouve susceptible d'être lue, copiée, supprimée, altérée ou falsifiée. La cryptologie répond aujourd'hui aux besoins du marché et constitue un domaine scientifique en pleine activité. Elle intervient dans de multiples applications et représente l'élément essentiel de la sécurisation du commerce électronique et du réseau Internet.

La transmission d'un message codé qui se veut sûr doit satisfaire les trois conditions suivantes :

« Confidentialité » : la technique de cryptage se doit de garantir le secret de l'information, aucun tiers ne doit pouvoir lire le message.

« Intégrité » : le crypto système ne doit engendrer aucune absence de modification de l'information, et le message ne doit pas pouvoir être modifié durant son « transport ».

« Authenticité »: le codage du message doit garantir l'origine de l'information, le message doit annoncer son émetteur sans le trahir.

2.1. Les algorithmes de chiffrements symétriques

La cryptographie à algorithmes symétriques fonctionne habituellement suivant deux procédés différents, le chiffrement par blocs et le chiffrement par flot (en continu).

2.1.1.1. Le chiffrement par flot

Pour comprendre le chiffrement en continu, il suffit de connaître par exemple les vidéos au format RealVideo très répandues sur internet : on visualise l'image au fur et à mesure que les données sont

reçues. Le principe est le même dans le cas de nos "stream-ciphers" : le chiffrement est effectué bit-à-bit sans attendre la réception complète des données à chiffrer.

Une technique de chiffrement, du nom de "One-Time Pad" est utilisée pour chiffrer les flux. C'est le chiffrement inconditionnel le plus sûr. Pour cela, on a besoin d'une chaîne aléatoire de la même longueur que le message d'origine, ce qui n'est pas pratique. Le but d'un streamcipher est de générer une chaîne aléatoire à partir d'une clé de longueur courte.

Une autre technique consiste à "xorer", c'est-à-dire à appliquer un OU exclusif (XOR) au message avec un autre message prédéfini. Bien entendu, cela nécessite que le destinataire (la personne qui déchiffre) connaisse le message prédéfini et donc cela rajoute de la complexité au schéma général. Les stream-ciphers sont utilisés aujourd'hui par différentes applications. Pour chiffrer les flux, l'algorithme RC4 est très utilisé [11].

2.1.1.2. Le chiffrement par bloc

Quatre modes de chiffrement par bloc sont utilisés : Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feed Back (CFB) ou Output Feed Back (OFB). Le chiffrement en blocs (block-cipher) est au contraire beaucoup plus utilisé et permet une meilleure sécurité. Les algorithmes concernés sont également plus connus (DES, AES, Skipjack...); leur nom leur vient du fait qu'ils s'appliquent à des blocs de données et non à des flux de bits (cf. stream-ciphers). Ces blocs sont habituellement de 64 bits mais cela dépend entièrement de l'algorithme utilisé et de son implémentation. De même, la taille de la clé varie suivant l'algorithme et suivant le niveau de sécurité requis; ainsi, un chiffrement de 40 bits (c'est-à-dire utilisant une clé longue de 40 bits) pourra être déclaré faible puisque aisément cassable. Un chiffrement de 56 bits (qui est le standard dans le cas du DES) sera qualifié de moyen puisque cassable mais nécessitant pas mal de moyens pour être exploitable (vis-à-vis du temps requis et de la valeur des données). Enfin, un chiffrement de 128 bits (valeur standard utilisée par Rijndael alias AES) est plutôt fort à l'heure actuelle [11].

Rappelons à cette occasion que la Loi de Moore prévoit le doublement de la puissance de calcul des processeurs tous les 18 mois (Loi toujours vérifiée de la fin des années 70 à nos jours). Sans entrer dans les détails, il faut savoir que le cassage de cryptés nécessite essentiellement des ressources processeur, RAM et éventuellement ROM ou disque dur si le cassage se fait par pré calcul. L'évolution générale est donc extrêmement rapide, sans parler des ordinateurs plus perfectionnés

(scientifiques ou autres), à architectures parallèles, ou distribuées... Il reste donc relatif de parler de sécurité absolue, en tout cas en ce qui concerne la cryptographie symétrique.

Les quatre modes cités précédemment sont plus ou moins indépendantes de l'algorithme choisi. Toutefois, tous les algorithmes ne permettent pas d'utiliser tout les modes possibles.

Pour mieux comprendre, voyons ces modes plus en détails. Pour désigner le processus de chiffrement simple (tel que décrit précédemment), on utilisera la notation suivante :

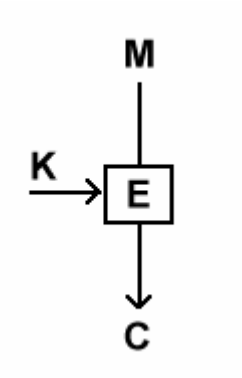


Figure 1.05 *: Mode dictionnaire*

où K désigne la clé utilisée par l'algorithme, E désigne le chiffrement en lui-même, M (ou m, mi) désigne le message en clair (c'est-à-dire un bloc) et C (ou c, ci) le chiffré résultant.

- Le mode Electronic Code Book (ECB) est le plus simple des modes et s'applique aux blocks ciphers. Il revient à chiffrer un bloc indépendamment des autres; cela permet entre autre de chiffrer suivant un ordre aléatoire (bases de données, etc...) mais en contrepartie, ce mode est très vulnérable aux attaques. Il est par exemple possible de recenser tous les cryptés possibles (code books) puis par recoupements et analyses statistiques recomposer une partie du message original sans avoir tenté de casser la clé de chiffrement. Il demeure que si la clé fait 128 bits ou plus, cette attaque n'est pas exploitable en pratique de nos jours [11].

Cette technique est sensible à l'inversion ou la duplication de blocs sans que le destinataire s'en aperçoive.

- Le mode Cipher Block Chaining (CBC) peut-être utilisé par les algorithmes en bloc [11].

C'est d'ailleurs le mode le plus courant. Il permet d'introduire une complexité supplémentaire dans le processus de chiffrement en créant une dépendance entre les blocs successifs; autrement dit, le chiffrement d'un bloc va être -d'une manière ou d'une autre- lié à ou aux blocs/chiffrés précédents.

Quant au cas où la longueur est celle de l'algorithme (à savoir n), le schéma de CFB se simplifie et ressemble quelque peu à celui de CBC (à quelques nuances près) :

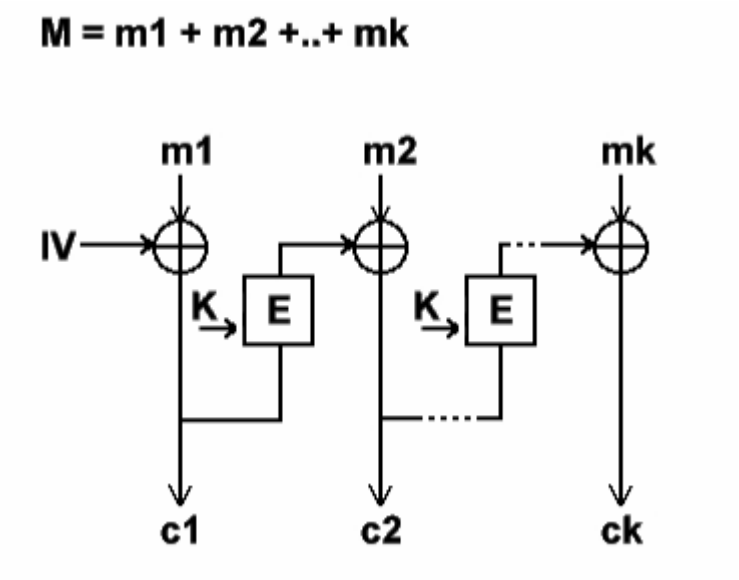


Figure 1.06 ***:*** *Cipher Block Chainning*

- Le mode Output Feed Back (OFB) est une variante de mode CFB précédemment abordé. Il est d'ailleurs parfois appelé internal feedback. Il présente beaucoup de problèmes de sécurité et il est peu conseillé sauf dans le cas où sa longueur est égale à celle de l'algorithme utilisé [11].

2.2. Le chiffrement AES Rijndael

Rijndael a été conçu par Joan Daemen et Vincent Rijmen, deux chercheurs de la Belgique, dans le but de devenir un candidat à l'Advanced Encryption Standard (AES) du NIST. Après avoir réussi à se classer dans les six premiers, Rijndael a été choisi le standard en 2000, prenant la place du premier véritable standard de la cryptographie : le DES [9].

Le chiffrement a une longueur de bloc variable, une longueur de clé variable et un nombre de rounds variables. Par contre, Rijndael version "AES" est restreint à des longueurs de clé de 128, 192 et 256 bits avec une longueur de bloc fixée à 128 bits(en fait, Rijndael supporte également des tailles de blocs variables, mais cela n'est pas retenu dans le standard).

Trois critères principaux ont été respectés dans sa conception :

– Résistance face à toutes les attaques connues.

– Rapidité du code sur la plus grande variété de plates-formes possible.

– Simplicité dans la conception.

Rijndael (1998) a été fortement influencé par son prédécesseur, l'algorithme Square (1997). Les algorithmes Crypton et Twofish utilisent aussi des opérations de Square. Rijndael se prononce "Raindal"

2.3. Fonctionnement du chiffrement

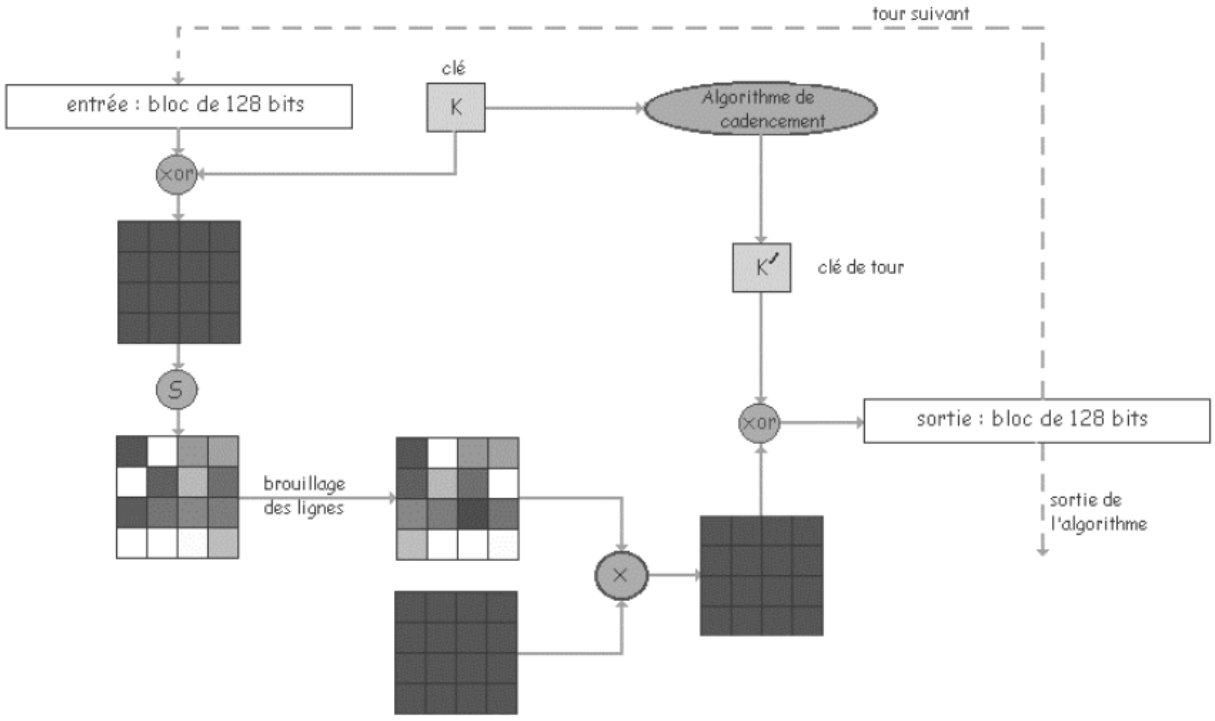


Figure 1.07 *: Fonctionnement d'un chiffrement AES bloc de 128 bits*

Pour AES les blocs de données en entrée et en sortie sont des blocs de 128 bits, c'est à dire de 16 octets.

Les clés secrètes ont au choix suivant la version du système : 128 bits (16 octets), 192 bits (24 octets) ou 256 bits (32 octets).

On découpe les données et les clés en octets et **on les place dans des tableaux [10]**. Les données comportent t_d = 16 octets $p_0,p_1, \cdots ,p_{15}$ qui sont classés dans un tableau ayant 4 lignes et 4 colonnes. Le tableau est rempli colonnes par colonnes.

De même la clé est découpée en octets (t_k = 16, t_k = 24 ou t_k = 32 octets) $k_0,k_1, \cdots ,k_{t-1}$. Ces octets sont aussi classés dans un tableau k de 4 lignes et N_k colonnes (N_k = 4, N_k = 6 ou N_k = 8).

p_0	p_4	p_8	p_{12}
p_1	p_5	p_9	p_{13}
p_2	p_6	p_{10}	p_{14}
p_3	p_7	p_{11}	p_{15}

k_0	k_4	k_8	k_{12}	k_{16}	k_{20}	k_{24}	k_{28}
k_1	k_5	k_9	k_{13}	k_{17}	k_{21}	k_{25}	k_{29}
k_2	k_6	k_{10}	k_{14}	k_{18}	k_{22}	k_{26}	k_{30}
k_3	k_7	k_{11}	k_{15}	k_{19}	k_{23}	k_{27}	k_{31}

Le système AES effectue **plusieurs tours** d'une même composition de transformations.

2.3.1. Le nombre de tours

Suivant la version (la taille de la clé), ce nombre de tours noté nr est différent. Le nombre nr est donné dans le tableau suivant.

K_k	4	6	8
n_r	10	12	14

2.3.2. La clé de tour

À partir de la clé initiale K, le système crée $n_r + 1$ clés de tour ayant chacune 16 octets. Ces clés seront stockées dans un tableau unidimensionnel TK et seront notées

TK [0], TK [1], ⋯ TK [n_r].

Nous verrons ultérieurement comment sont calculées ces clés en fonction de la clé K du système

2.3.3. Vue globale du fonctionnement

La procédure suivante décrit le fonctionnement global du système AES. Elle prend en entrée un tableau de données St (texte clair) qui est modifié par la procédure et renvoyé en sortie (texte chiffré).

Entrée : le tableau *St* et la clé *K*

Sortie : le tableau *St* modifié

AES (St, K)

Début

KeyExpansion (K, TK);

AddRoundKey (St, TK [0]);

Pour *(i=1; i<n_r; i++) Round (St, TK[i]);*

FinalRound (St, TK [n_r]);

fin

Les procédures appelées Round et FinalRound sont elles-mêmes composées

Entrée : le tableau d'état *St* et une clé de tour *T*

Sortie : le tableau *St* modifié

Round (St, T)

Début

SubBytes(St) ;

ShiftRows(St) ;

MixColumns(St);

AddRoundKey(St, T);

Fin

Entrée: le tableau d'état *St* et une clé de tour *T*

Sortie : le tableau *St* modifié

FinalRound(St, T)

Début

SubBytes(St, T) ;

ShiftRows(St) ;

AddRoundKey(St, T);

fin

2.3.4. Les détails

2.3.4.1. la procédure SubBytes

Cette procédure est la seule transformation qui ne soit pas linéaire. C'est donc grâce à celle-ci que le système est résistant. Elle utilise une opération sur le corps ni à 256 éléments.

2.3.4.2. le corps ni à 256 éléments

Considérons le polynôme

$$P(X) = X^8 + X^4 + X^3 + X + 1 \quad (1.01)$$

Ce polynôme à coefficients dans le corps à 2 éléments $\mathbf{F_2 = \{0, 1\}}$ est irréductible sur ce corps.

Les éléments du corps à 256 éléments seront les octets

$$b_7 b_6 b_5\, b_4 b_3 b_2 b_1 b_0$$

Considérés comme des polynômes

$$b(x) = b_7 X^7 + b_6 X^6 + b_5 X^5 + b_4 X^4 + b_3 X^3 + b_2 X^2 + b_1 X + b_0 \quad (1.02)$$

Ce qui nous permet de définir les deux opérations suivantes :

Addition

$$a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 + b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 = c_7 c_6 c_5 c_4 c_3 c_2 c_1 c_0, \quad (1.03)$$

avec

$$c(x) = a(x) + b(x) \quad (1.04)$$

Ce qui donne aussi

$$c_i = a_i + b_i \quad (1.05)$$

Multiplication

$$a_7a_6a_5a_4a_3a_2a_1a_0 \times b_7b_6b_5b_4b_3b_2b_1b_0 = c_7c_6c_5c_4c_3c_2c_1c_0\,, \qquad \textbf{(1.06)}$$

avec

$$c(X) = a(X) \times b(X) \mod P(X). \qquad (1.07)$$

On a ainsi une structure de corps et donc tout élément non nul n'est inversible. Nous noterons g l'application de F_{256} dans F_{256} dénie par

$$g(x) = \begin{cases} 0 \ si \ x = 0 \\ x^{-1} \ sinon \end{cases} \qquad (1.08)$$

L'inverse d'un élément **b(X)** se trouve par l'algorithme d'Euclide étendu.

2.3.4.3. La fonction affine f

Définissons **b = f(a)** grâce à une matrice

$$\begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

La matrice carrée qui intervient est une matrice circulante, elle correspond donc à une multiplication de polynômes modulo $\mathbf{X^8 - 1}$.

$$\mathbf{b(x) = ((X^4 + X^3 + X^2 + X + 1) \times a(X) \mod (X^8 + 1)) + (X^6 + X^5 + X + 1)}. \qquad (1.09)$$

On remarque que $g^{-1} = g$ et que f^{-1} est dénie par

$$\begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

2.3.4.4. La procédure SubByte

On définit alors

s(a) = f (g(a)) (1.10)

On a donc aussi

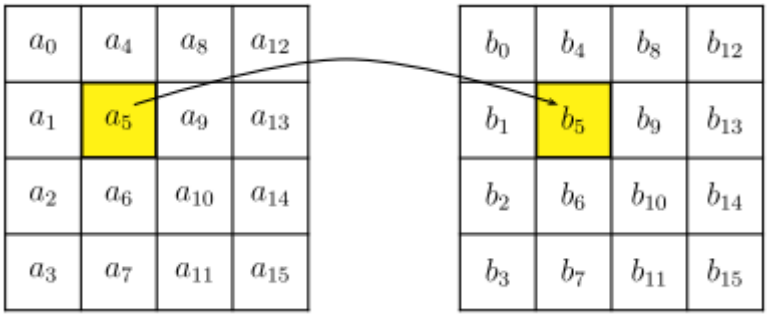
s^{-1}(b) = g f^{-1}(b). (1.11)

La procédure SubByte applique s à chaque octet de l'entrée St.

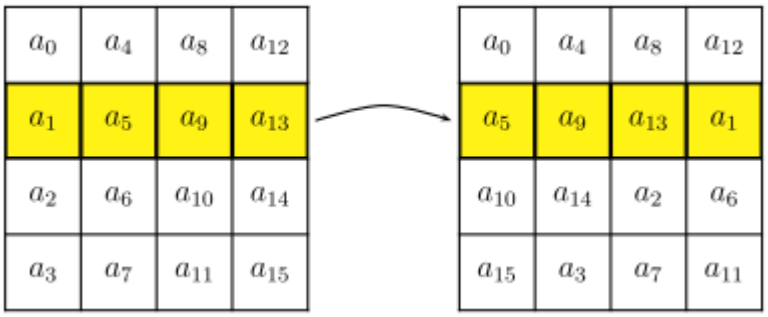
2.3.4.5. La procédure ShiftRows

La procédure consiste à opérer une rotation à gauche sur chaque ligne du tableau d'entrée. Le nombre de cases dont on décale la ligne i ($0 \leq i \leq 3$) est de i.

La transformation inverse est immédiate à calculer.



Transformation SubBytes



Transformation ShiftRows

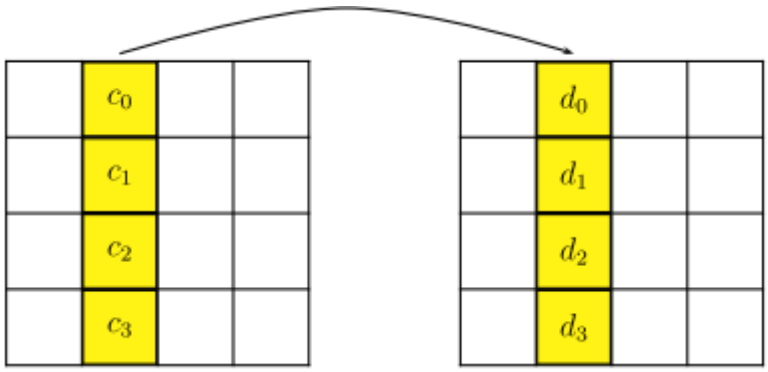
2.3.4.6. La procédure MixColumns

La transformation MixColums consiste à appliquer à chaque colonne du tableau des données une même transformation que nous allons décrire.

Considérons une colonne

$$c = (c_1, c_2, c_3, c_4)^t \tag{1.12}$$

Les éléments c i sont des éléments de F_{265}. Chaque colonne c est transformée en une colonne d grâce à la transformation linéaire suivante donnée par sa matrice dont les coefficients sont dans F_{256} et que nous écrivons comme des octets en hexadécimal :



Transformation MixColums

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \times \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

Là encore la matrice utilisée est circulante. La transformation correspond en fait à une multiplication par un polynôme fixe

$$\mathbf{A(X) = 03.X^3 + 01.X^2 + 01.X + 02,} \quad (1.13)$$

modulo $1 + X^4$:

$$\mathbf{d(X) = A(X) \times c(x) \mod X^4 + 1} \quad (1.14)$$

où

$$\mathbf{c(X) = c_0 + c_1 X + c_2 X^2 + c_3 X^3,} \quad (1.15)$$

et

$$\mathbf{d(X) = d_0 + d_1 X + d_2 X^2 + d_3 X^3}. \quad (1.16)$$

Le polynôme A(X) est premier avec $\mathbf{X^4 + 1}$, il est donc inversible modulo $\mathbf{X^4 + 1}$ est son inverse est

$$\mathbf{B(X) = 0B.X^3 + 0D.X^2 + 09.X + 0E.} \quad (1.17)$$

On retrouve donc **c(X)** à partir de **d(X)** en effectuant le produit

$$\mathbf{c(X) = B(X) \times d(X) \mod X^4 + 1,} \quad (1.18)$$

ou en effectuant le produit matriciel

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \times \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

2.3.4.7. La procédure AddRoundKey

La procédure *AddRoundKey* est très simple. Elle consiste à faire un ou exclusif entre les 128 bits de l'état St et les 128 bits de la clé de tour T . On obtient une nouvelle valeur de l'état.

$St = St \oplus T$.

2.3.4.8. La procédure KeyExpansion

La clé de chiffrement K stockée dans un tableau de 4 lignes et N_k colonnes (N_k = 4, 6, 8) est étendue en un tableau W ayant 4 lignes et $\mathbf{4 * n_r + 1}$ colonnes. La clé de tour T K[i] ($0 \leq i \leq n_r$) est donnée par les 4 colonnes $\mathbf{4 * i, 4 * i + 1, 4 * i + 2, 4 * i + 3}$ du tableau W .

Il y a deux façons de construire le tableau W suivant que N_k = 4, 6 ou N_k = 8. La procédure de construction est nommée *ExpandedKey*.

1) Cas N_k = 4 ou N_k = 6.

Entrée : la clé *K* (sous forme de tableau)

Sortie : le tableau *W*

ExpandedKey (K,W)

Début

Pour (j=0 ; j<N_k ; j++)

Pour(i=0; i<4; i++) W [i,j] = K[i,j];

Pour (j=N_k ; j<4(n_r + 1);j++)

Si(j modN_k==0)

Alors

$$W[0, j] = W[0, j - N_k] \oplus s(W[1, j-1]) \oplus RC[j / N_k];$$

Pour (i=1 ; i<4 ; i++)

$$W[i, j] = W[i, j - N_k] \oplus s(W[i+1 \bmod 4, j-1]);$$

Sinon

Pour (i=0 ; i<4 ; i++)

$$W[i, j] = W[i, j - N_k] \oplus s(W[i, j-1]);$$

Fin si ;

Fin

La procédure utilise la fonction s sur les octets définie précédemment. Elle utilise aussi des constantes de F_{256}, données par

$$RC[i] = \alpha^i;$$

où α est l'élément de F_{256} correspondant au polynôme **X (α = 02).**

L'élévation à la puissance i se fait dans le corps F_{256}.

2) Cas N_k = 8.

Entrée : la clé *K* (sous forme de tableau)

Sortie : le tableau *W*

ExpandedKey (K, W)

Début

Pour (j=0; j<$N_{k;}$ j++)

Pour (i=0; i<4; i++) W [i,j] = K[i,j];

Pour (j=N_k ; j<4(n_r + 1);j++)

Si (j modN_k==0)

Alors

$$W[0, j] = W[0,\ j\ -\ N_k] \oplus s(W[1, j-1]) \oplus RC[j / N_k];$$

Pour (i=1 ; i<4 ; i++)

$$W[i, j] = W[i,\ j\ -\ N_k] \oplus s(W[i+1 \bmod 4,\ j-1]);$$

Sinon si (j mod N_k== 4)

Pour (i=0 ; i<4 ; i++)

$$W[i, j] = W[i,\ j\ -\ N_k] \oplus s(W[i, j-1]);$$

Sinon

Pour (i=0; i<4 ; i++)

$$W[i, j] = W[i,\ j\ -\ N_k] \oplus W[i, j-1];$$

Fin si ;

Fin

2.4. Etapes de chiffrement

2.4.1. Tour initial

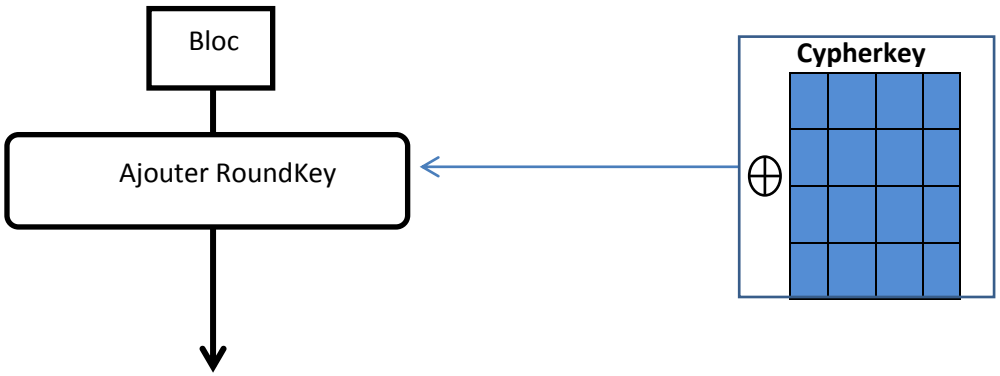


Figure 1.08 *: tour initial du chiffrement*

2.4.2. Tours intermédiaires

1 < T < nr +1

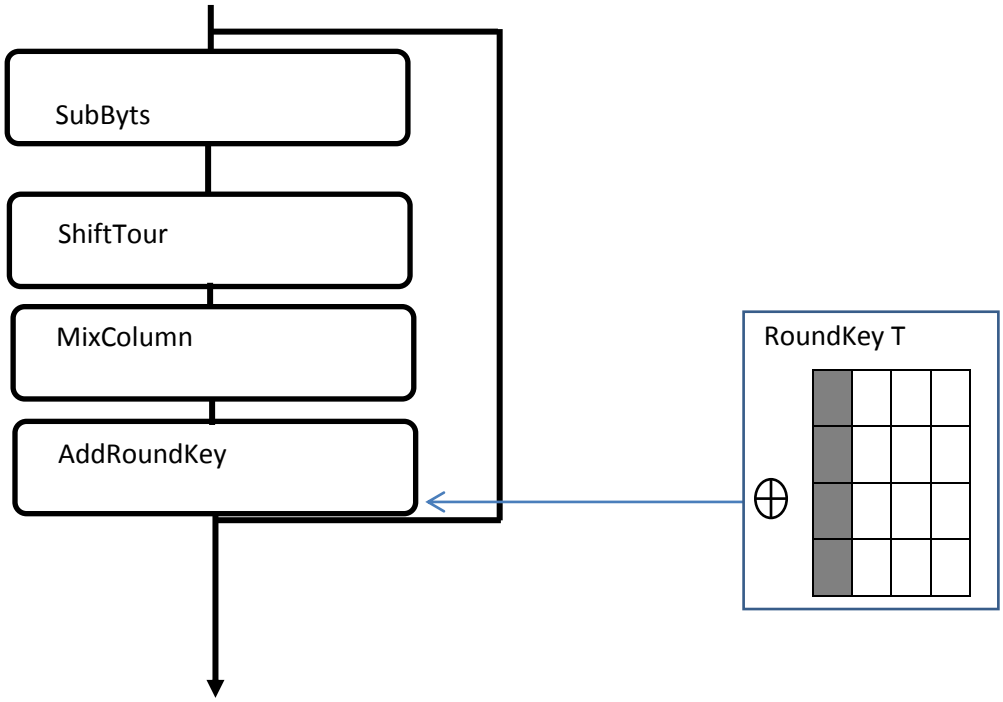


Figure 1.09 *: Tours intermédiaires du chiffrement*

2.4.3. Tour final

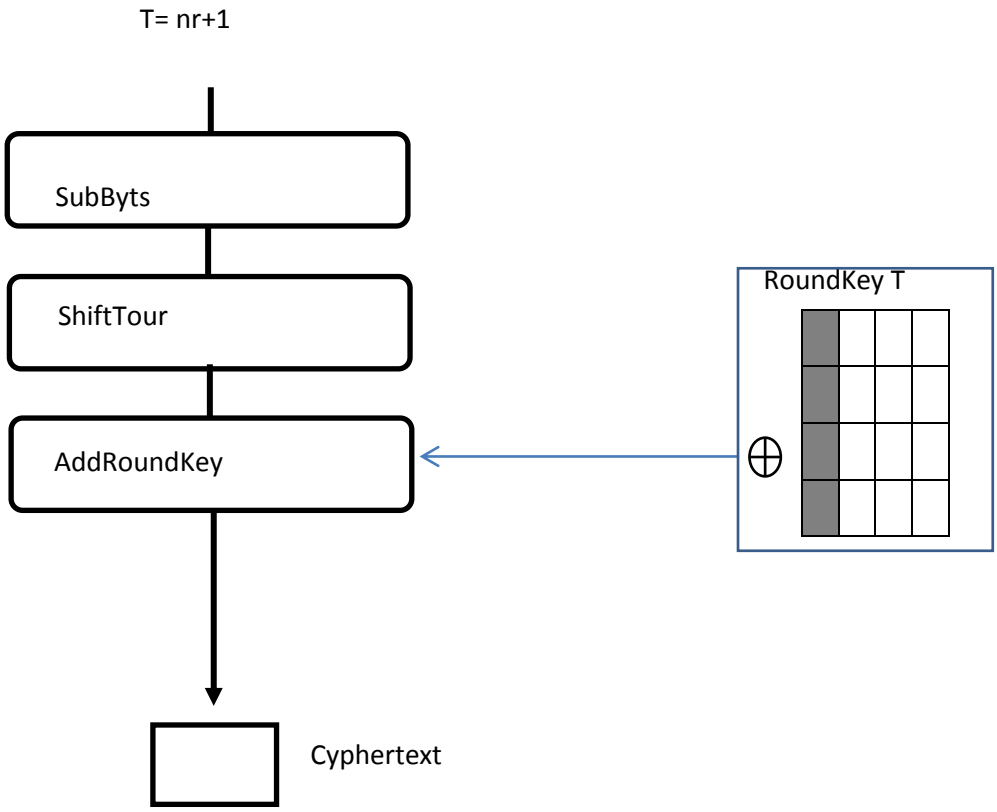


Figure 1.10 : *Tour finale du chiffrement*

3. Conclusion

SIP est un jeune protocole de la couche application qui permet de gérer les sessions multimédia et s'appuie sur des protocoles de transport TCP et UDP. Le protocole possède des mesures de protection interne pour assurer l'intégrité des données transmises. Toutes fois il n'est pas à l'abri de diverse attaques.

Les systèmes de chiffrement symétriques sont constitués, d'un côté, par les chiffrements en flot ou en continue dont le cryptage se fait bit à bit. D'un autre côté, il y a le cryptage par bloc ; les plus répandu de ce dernier groupe sont les méthodes AES et DES. L'AES Rijndael est l'un des chiffrements les plus sûrs au monde.

CHAPITRE 2 : Matériels et outils pré-requis

1. Environnement matériel

Pour le développement du programme j'ai eu à ma disposition les matériels de bureau suivant :

PC de bureau:

Pentium IV

Ram: DDR2 1Go

HDD: 120Go

Carte réseau :

2. Environnement Logiciel

2.1. Operating system

2.1.1. Windows trust

2.1.1.1. Présentation

Microsoft Windows Trust est un système d'exploitation répondant aux exigences d'un développement d'application sous windows. C'est l'un des systèmes de windows autorisant les tweak (modifications des réglages de fonctionnement du système). Cette spécificité rend Windows Trust plus souple en gestion de sécurité et de service [12].

Toutes les solutions logicielles proposées dans Windows Trust ont été entièrement recodées pour vous offrir davantage d'ergonomie et de possibilités de paramétrage.

Windows Trust est épaulé de long en large par des solutions logicielles simples et efficaces qui permettront de gagner un temps considérable lors d'une réinstallation. Tout a été pensé pour simplifier la tâche.

Ces programmes doivent s'exécuter en administrateur.

2.1.1.2. Windows Trust ASO:

L'ASO est un programme de type tweaker qui a pour but de vous simplifier la configuration de Windows Trust. Une foule de possibilités s'offre à vous…

Le principal avantage réside dans la sélection des mises à jour, les composants supprimés ne seront donc pas visés et le WGA ne vous sera pas imposé.

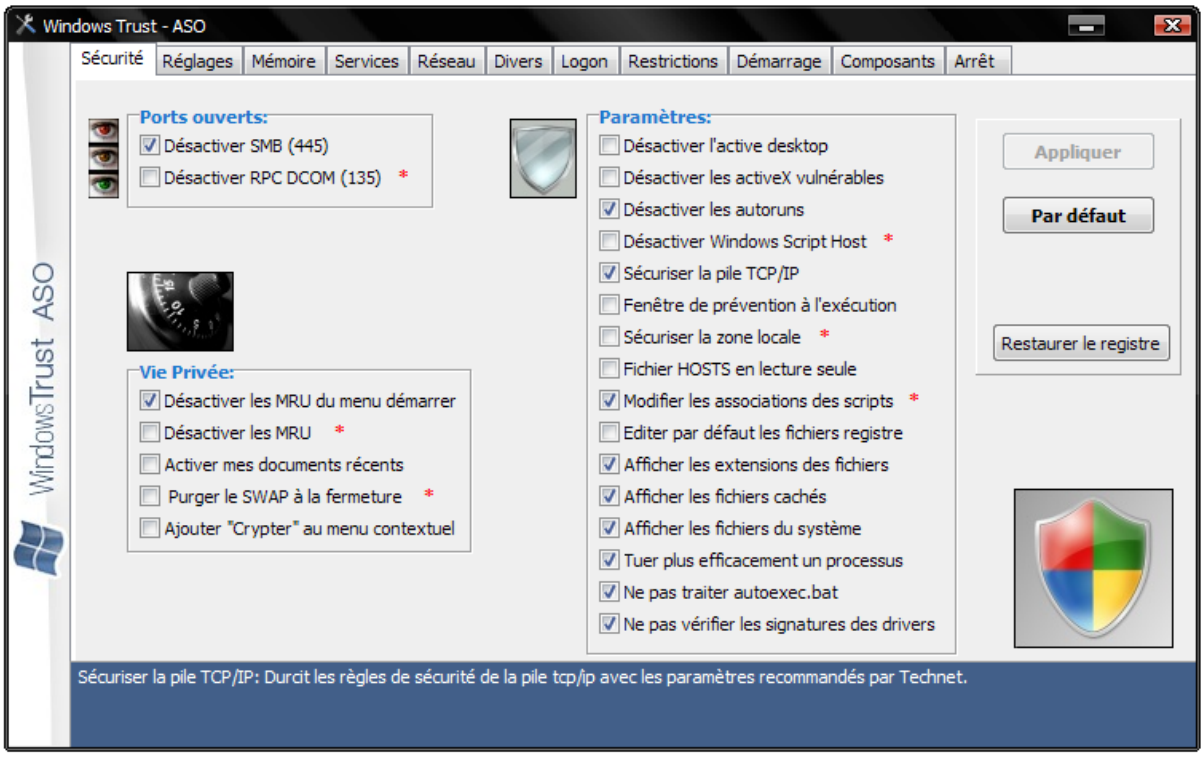


Figure 2.01 : *Fenêtre principale du wASO*

Changer les associations scripts : Associera les fichiers scripts (vbs, jvs, wsh,…) au bloc-notes plutôt que de les exécuter avec Windows Script Host [12].

La modification des associations de scripts permet de bloquer leurs exécutions automatiques. Ce qui aide à la prévention de certaines attaques.

Figure 2.02 *: Configurations disponibles pour les réseaux*

Services de base : Le minimum vital pour pouvoir faire communiquer deux machines au sein d'un réseau.

Ouverture de session réseau : Permet d'authentifier votre machine au sein d'un domaine en entreprise.

NLA : Permet d'évaluer dynamiquement le statut de vos connexions. Il est préférable de le laisser activé.

Support matériel: Activez uniquement le support des périphériques que vous êtes susceptible d'utiliser.

Installation automatique : Permet d'installer sans assistance les logiciels que vous aurez sélectionnés via le patcheur. Très utile pour un déploiement rapide.

Terminal Server : Service indispensable au bon fonctionnement du Bureau à distance.

Services Com+: La plupart des utilisateurs n'en auront que faire, les désactiver engendre cependant des erreurs dans l'observateur d'événements.

Sécuriser la pile TCP/IP conduit à un durcissement des règles de sécurité de la pile TCP/IP avec les paramètres recommandés par Technet.

Fonction Wupdate

Windows Trust Update quant à lui se charge de vous tenir à jour aussi bien au niveau des logiciels intégrés nativement que des mises à jour de sécurité proposées par microsoft. Windows Trust Update est donc une alternative à Windows Update [12].

Si vous avez pris la peine d'activer dans WTUpdate l'option « Conserver les mises à jour », il est possible de les intégrer avec le nouveau patcheur. Des versions actualisées du WTIS seront aussi disponibles au fil des mois.

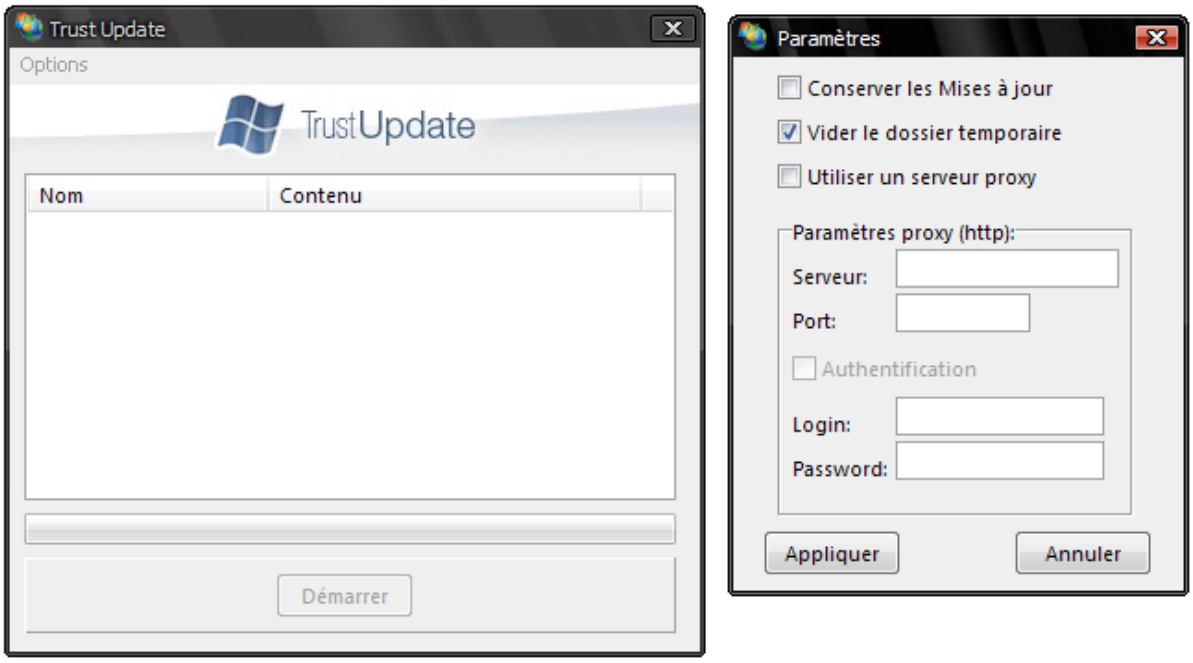


Figure 2.03 *: Présentation du WTUpdate*

2.1.1.3. Process Hacker

C'est un gestionnaire de tache personnalisé pour une meilleure prise en main du fonctionnement des logiciels de PC. Je ne vais citer que les avantages du Process Hacker par rapport au Task Manager de Windows (gestionnaire de tâche par défaut).

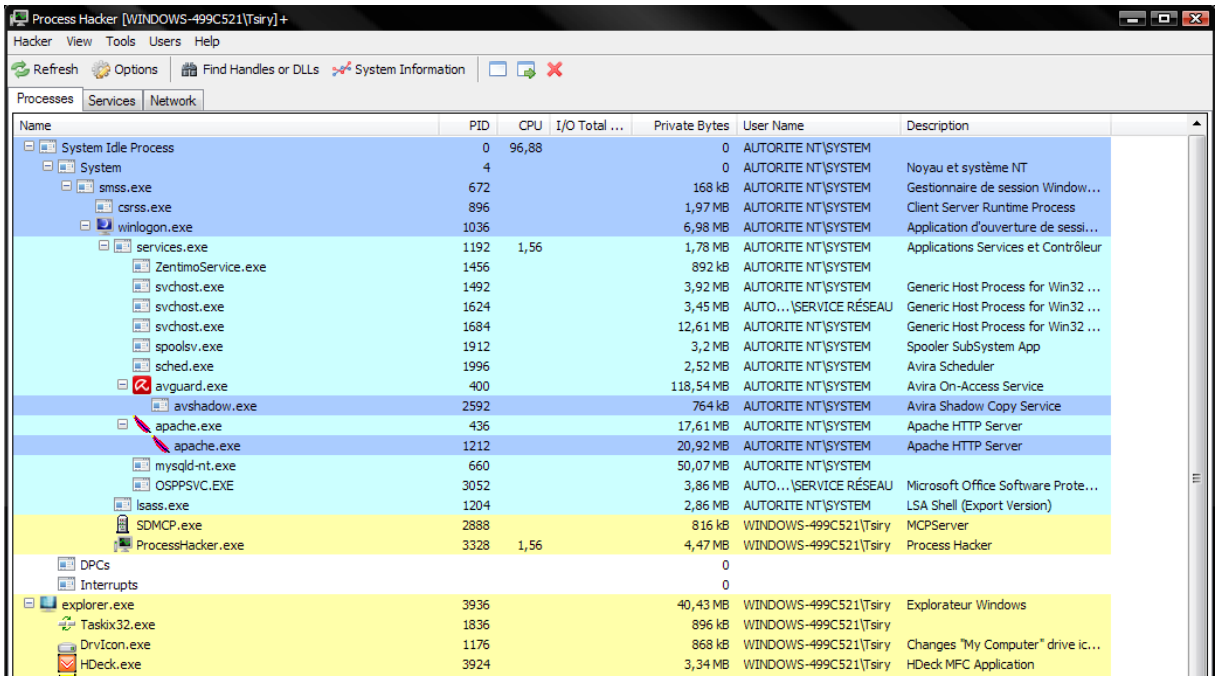


Figure 2.04 *: Présentation du Process Hacker*

Processus

- Permet de voir les processus en arborescence
- Affiche les informations au contexte d'utilisation d'un processus
- Permet de manipuler (stopper, suspendre) plusieurs processus en même temps
- Redémarrage d'un processus
- Présente et offre la possibilité de modifier les descriptions de sécurité d'un processus

Jetons

- Voir tous les détails symboliques, y compris l'utilisateur, propriétaire, le groupe primaire, l'ID de session, l'état de l'altitude, et plus
- Voir les groupes de jetons
- Voir privilèges et même activer, désactiver ou les supprimer
- Afficher et modifier les descripteurs de jeton de sécurité

Modules

- Voir les modules et les fichiers mappés dans une liste
- Décharger les DLL

- Afficher les propriétés de fichiers et de les ouvrir dans l'Explorateur Windows

Mémoire

- Voir une liste de la mémoire virtuelle
- Lire et modifier la mémoire en utilisant un éditeur hexadécimal
- Vider la mémoire dans un fichier
- Libérer de la mémoire ou au dégagement
- Numérisation pour les chaînes

Handles

- Voir les poignées de processus, complets avec mise en évidence pour les attributs
- Recherche pour les poignées (et les DLL et les fichiers mappés)
- fermer les poignées
- (32-bit only) Set handle attributes - Protected and Inherit
- Accès accordé à des poignées peut être considérée symboliquement au lieu de simples nombres hexadécimaux
- Voir les propriétés des objets détaillés lorsqu'elles sont soutenues
- Afficher et modifier les descripteurs de sécurité des objets

Services

- Voir une liste de tous les services
- Créer des services
- Démarrer, arrêter, suspendre, de continuer ou supprimer des services
- Modifier les propriétés du service
- Afficher les dépendances de services et personnes à charge
- Afficher et modifier les descripteurs de sécurité des services

Réseau

- Voir une liste des connexions réseau
- Fermer les connexions réseau
- Utiliser des outils comme whois, traceroute et ping

2.1.1.4. Soft Perfect Network Scanner

SoftPerfect Network Scanner est un scanner à multifonction (IP, NetBIOS et SNMP) avec une interface modem. Le programme scan tout les ports en mode écoute pour les protocoles TCP/UDP et affiche les types de ressources partagé sur le réseau (incluant les ressources caché et les ressources systèmes).

En plus, il vous permet de monter les dossiers partagés en tant que des disques réseau. Ce logiciel permet aussi de contrôler les ports prédéfinis pour un utilisateur et de faire un rapport si celui-ci est ouvert. Il peut aussi résoudre les noms d'hôtes et détecter automatiquement l'IP locale et extérieur du PC utilisé.

- Pings les ordinateurs et affiche ceux qui sont.
- Détecte les adresses MAC, même au travers de routeurs.

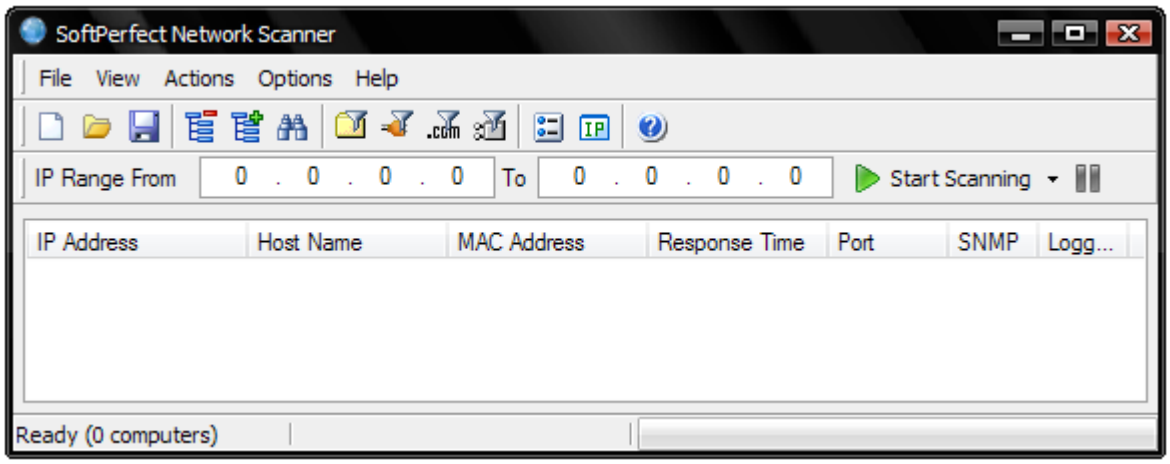


Figure 2.05 *: Interface principale de Soft Perfect Network Scanner*

2.1.1.5. CurrPorts

Currports est un moniteur réseau qui affiche la liste des ports TCP/IP et UDP ouvert sur le PC local. Pour chaque port identifié, les informations concernant le processus qui a ouvert ce port est aussi affiché, incluant le nom du processus, son adresse, les détails (nom du produit, description du fichier, …), le temps d'activité du processus et l'utilisateur qui l'a activé.

Cet application permet aussi de fermer une connexion indésirable et de mettre fin au processus qui a ouvert le port utilisé.

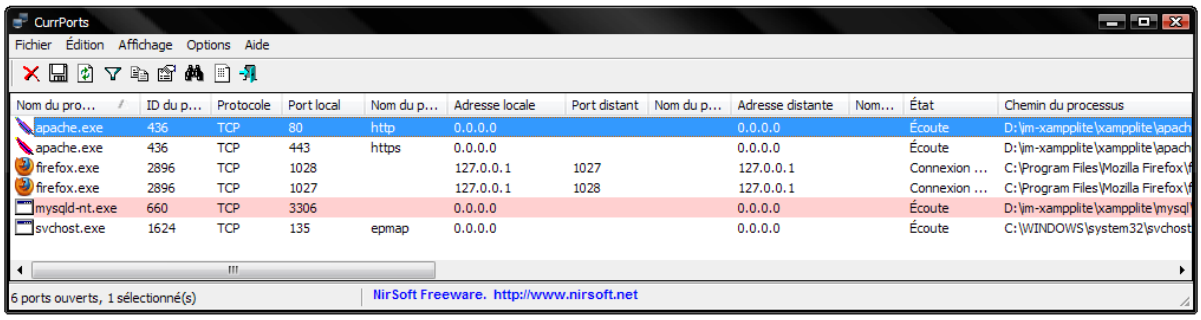


Figure 2.06 *: Interface principale de CurrPorts*

2.1.1.6. Sharp Develop

Sharp Devellop est un environnement de programmation dédié aux scripts windows. Il supporte les langages suivants :

- C# (Code Completion, Windows Forms Designer)
- VB.NET (Code Completion, Windows Forms Designer)
- Boo (Code Completion, Windows Forms Designer)
- IronPython (Code Conversion, Windows Forms Designer, partial Code Completion)
- IronRuby (Code Conversion, Windows Forms Designer)
- F# langage de programmation fonctionnel, impératif et orienté objet pour la plate-forme .NET

Pour information, c'est une application libre il son code source est en C#. Il est distribué sous licence GNU GPL. SharpDevellop est l'un des rivaux du Visual Studio de Microsoft. Contrairement à ce dernier SharpDevellop est une application moins exigeante au niveau matériel. En effet, Sa consommation d'espace mémoire HDD est de l'ordre de 50 Mo contre 500 Mo pour le module de compilation C# de Visual Studio.

En ce qui concerne la configuration générale, les principes ne diffèrent pas vraiment de Visual Studio : d'abord, on crée un fichier appeler solution (ou projet) regroupant tous les informations sur les ressources ainsi que les configurations requis pour l'application que l'on va développer. Ensuite on crée l'interface graphique grâce au concepteur intégré au programme, la manipulation des Frameworks permet de se passer de taper plusieurs lignes de code. Enfin, on écrit les lignes de codes du programme.

Les concepteurs et les Frameworks

- Windows PresentationFoundation (WPF)
- Windows Forms
- ASP.NET MVC
- Entity Framework (EF EDM Designer)

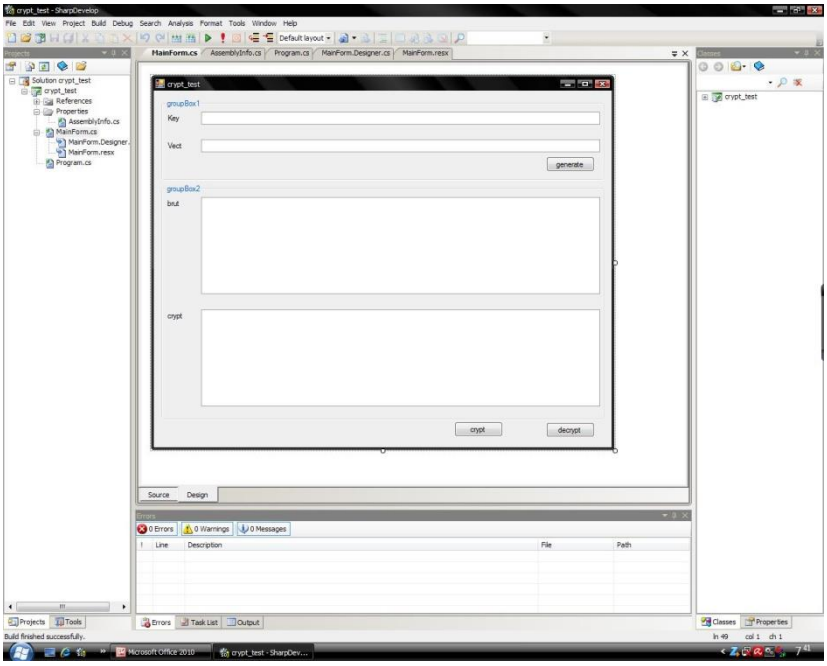


Figure 2.07 *: Résultat de l'interface graphique par le concepteur Framework*

Concernant son gestionnaire de script, le logiciel a la propriété de détecter les erreurs plus ou moins basiques d'un script et de les signaler avant la compilation touts fois il offre à l'utilisateur la possibilité de forcer la compilation dans certaines condition. Certains scripts ne présentent de disfonctionnement qu'à l'utilisation après compilation (erreur d'attribution de variable, …). Pour remédier à ce genre de situation C# est en mesure de simuler une compilation, c'est-à-dire assister le programme même après compilation. Cette fonction permet de connaitre des failles internes au programme qui ne peuvent être détecté qu'après usage du dit programme.

Outils de la qualité

- débogueur intégré (y compris les fonctionnalités de débogage dynamique)

- Analyseur de code (FxCop)
- Testeur d'unité
- Couverture de code
- Profiler
- Prise en charge intégrée Subversion
- Prise en charge intégrée Git
- StyleCopaddin

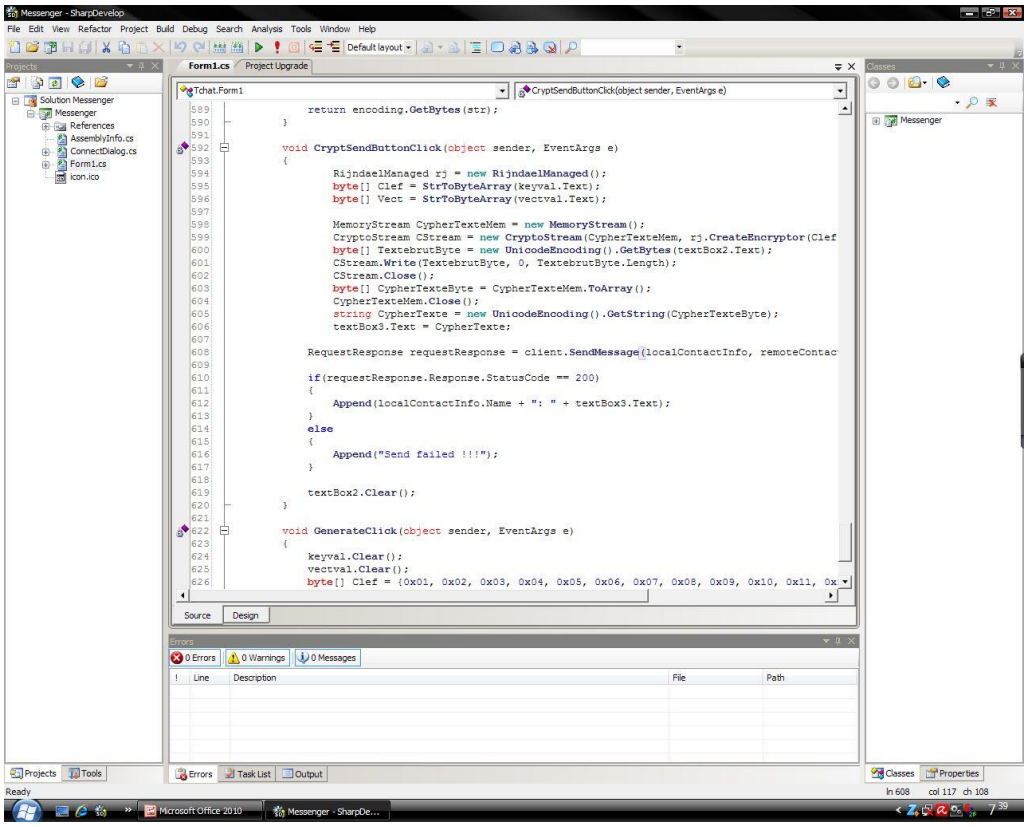


Figure 2.08 *: Editeur de script*

Pour finir Sharp Devellop a la capacité de mettre à jours automatiquement la bibliothèque d'un programme. En effet selon le script du programme il est en mesure de faire passer un programme de la bibliothèque .NETFramework 3.0 à .NETFramework 4.0 sans effets secondaires sur les propriétés du script.

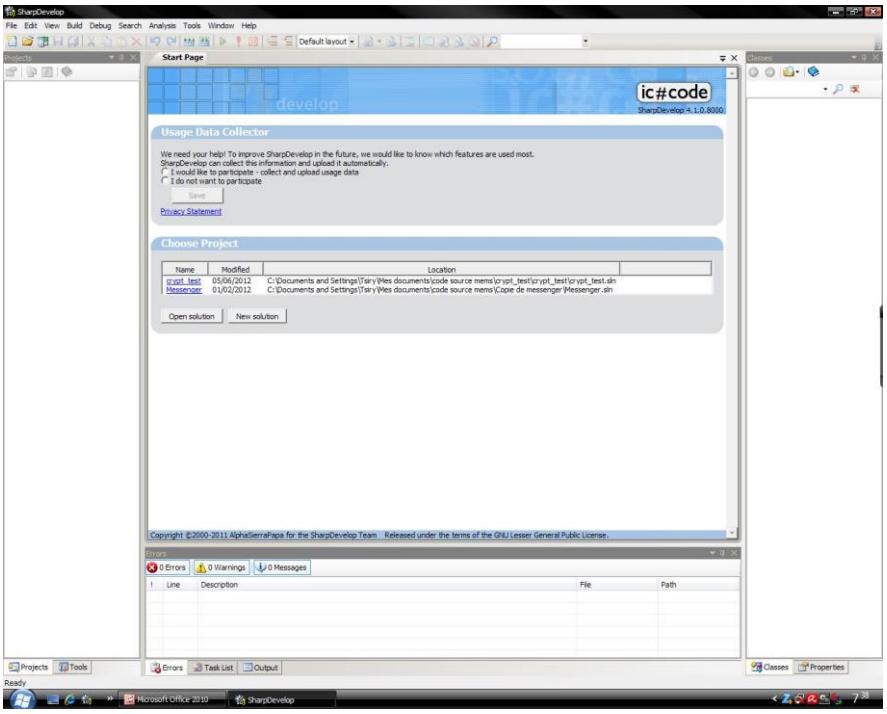


Figure 2.09 *: fenêtre principale de SharpDevelopp*

2.2. Outils de simulation et de vérifications

2.2.1. Asterisk

Asterisk est un autocommutateur téléphonique privé (PABX) open source et propriétaire (publié sous licence GPL et licence propriétaire) pour systèmes UNIX, Mac OS et Windows. Il permet, entre autres, la messagerie vocale, les files d'attente, les agents d'appels, les musiques d'attente et les mises en garde d'appels, la distribution des appels. Il est possible également d'ajouter l'utilisation des conférences par le biais de l'installation de modules supplémentaires et la recompilation des binaires.

Asterisk implémente les protocoles H.320, H.323 et SIP, ainsi qu'un protocole spécifique nommé IAX (Inter-AsteriskeXchange). Ce protocole IAX permet la communication entre deux serveurs Asterisk ainsi qu'entre client et serveur Asterisk. Asterisk peut également jouer le rôle de registrar et passerelle avec les réseaux publics (RTC, GSM, etc.) Asterisk est extensible par des scripts ou des modules en langage Perl, C, Python, PHP, et Ruby [13].

2.2.1.1. Fonctionnalités

Asterisk comprend un nombre très élevé de fonctions permettant l'intégration complète pour répondre à la majorité des besoins en téléphonie. Il permet de remplacer totalement, par le biais de cartes FXO/FXS, un PABX propriétaire, et d'y adjoindre des fonctionnalités de VoIP pour le transformer en

PBX IP. Il permet également de fonctionner totalement en VoIP, par le biais de téléphones SIP ou IAX du marché. Enfin, des fonctionnalités de routage d'appel, menu vocal et boites vocales—entre autres—le placent au niveau des PBX les plus complexes. Au sein des grandes installations d'Asterisk, il est courant de déployer les fonctionnalités sur plusieurs serveurs. Une unité centrale ou plus seront dédiées au traitement des appels et seront épaulées par des serveurs auxiliaires traitant les tâches secondaires (comme une base de données, les boîtes vocales, les conférences).

Des modules tiers permettent de visualiser ou paramétrer le PBX via une interface Flash ou via un client léger.

Enfin, une distribution particulière d'Asterisk, trixbox (anciennement Asterisk@home), est dédiée au PBX léger sur un réseau domestique. Trixbox propose avec deux versions : CE et PRO. trixbox en profite également pour distribuer un PC industriel au format 19" équipé de disques RAID et d'un panneau LCD en face avant [13].

2.2.1.2. Extensibilité

* Aucune latence ;
* Connexions directes de Asterisk ;
* Permet l'intégration de systèmes physiquement séparés ;
* Permet le déploiement d'un plan téléphonique à travers plusieurs bureaux ;
* TDMoE ;
* Utilisation de matériel réseau standard ;
* Voix sur IP / Téléphonie sur IP (VoiP) ;
* Passerelle et terminal Bluetooh ;

2.2.1.3. Interopérabilités

Interopérabilités avec la téléphonie traditionnelle via E&M, E&M Wink, Feature Group D, FXS, FXO, GR-303, Loopstart, Groundstart, Kewlstart, MF, DTMF et Robbed-bit Signaling (RBS) [13].

Asterisk permet aussi l'interopérabilité matérielle avec RTC, RNIS, Wi-Fi, Ethernet, Bluetooth et les cartes de son.

2.2.1.4. Distributions

Plusieurs solutions intégrées basées sur Asterisk sont distribuées par des entreprises : AsteriskNOW (développé par les créateurs d'Asterisk), Trixbox, Fonisk, Elastix, Thirdlane PBX.

2.2.1.5. Fichier de configuration Sip.conf

Rôle

Le fichier sip.conf est utilisé pour configurer les logins et mots de passe de tous les périphériques. Ces périphériques peuvent être des téléphones, des passerelles analogiques ou encore d'autres serveurs. Ce fichier est organisé en différentes zones appelées « context ».

Context générale

Le context général définit :

- Le context par défaut des comptes créés.
- les paramètres TCP/IP du serveur.
- le langage des messages vocaux.

Voici un exemple opérationnel :

```
[general]
context=local                ; context par défaut pour les utilisateurs
bindport=5060                ; port UDP du protocole SIP
bindaddr=0.0.0.0             ; adresse IP de l'interface sur lequel le serveur va écouter le
                             ; trafic 0.0.0.0 pour toutes les interfaces
language=fr                  ; messages vocaux en français
```

Figure. 2.10 *: Context générale*

Context utilisateur

D'autres contexts sont utilisés pour créer des comptes utilisateur. Les paramètres des comptes peuvent être :

- le login
- le mot de passe
- context, ce paramètre permet de gagner de la souplesse dans le routage des appels

- mailbox, ce paramètre est utile pour la messagerie vocale
- c'est avec les paramètres nat et cannreinvite que l'on peut contrer le problème du routage NAT

```
[John]                      ; obligatoire ; login SIP
secret=azerty               ; obligatoire ; mot de passe SIP
callerid="John" <200>       ; facultatif ; nom affiche et numéro affiche sur le
                            ; téléphone de l'appeler
context=local               ; obligatoire ; les appels que fait l'utilisateur
                            ; seront gérés dans le context "local" du fichier
                            ; extension.conf
mailbox=200@default         ; facultatif ; compte de messagerie vocal, voir
                            ; voicemail.conf
type=friend                 ; obligatoire ; autorise les appels entrant et sortant
host=dynamic                ; obligatoire ; adresse IP du client
nat=yes                     ; facultatif ; résoud le probleme de l'enregistrement SIP
                            ; quand le téléphone est derrière un NAT
canreinvite=yes             ; facultatif ; résoud le problème du flux RTP quand le
                            ; telephone est derrière un NAT
```

Figure 2.11 *: Context utilisateur*

Context pour les passerelles

Il existe différentes passerelles. Ces passerelles permettent les communications vers le réseau France Télécom analogique ou numérique mais aussi GSM. Pour pouvoir fonctionner, ces passerelles doivent avoir des comptes. Ces comptes se configurent de la même façon que les comptes utilisateurs, exemple :

```
[SPA-3102-PSTN]
secret=azerty
context=local
type=friend
host=dynamic
```

Figure 2.12 *: Context passerelle*

2.2.2. Microsoft .NET

Microsoft .NET est le nom donné à un ensemble de produits et de technologies informatiques de l'entreprise Microsoft pour rendre des applications facilement portables sur Internet. Le but est de

fournir un serveur web local permettant de gérer des services et évitant d'externaliser des données privées sur un service web de stockage ou un hébergement web tiers.

.NET se base sur plusieurs technologies :

* des protocoles de communication basés sur le Framework .NET et non plus sur les modèles COM ou OLE ;

* une bibliothèque compatible Framework .NET et non plus MFC, GDI… ;

* un environnement d'exécution de code basé sur la CLI multi-langage ;

* MSBuild : un outil de gestion de projet avec plusieurs compilateurs ;

* Visual Studio : un IDE de développement utilisant la méta programmation et compatible avec Visual C++ ;

* Windows Live ID, Framework .NET : un ensemble de bibliothèques de haut niveau ;

* une portabilité pour les systèmes d'exploitation Windows et Windows Mobile ;

* des composants facilitant le développement de services (MapPoint) et d'applications locales ou web (ASP.NET).

Le Framework .NET a été conçu par Anders Hejlsberg, le père de Delphi. Celui-ci a développé entre autres le langage C#.

2.2.2.1. Principales caractéristiques de .NET

a) Interopérabilité

Du fait de la nécessité de pouvoir interagir avec les anciennes applications, le framework fournit des moyens pour accéder aux fonctionnalités en dehors de l'environnement .NET. La possibilité d'accéder aux composants COM est fournie par les espaces de noms System.Runtime.InteropServices et System.EnterpriseServices. L'accès aux autres fonctionnalités est fourni grâce à P/Invoke.

b) Common RuntimeEngine

Les langages de programmation du framework sont compilés dans un langage intermédiaire appelé Common Intermediate Language, ou CIL (anciennement connu sous le nom de Microsoft

Intermediate Language, ou MSIL). Ce langage n'est pas interprété, mais subit une compilation à la volée et une compilation au niveau de la Common Language Runtime (CLR). La CLR est l'implémentation de la CLI.

c) Indépendance du langage

La spécification du Common Type System (ou CTS) définit l'ensemble des types de données et structures de programmation supportés par la CLR ainsi que leurs interactions. Par conséquent, le .NET Framework supporte l'échange des instances des types entre les programmes écrits dans un des langages .NET.

2.2.2.2. Inconvénients

* Microsoft ne fournit le framework .NET que pour ses systèmes d'exploitations tel que Windows, Windows CE et la Xbox 360. Le portage sur d'autres systèmes d'exploitations est possible car les spécifications de la CLI sont disponibles (mais pas des frameworks). Les classes de bases Base Class Library (BCL), sont une partie de la bibliothèque de classes du framework (Framework Class Library ou FCL). La BCL fournit des classes qui encapsulent un certain nombre de fonctions courantes, comme la lecture et l'écriture de fichiers, le rendu graphique, l'interaction avec les bases de données, la manipulation de documents XML, etc. Le code source de la BCL a été rendu disponible pour faciliter le débogage des sessions dans Visual Studio 2008.

* La totalité du code source du framework n'est pas encore disponible. Il est actuellement possible de télécharger une bonne partie de la source du framework.

* .NET ne fonctionnant pleinement que sous Windows, il est très difficile de changer de système d'exploitation ce qui crée une relation de dépendance au seul fournisseur Microsoft. Cependant, le projet Mono essaie de pallier ce problème.

* Les applications fonctionnant avec du code managé en utilisant les environnements tels que le CLR du Framework Microsoft ou la JVM Java ont tendance à nécessiter plus de ressources systèmes que des applications fournissant les mêmes fonctionnalités mais qui accèdent plus directement aux ressources. Cependant, certaines applications ont montré de meilleures performances en utilisant le .NET Framework qu'en utilisant leur version en code natif. Ceci peut être attribué aux optimisations du runtime rendu possible par un tel environnement, à l'utilisation de fonctions performantes au sein du framework, à la compilation à la volée du « managed code » ou encore à certains aspects de la CLR.

* Les langages compilés à la volée produisent du code qui peut être plus facilement rétro-analysé que s'ils étaient écrits en code natif, par conséquent il y a un risque en ce qui concerne la perte de secrets et le risque de passer outre les mécanismes de contrôle de licence. Plusieurs techniques pour rendre le code impénétrable ont déjà été développées pour l'empêcher. Visual Studio 2005 inclut un tel outil (dotfuscator).

* Puisque le framework n'est pas porté sur les anciennes versions de Windows, une application utilisant un framework doit vérifier qu'il est présent, et s'il n'est pas présent, l'application doit guider l'utilisateur pour l'installer. Cette contrainte peut dissuader certains utilisateurs d'utiliser l'application.

* Les versions récentes du framework (3.5 et supérieures) ne sont pas préinstallées quelle que soit la version de Windows. Certains développeurs sont dérangés par la taille importante du framework (environ 54 Mo pour le .NET 3.0 et 197 Mo pour la version 3.5) et par le manque de fiabilité des installateurs. Ce problème est en partie résolu par l'introduction d'une version allégée .Net Client Profile, et qui ne pèse que 26,5 Mo.

3. Langage C#

Le C＃ est un langage de programmation orienté objet à typage fort, créé par la société Microsoft.

Le C＃ est, d'une certaine manière, le langage de programmation qui reflète le mieux l'architecture Microsoft .NET qui fait fonctionner toutes les applications .NET, et en est par conséquent extrêmement dépendant. Les types natifs correspondent à ceux de .NET, les objets sont automatiquement nettoyés par un ramasse-miettes (*garbage collector* en anglais), et beaucoup de mécanismes comme les classes, interfaces, délégués, exceptions, ne sont que des moyens explicites d'exploiter les fonctionnalités de la bibliothèque .NET. Pour achever de marquer cette dépendance, le CLR (*Common Language Runtime*) est obligatoire pour exécuter des applications écrites en C＃, comme l'est la JVM (Java Virtual Machine ou Machine virtuelle Java) pour des applications Java.

Le langage compte un certain nombre de changements par rapport au C/C++ ; On notera particulièrement les points suivants :

- La manipulation directe de pointeurs ne peut se faire qu'au sein d'un code marqué *unsafe*, et seuls les programmes avec les permissions appropriées peuvent exécuter des blocs de code *unsafe*. La plupart des manipulations de pointeurs se font via des références sécurisées, dont l'adresse ne peut être directement modifiée, et la plupart des opérations de pointeurs et d'allocations sont contrôlées

contre les dépassements de mémoire. Les pointeurs ne peuvent pointer que sur des types de valeurs, les types objets, manipulés par le ramasse-miettes, ne pouvant qu'être référencés.

- Les objets ne peuvent pas être explicitement détruits. Le ramasse-miettes s'occupe de libérer la mémoire lorsqu'il n'existe plus aucune référence pointant sur un objet. Toutefois, pour les objets gérant des types non managés, il est possible d'implémenter l'interface *IDisposable* pour spécifier des traitements à effectuer au moment de la libération de la ressource.
- L'héritage multiple de classes est interdit, mais une classe peut implémenter un nombre illimité d'interfaces, et une interface peut hériter de plusieurs interfaces.
- Le C# est beaucoup plus typé que le C++ ; les seules conversions implicites sont celles entre les différentes gammes d'entiers et celles d'un type dérivé à un type parent. Aucune conversion implicite n'a lieu entre booléens et entiers, entre membres d'énumération et entiers, ni de pointeurs sur un type void (quoique pour ce dernier point l'utilisation de références sur le type Object permette d'obtenir le même effet). Les conversions définies par l'utilisateur peuvent être définies comme implicites ou explicites.
- Les membres d'une énumération sont rassemblés dans leur propre espace de noms.
- Le C# ne supporte pas les templates, mais cette fonctionnalité a été remplacée par les types génériques apparus avec C# 2.0.
- Les propriétés ont été introduites, et proposent une syntaxe spécifique pour l'accès aux données membres (ainsi que la facilitation de l'accès simultané par plusieurs threads).
- La réflexion totale des types est disponible.
- Les délégués, qui sont des listes de pointeurs sur fonctions, sont utilisés notamment pour la programmation événementielle.
- le typage dynamique des variables ;
- les paramètres optionnels ;
- la nouvelle bibliothèque parallèle : Task Parallel Library ;
- une version optimisée de la plateforme entité d'accès aux Base de Données (Entity Framework) via l'utilisation de LINQ ;
- L'ajout des mots-clefs select, from et where pour permettre la formation et l'exécution de requêtes SQL, XML, ou directement sur des collections. Cette fonctionnalité fait partie du programme Language Integrated Query (LINQ) .
- Nouvelle possibilité d'initialisation d'un objet ;

- Inférence du type des variables locales : string s = "Dupont" peut être remplacé par var s = "Dupont"
- Introduction des types anonymes ;
- Les arbres d'expressions (expression trees) : permettent la compilation du code sous formes d'arbres d'objets facilement analysables et manipulables.
- Méthodes étendues : permet d'ajouter des méthodes à une classe en y ajoutant un premier paramètre this.
- Les classes partielles, permettant de répartir l'implémentation d'une classe sur plusieurs fichiers.
- Les types génériques, qui ne sont pas une simple copie des templates C++. Par contre, il est impossible d'utiliser des expressions comme paramètres pour la généralisation.
- Un nouvel itérateur qui permet l'utilisation de coroutines via le mot-clé yield, équivalent du yield que l'on trouve en Python.
- Les méthodes anonymes avec des règles de fermeture configurables.
- Les types « nullables », c'est-à-dire la possibilité de spécifier qu'un type de valeur peut être nul. Ceux-ci sont déclarés avec le caractère point d'interrogation « ? » suivant le nom du type, comme ceci : int? i = null;.
- Le nouvel opérateur double point d'interrogation « ?? » utilise deux opérandes et retourne le premier non nul. Il a été introduit pour spécifier une valeur par défaut pour les types « nullables ».

3.1. Differences entre C# et Java:

* Java n'autorise pas la surcharge des opérateurs,

* Java n'a pas de mode unsafe permettant l'arithmétique de pointeurs,

* Java a des exceptions vérifiées, alors que les exceptions du C# ne sont pas vérifiées, comme en C++,

* Java permet la génération automatique de la documentation HTML à partir des fichiers sources à l'aide des descriptions Javadoc-syntax, tandis que le C# utilise des descriptions basées sur le XML,

* C# supporte indexers (indexeurs), delegates (délégué ou liste de pointeurs sur fonctions) et events (événements),

* C# supporte les structures en plus des classes (les structures sont des types valeur : on stocke le contenu et non l'adresse, elles ne peuvent pas contenir des méthodes),

* C# utilise une syntaxe intégrée au langage (DllImport) et portable pour appeler une bibliothèque native, tandis que Java utilise Java Native Interface.

4. Conclusion

Le cadre de travail a été organisé de sorte à permettre au programme de fonctionner dans un environnement à équipement standard. Le serveur ainsi que les modules nécessaires pour le développement et le fonctionnement du programme sont intégré à l'OS Windows SP3 standard.

Le langage de programmation aussi a été choisi dans ce sens pour que le programme dépende le moins possible d'intervention de bibliothèque extérieur.

CHAPITRE 3 : Développement du programme et tests

1. Cadre de travail

1.1. Environnement matériel

Physiquement pour un test effectif il faut avoir en possession 2 PC pour une communication test. Pour les tests que nous avons effectués, on a utilisé une seule machine physique ayant pour fonction : terminal utilisateur A, serveur Asterisk, et support du terminal virtuel de l'utilisateur B.

Cela entraine le fait que l'usage de carte réseau physique est inutile. L'interface réseau utilisé est celle qui relie le terminale A au terminale B donc les interfaces virtuels des B machines sont celles qui entrent en action.

On constate par cela une diminution relative de la puissance de la machine hôte. La puissance de son calculateur est divisée en B : une partie pour le serveur et le terminal A, et l'autre partie pour la machine virtuelle. Il en va de même pour toutes les ressources physiques de la machine physique.

On notera que cette partage de ressource n'affectera pas de façon effective les tests et n'influera pas sur les résultats des mesures.

1.2. Environnement logiciel

Les modules qui doivent être paramétrés pour la simulation sont:

1.2.1. Les interfaces réseau:

- Configurer les adresses IP de chaque terminal

La configuration se fait par l’attribution d’adresse fixe pour chaque terminal à travers les propriétés de protocole internet.

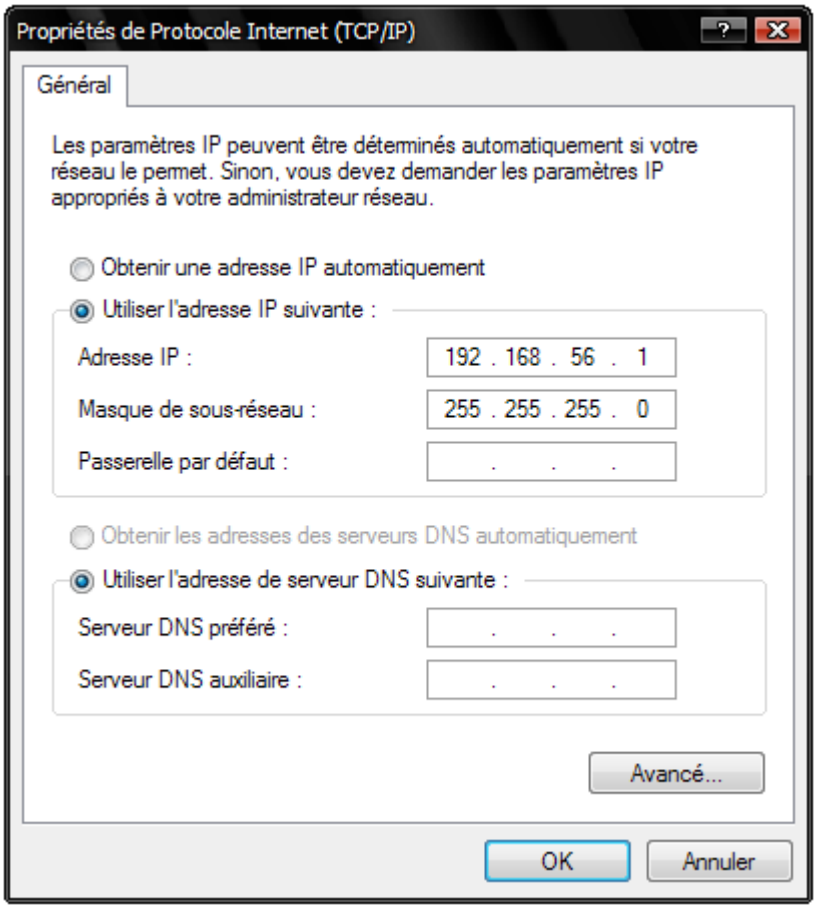


Figure 3.01 *: Propriétés de protocole internet*

- Attribuer un port d'accès pour le protocole SIP (Optionnel)

Par défaut ce port est le port numéro : 5060. Toutes fois il est possible de d'attribuer un autre port au protocole par raison de sécurité.

1.2.2. Le serveur Asterisk:

Il est indispensable d'enregistrer les utilisateurs su le serveur pour être identifiable par la suite.

Cette interface permet de faciliter la manipulation de la base de données du serveur. Tous les paramètres nécessaires sont affiché sous forme de formulaire à remplir et prenant en compte les paramètres tels que les classes d'accès ou les options supplémentaires autorisé pour l'utilisateur.

La figure suivante représente le résultat de l'enregistrement d'un utilisateur sur le serveur.

Toutes fois, nous avons utilisé une version Win32 du serveur, c'est pourquoi nous avons dus configurer directement le fichier de configuration du serveur.

Les étapes à suivre pour cette mode de configurations sont les suivantes:

- Lacer l'application « PBX Manager & Console »,
- Démarrer le PBX
- Vérifier par la commande "sip show peers" la liste des utilisateurs déjà présents sur le serveur et leurs statuts.
- Une fois avoir consulté la liste des peers, déconnecter le PBX
- Arrêter le serveur.
- Ouvrir le fichier « Sip.conf »
- Puis rajouter manuellement les utilisateurs souhaités de la manière suivante
- Puis redémarrer le serveur et reconnecter le
- Ensuite vérifier la liste des utilisateurs actifs
- La transmission brute peut être testée de la manière suivante

1.3. Modèle conceptuelle

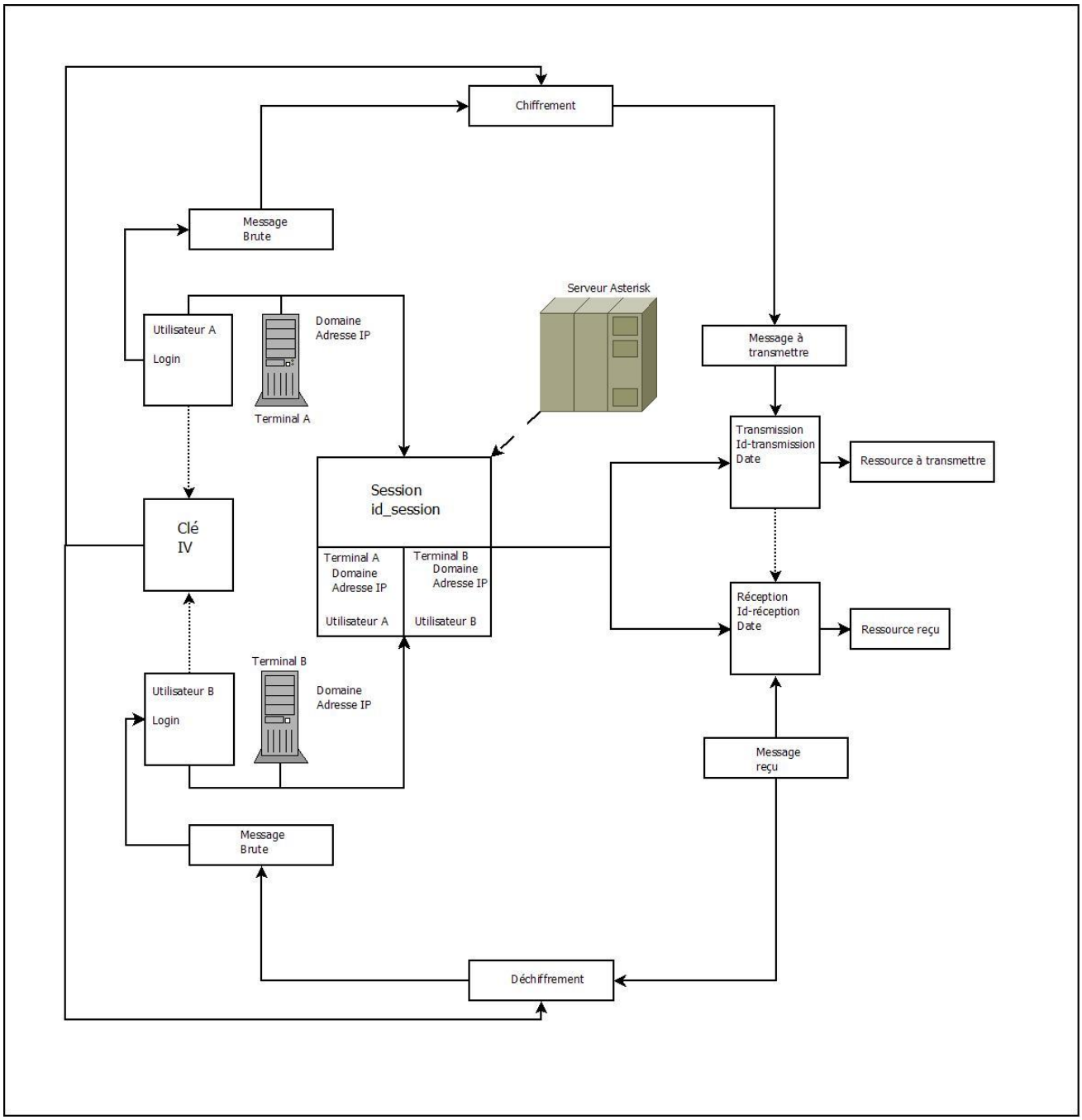


Figure 3.02 *: Représentation logique du concept*

1.4. Les codes

1.4.1. la messagerie

La messagerie possède une base qui peut supporter une liaison directe sans serveur. C'est-à-dire qu'il est possible de relier les terminaux directement sans passer par les étapes usant d'un serveur. Toutes fois pour améliorer la sécurité du réseau il est conseiller d'obliger les utilisateurs de s'identifier au près du serveur SIP locale.

La connexion et l'identification au serveur et à l'hôte distant

```
client
= new SipClient(dialog.RemoteIPAddress, Int32.Parse(dialog.RemotePort), Independentsoft.Sip.ProtocolTyp
e.Udp);
            client.Logger = new Logger("c:\\temp\\" + dialog.LocalName + ".txt");

            client.ReceiveRequest += new ReceiveRequestEventHandler(OnReceiveRequest);
            client.ReceiveResponse += new ReceiveResponseEventHandler(OnReceiveResponse);

            IPHostEntry hostEntry = Dns.GetHostEntry(dialog.LocalIPAddress);

            IPAddress address = hostEntry.AddressList[0];

            if (hostEntry.AddressList.Length > 1)
            {
               for (int i = 0; i < hostEntry.AddressList.Length; i++)
               {
                  if (hostEntry.AddressList[i] != null && hostEntry.AddressList[i].ToString().IndexOf(":") == -1)
                  {
                     address = hostEntry.AddressList[i];
                     break;
                  }
               }
            }
IPEndPoint localIPEndPoint = new IPEndPoint(address, Int32.Parse(dialog.LocalPort));
            client.Bind(localIPEndPoint);
```

Figure 3.03 *: Code source du lancement de la connexion*

Ces lignes résument la connexion de l'application ainsi que l'identification du fichier historique de l'application.

La transmission de message

```
RequestResponse requestResponse =
client.SendMessage(localContactInfo, remoteContactInfo, textBox2.Text);
```

Figure 3.04 *: Transmission de message*

Le message est transmis en texte clair.

Le partage de ressource

```
Socket sendFileSocket
= new Socket(AddressFamily.InterNetwork, SocketType.Stream, System.Net.Sockets.ProtocolType.Tcp);

        int remoteFileTransferPort = responseSessionDescription.Media[0].Port;
        IPEndPoint remoteIPEndPoint
= new IPEndPoint(Dns.Resolve(remoteIPAddress).AddressList[1],remoteFileTransferPort);

        FileStream file = new FileStream(fileName,FileMode.Open);
        byte[] buffer = new byte[file.Length];
        file.Read(buffer,0,buffer.Length);

        sendFileSocket.Connect(remoteIPEndPoint);
        sendFileSocket.Send(buffer);
        sendFileSocket.Close();

        file.Close();
        Append("File transfer completed.");
```

Figure 3.05 *: Partage de ressource*

1.4.2. le chiffrement

Le chiffrement est un chiffrement Rijndael de 256 bits, réputé pour sa résistance aux attaques.

La génération de la clé et du vecteur initial IV

```
void GenerateClick(object sender, EventArgs e)
      {
         keyval.Clear();
         vectval.Clear();
         byte[] Clef
= {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16};
         byte[] Vect
= {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16};
         System.Text.Encoding enc = System.Text.Encoding.ASCII;
         keyval.Text = enc.GetString(Clef);
         vectval.Text = enc.GetString(Vect);

      }
      public static byte[] StrToByteArray(string str)
      {
         System.Text.UTF8Encoding  encoding=new System.Text.UTF8Encoding();
```

Figure 3.06 *: Générateur par défaut de vecteur initial et de clé*

La cryptographie utilisée est symétrique ; c'est à dire qu'elle ne nécessite qu'une clef privée, qui est la même pour le cryptage et le décryptage. C'est pourquoi nous déclarons un tableau de "byte" nommé Clef, qui contient la clef de 16 octets que l'algorithme de cryptage utilisé nécessite. Les octets doivent être écrits en hexadécimal, sous la forme 0xAB ou A et B représentent des caractères hexadécimaux (de 0 à F).Vous pouvez donc les modifier, puisqu'il s'agira de votre clef.

Egalement nécessaire à l'algorithme, le "vecteur d'initialisation" intervient comme opérande de la fonction XOR réalisée par l'algorithme avec la Clef et les données. Il est aussi unique et privé, vous pouvez le modifier. Il est écrit en hexadécimal.

Le logiciel est toutes fois en mesure de fournir une clé et un vecteur initial.

L’algorithme de chiffrement

```
MemoryStream CypherTexteMem = new MemoryStream();
        CryptoStream CStream
= new CryptoStream(CypherTexteMem, rj.CreateEncryptor(Clef, Vect),    CryptoStreamMode.Write);
        byte[] TextebrutByte = new UnicodeEncoding().GetBytes(textBox2.Text);
        CStream.Write(TextebrutByte, 0, TextebrutByte.Length);
        CStream.Close();
        byte[] CypherTexteByte = CypherTexteMem.ToArray();
        CypherTexteMem.Close();
        string CypherTexte = new UnicodeEncoding().GetString(CypherTexteByte);
```

Figure 3.07 *: Algorithme de chiffrement*

L'algorithme de déchiffrement

```
MemoryStream CypherTexteMem = new MemoryStream(new UnicodeEncoding().GetBytes(text));
        CryptoStream CStream
= new CryptoStream(CypherTexteMem, rj.CreateDecryptor(Clef, Vect),CryptoStreamMode.Read);
        MemoryStream TextebrutMem = new MemoryStream();
        do
          {
            byte[] buf = new byte[100];
            int BytesLus = CStream.Read(buf,0,100);
            if (0 == BytesLus)
                break;
              TextebrutMem.Write(buf,0,BytesLus);

          }while(true);

        CStream.Close();
        CypherTexteMem.Close();
        byte[] TextebrutByte = TextebrutMem.ToArray();
        TextebrutMem.Close():
```

Figure 3.08 : *Déchiffrement*

1.5. Réalisation

Le programme fonctionne de sorte à ce que le chiffrement et la transcription des messages se fait facultativement. Chaque message peut être chiffré indépendamment, la clé ainsi que le vecteur d'initialisation peuvent être modifié au cours de la discussion en cas de soucis de sécurité.

Le module de chiffrement est indépendant du module de traduction. Il est donc possible de transmettre un message avec une clé K et de déchiffrer un autre message reçu par une clé K' différente de K.

2. Objectif

Mes travaux porteront sur la création d'une application de messagerie instantanée sécurisée. Nous allons utiliser un protocole récent et encore très peut exploiter. Le programme devrai être en mesure de prioriser l'intégrité de la transmission au mieux possible pour assurer l'authenticité des données transmises. Cette application est à l'usage local pour réduire les problèmes d'intrusion dans les sessions.

2.1. Capacité fonctionnelle

Fonctionnement en réseau locale ou globale.

Spécifications des caractéristiques de transmission

Messages indépendantes

Partage de ressources

Sécurisation supplémentaire par chiffrement de message

2.2. Facilité d'utilisation

Les options fournis par le programme sont pour la majorité inscris sur la fenêtre principale.

Le partage de ressource se fait facilement.

Le programme demande une seule fois l'autorisation d'outre passer les pare-feu du système.

2.3. Rendement

Le temps que prend le traitement des informations est moins d'une seconde toutes fois le partage de ressource dépend du trafic générale du réseau.

2.4. Portabilité

Le programme ne dépend que de la bibliothèque principale de Microsoft Windows, bibliothèque intégré au système à partir du Service Pack 3.

L’application ne nécessite aucune installation.

Il ne modifie pas le registre du système en ajoutant automatiquement son entrée.

3. Principe de base

La messagerie instantanée diffère du courrier électronique du fait que les conversations se déroulent instantanément (quasiment en temps réel, les contraintes temporelles n’étant pas fortes dans ces systèmes).

Dans les tout premiers programmes de messagerie instantanée, chaque lettre apparaissait chez le destinataire dès qu’elle était tapée, et quand des lettres étaient effacées pour corriger une faute, cela se voyait également en temps réel. Cela faisait ressembler la communication à un coup de téléphone plutôt qu’à un échange de messages. Dans les programmes modernes de messagerie instantanée, le destinataire ne voit le message de l’expéditeur apparaître que lorsque celui-ci l’a validé.

La plupart des applications de messagerie instantanée permettent de régler un message de statut, qui remplit la même fonction qu’un message de répondeur téléphonique, par exemple pour indiquer la cause d’une indisponibilité.

Le paysage des systèmes de messagerie instantanée est arrivé à un morcellement et une fragmentation tels que les utilisateurs de réseaux propriétaires et fermés sont dans l’incapacité de communiquer avec les autres réseaux et protocoles : ils sont enfermés et ne peuvent plus en sortir à cause de l’effet réseau (il leur faudrait basculer tous leurs contacts vers un réseau et protocole standard ouvert).

On assiste à un cloisonnement extrême qui ne s’est quasiment jamais vu dans aucun autre domaine : trois grands réseaux propriétaires sont utilisés par plusieurs dizaines ou centaines de millions d’utilisateurs ; ils sont enfermés et ne peuvent pas communiquer avec les centaines de millions d’utilisateurs des autres réseaux.

3.1. Principe d'accès

Un utilisateur voulant bénéficier de l’accès au réseau doit s’identifier au préalable.

D’une part, envoyer les informations concernant l’utilisateur local pour la reconnaissance du serveur. Cela permet à la fois de s’identifier et de valider sa présence sur le réseau.

D’autre part, envoyer les informations du correspondant distant pour une vérification si celui-ci existe et en état de recevoir une requête. C’est la partie Invite du mécanisme du protocole SIP qui demande l’autorisation au correspondant de se connecter.

Les informations utiles pour cette identification sont les informations basiques d’identité d’un client de réseau informatique. Ce sont, pour une partie, des données statiques qui doivent être disponible sur le serveur du réseau : le login et l’adresse de la boite de messagerie ; l’autre partie est faite de donnée dynamique et changent surtout en fonction des terminaux en connexion : l’adresse IP et le numéro de port utilisé. Ces dernières informations sont configurable (fixé ou paramétré) à l’avance, permettant une exclusivité de canal de transmission.

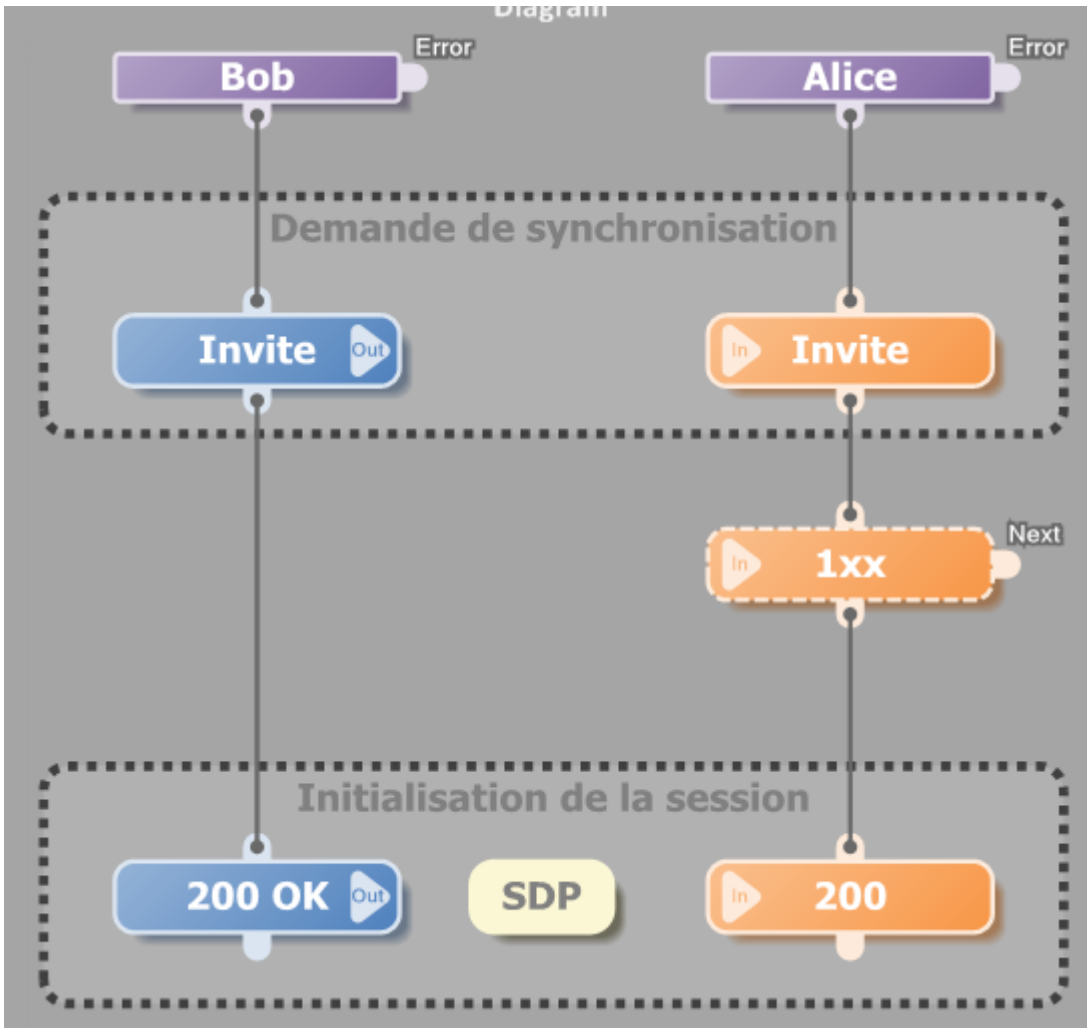


Figure 3.09 : *diagramme D’initiation de session*

Ce diagramme représente les étapes importantes d’une requête d’ouverture de session :

Nous pouvons constater que la transmission de la réponse de l'invité est transmise par un protocole SDP pour assurer la bonne réception du message.

Chaque utilisateur se fait attribuer un "level" par l'administrateur du serveur pour définir les droit de chaque utilisateurs et ses caractères supplémentaires tel que les adresses IP et le numéro de port.

Les "levels" sont modifiables, c'est à dire que l'accès aux options, aux modes et à d'autres caractéristiques peuvent être redéfinis. Par exemple il est possible de faire un canal où aucun utilisateur n'a le droit de partager des ressources.

Notons que dans la majorité des messageries instantanées chaque utilisateur possède un paramètre "statut" qui définit l'activité de l'utilisateur :

Actif : utilisateur connecté et actif (ou inactif depuis un délai prédéfini),

Occupé/absent : utilisateur connecté mais indisponible,

Déconnecté : un utilisateur non connecté sur le réseau

De cette manière chaque utilisateur peu connaitre si l'utilisateur qu'il désire joindre est disponible ou pas.

3.2. Principe de discussion

Une fois l'étape de la connexion achevée, il est désormais possible d'émettre et de recevoir des messages.

Dans la messagerie instantanée, c'est la partie la plus vulnérable du système ; chaque message est un courrier électronique indépendant et complet qui contient toutes les informations générales des interlocuteurs ainsi que les messages transmis.

Les messages sont transmis après validation de l'utilisateur. Les entrée simultanée sont quasi impossible donc il y a toujours un ordre d'arrivé pour les messages. Durant la discussion il est possible de partager des éléments multimédia comme des images. Dans les deux cas le programme affiche les messages ainsi qu'un aperçu des fichiers envoyé sur l'interface de l'ôte locale. Cette disposition à été prise pour permettre à l'utilisateur d'être sur que l'ôte distant aie bien reçu la transmission.

Une discussion peut être considérée comme un canal privé, inaccessible à tout individu sauf les interlocuteurs. Ce canal est sécurisé et n'est libéré qu'une fois la session terminé.

Une session et les informations concernant une session sont présentes sur le serveur ainsi que les machines des utilisateurs durant la session.

3.3. La déconnection

La déconnection est la terminaison de la session dans ce cas précis une session ne se termine que lorsque les utilisateurs se déconnectent du réseau. Si l'un des utilisateurs reste connecté après la session, les informations ainsi que touts les messages transmis durant la session sont toujours présent dans la machine de ce dernier. Les fichiers caches ne sont effacé que lorsque l'utilisateur passe son statut à déconnecté.

C'est la partie qui se charge du nettoyage du système et détruit toutes les informations concernant la session qui a été ouverte.

4. Fonctionnement sans chiffrement

Le programme fonctionne en mode non sécurisé par défaut. Ce mode illustre le fonctionnement de base d'une messagerie instantanée.

4.1. Connexion :

C'est la première étape qu'un utilisateur « enregistré » doit faire pour pouvoir accéder aux services qu'il dispose. Cette étape est cruciale pour la suite car ceci est l'étape d'identification des utilisateurs.

Une fenêtre est regroupe toutes les données nécessaires pour initier une session, il nécessite de remplir un formulaire qui sera transmise au serveur pour validation.

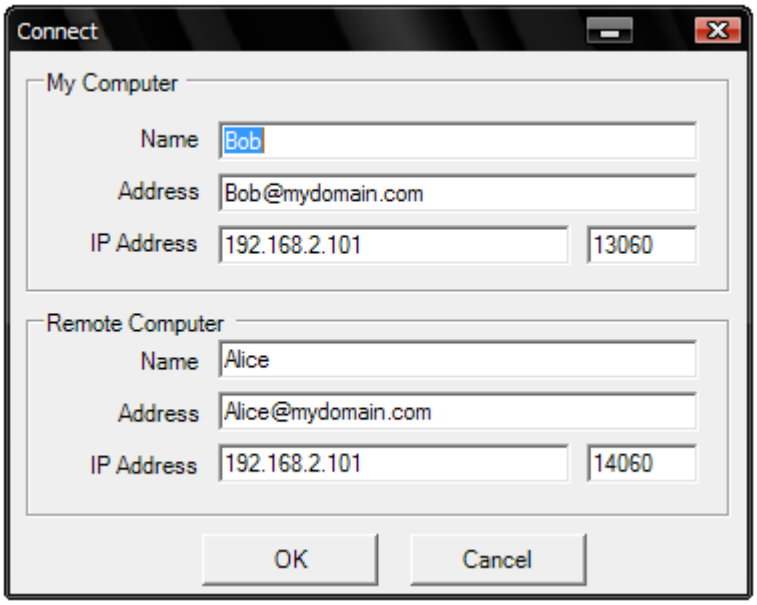


Figure 3.10 *: Interface de connexion*

Les données spécifiques des utilisateurs doivent être présentes pour permettre au serveur d'autoriser l'utilisateur local de contacter un autre utilisateur ayant une identité spécifique sur ce même serveur.

- ✓ le login
- ✓ le mot de passe
- ✓ context, ce paramètre permet de gagner de la souplesse dans le routage des appels
- ✓ mailbox, ce paramètre est utile pour la messagerie vocale

```
INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[call_number]
To: sut <sip:[service]@[remote_ip]:[remote_port]>
Call-ID: [call_id]
CSeq: 1 INVITE
Contact: sip:sipp@[local_ip]:[local_port]
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length: [len]
```

Figure 3.11 *: Forme de l'entête de message*

[service]: c'est l'identifiant ou le login de l'utilisateur,

[remote_ip]/ [local_ip]: le domaine de l'utilisateur distant/locale (adresse IP),

[remote_port]/ [remote_port]: le port de transmission de l'utilisateur distant/locale,

[transport]: le mode de transmission (exemple : SDP),

[call_id]: numéro identification de l'appel,

C'est la partie du script qui prend en charge l'identification des utilisateurs. C'est la forme générale mais au cours des utilisations s'ajoutent certains paramètres.

```
[last_Via:]
[last_From:]
[last_To:];tag=[call_number]
[last_Call-ID:]
[last_CSeq:]
```

Figure 3.12 *: Identification de la transmission précédente*

Ce sont les paramètres des requêtes précédentes. Pour ne pas avoir à commettre des erreurs de collision de transmission.

Suit enfin, les paramètres optionnels :

```
v=0
o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
s=-
c=IN IP[media_ip_type] [media_ip]
t=0 0
m=audio [media_port] RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure 3.13 *: paramètres optionnels*

Ce sont les paramètres optionnels accordés à l'utilisateur, ces paramètres sont configurés grâce au *level* de l'utilisateur.

Une fois identifié par le serveur l'application crée un fichier destiné à stocker les transmissions à suivre par l'utilisateur. L'emplacement de ce fichier est « C :\Temp ». L'application crée un fichier, portant le nom de l'utilisateur, pour chaque utilisateur qui s'authentifie au travers du programme.

4.2. Création de session

La création de session est l'étape après l'identification, elle est la partie qui valide les requêtes de celui qui invite et par cela ouvre un canal fixe décri par les paramètres envoyer durant la procédure d'invitation : login, adresses IP et numéro de port des 2 utilisateurs

4.3. Les transactions entre les utilisateurs durant la session

Une fois la session validée, les utilisateurs peuvent commencer à émettre et à recevoir des messages (texte ou fichier multimédia). Rappelons que tous les transactions effectué sont répertorié dans un fichier cache.

Chaque message est de la forme suivante :

```
MESSAGE sip:Alice@localhost SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:2525;branch=z9hG4bK1708947935
Max-Forwards: 70
From: "bob" <Bob@localhost>;tag=1917535849
To: "alice" <Alice@localhost>
Call-ID: 361006194-1902036624-1830878950
CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 7
User-Agent: SIP .NET 1.

Warning
```

Figure 3.14 *: Exemple de message SIP*

Cet exemple illustre la représentation de l'historique d'une transaction émise par l'utilisateur "bob" vers l'utilisateur "Alice".

Nous pouvons constater que les informations générales des interlocuteurs sont transmises en textes clairs

5. Fonctionnement avec chiffrement

La modification apportée est le chiffrement du message par un encodeur AES Rijndael. Cette méthode permet de sécuriser le message car comme nous l'avons constaté le message est transmis en texte clair et peut être intercepté facilement.

Le chiffrement est un module à la disposition de l'utilisateur en cas de transmission d'information sensible.

5.1. Etapes du chiffrement

Le schéma suivant décrit succinctement le déroulement du chiffrement :

- BYTE_SUB (Byte Substitution) est une fonction non-linéaire opérant indépendamment sur chaque bloc à partir d'une table dite de substitution.

- SHIFT_ROW est une fonction opérant des décalages (typiquement elle prend l'entrée en 4 morceaux de 4 octets et opère des décalages vers la gauche de 0, 1, 2 et 3 octets pour les morceaux 1, 2, 3 et 4 respectivement).
- MIX_COL est une fonction qui transforme chaque octet d'entrée en une combinaison linéaire d'octets d'entrée et qui peut être exprimée mathématiquement par un produit matriciel sur le corps de Galois.
- le + entouré d'un cercle désigne l'opération de OU exclusif (XOR).
- Ki est la ième sous-clé calculée par un algorithme à partir de la clé principale K.

Le déchiffrement consiste à appliquer les opérations inverses, dans l'ordre inverse et avec des sous-clés également dans l'ordre inverse.

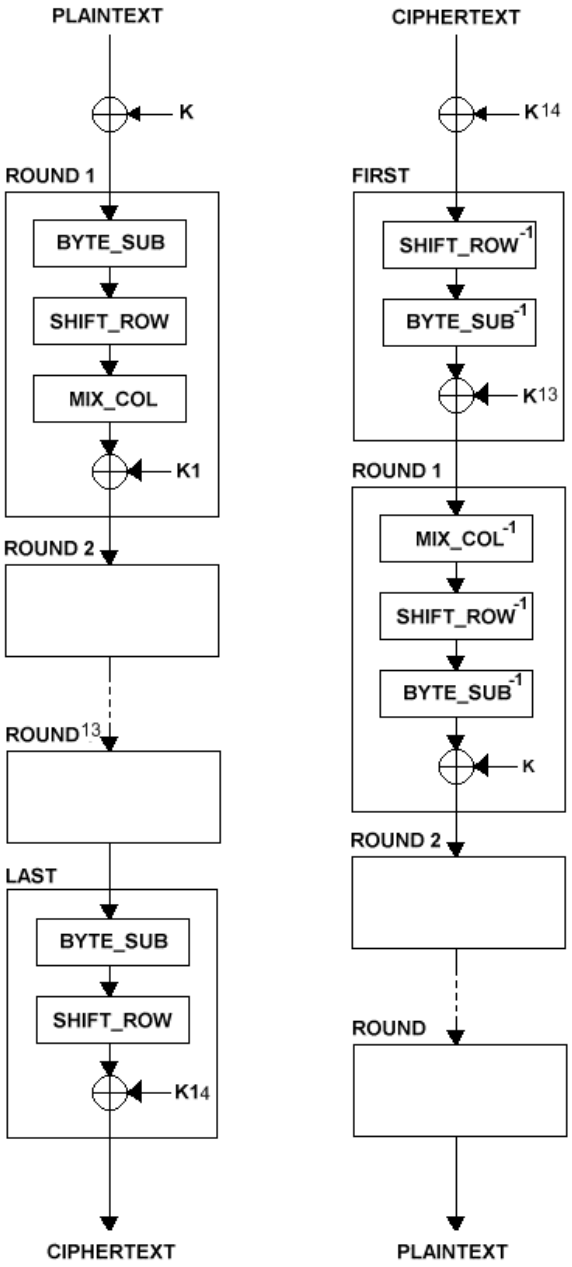


Figure 3.15 *: Etapes de chiffrement et de déchiffrement*

5.2. Génération de clé et de vecteur initial (IV)

En générale une clé de cryptage est de longueur minimale de 4 caractères pour une sécurité minimale des données à transmettre

Comme nous l'avons constaté précédemment la clé doit être manipulée au préalable avant son utilisation.

C’est la clé par défaut proposé par le programme. Cette clé par défaut assure une sécurité maximale puisqu’elle est de 256 bits.

6. Résistance du cryptage

Le choix de cet algorithme répond à de nombreux critères plus généraux dont nous pouvons citer les suivants :

- sécurité ou l'effort requis pour une éventuelle cryptanalyse.
- facilité de calcul : cela entraine une grande rapidité de traitement
- besoins en ressources et mémoire très faibles
- flexibilité d'implémentation: cela inclut une grande variété de plateformes et d'applications ainsi que des tailles de clés et de blocs supplémentaires.
- hardware et software : il est possible d'implémenter l'AES aussi bien sous forme logicielle que matérielle (câblé)
- simplicité : le design de l'AES est relativement simple

Si l'on se réfère à ces critères, on voit que l'AES est également un candidat particulièrement approprié pour les implémentations embarquées qui suivent des règles beaucoup plus strictes en matière de ressources, puissance de calcul, taille mémoire, etc...

7. Test de transmission avec chiffrement

Nous avons pris pour ce test la clé et le vecteur d’initialisation par défaut proposé par le logiciel pour sécuriser la transmission.

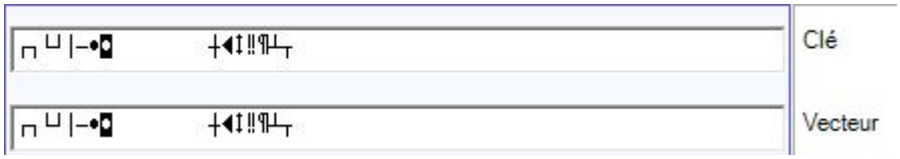
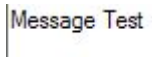
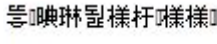


Figure 3.16 *:* *Clé et vecteur d’initialisation*

Nous avons ensuite intercepté le message lors de sa transmission. Les représentations suivantes sont celle du message brut (avant chiffrement) *Figure 3.17 (a)* et celle du message chiffré (message intercepté) *Figure 3.17 (b).*



(a)



(b)

Figure 3.17 : *Message brut et message crypté*

8. Conclusion

Les tests et simulation nous a permis de déduire que le programme peut fonctionner en réseau locale que globale. Il peut fonctionner grâce à un serveur SIP quelconque même si celui-ci est virtuellement installer sur l'un des machines client.

Le chiffrement des messages intégré au logiciel réduit les risques d'interception de messages pour les transmissions de données sensibles.

CONCLUSION

Il est plus rassurant de savoir que les données que nous faisons circuler via le net ne peuvent tomber entre les mains d'autre personne que le destinataire. Pourtant ce risque existe, alors nous dissimulons les données de diverse manière pour faire en sorte que même si le message soit intercepté il soit, du moins, très difficile de le lire.

Nous ne pouvons pas avoir confiance au réseau global, qui possède des moyens de sécurisation rudimentaires. Ces réseaux mettent nos données à disposition de toute personne ayant un minimum de connaissance en informatique et réseau.

Nous proposons de mettre en place une sécurisation personnalisé, indépendant de toute intervention extérieur. Mettre à disposition une ligne de transmission dynamique entre 2 interlocuteurs, avec la possibilité de chiffrer les messages dans la mesure où leurs discussions sont sur écoute.

Au vu de l'évolution des techniques de cryptanalyse de simple chiffrement ne seront plus suffisant pour sécuriser les informations numériques que nous transmettons. Mais elle reste un moyen réaliste de sécuriser aux mieux nos données.

ANNEXE 1 : BASES ET ARCHITECTURE TCP/IP

L'architecture TCP/IP comprend de nombreux programmes applicatifs, utilitaires et protocoles complémentaires (*Figure A1.01)*. À l'origine TCP/IP ne spécifiait aucun protocole de ligne, il s'appuyait sur les réseaux existants. L'utilisation massive de TCP/IP a fait apparaître des réseaux tout IP et la nécessité de disposer d'un protocole de liaison (SLIP, PPP). De même, TCP/IP a été adapté aux protocoles dits « Haut Débit » comme le Frame Relay et l'ATM qui constituent aujourd'hui le cœur de la plupart des réseaux privés et d'opérateurs.

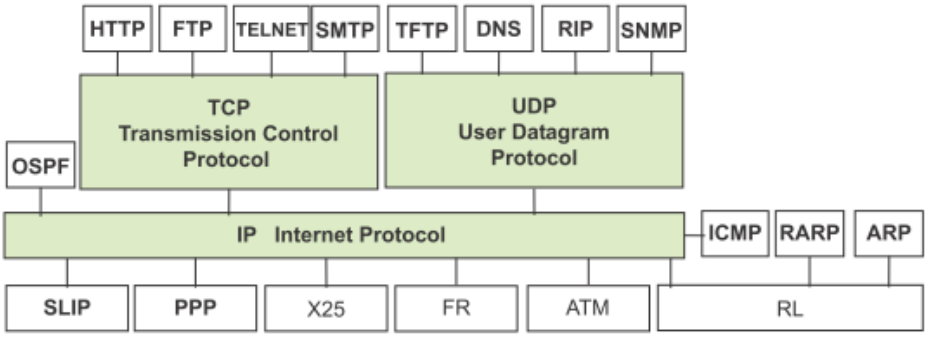


Figure A1.01 *: Protocoles et applications de TCP/IP.*

Les principaux protocoles et applications de l'environnement TCP/IP sont :

– HTTP, HyperText Transport Protocol, assure le transfert de fichiers hypertextes entre un serveur Web et un client Web ;

– FTP, File Transfer Protocol, est un système de manipulation de fichiers à distance (transfert, suppression, création...) ;

– TELNET, TELetypewriter NETwork protocol (ARPA) ou TERminaL NETwork protocol, système de terminal virtuel, permet l'ouverture de sessions avec des applications distantes ;

– SMTP, Simple Mail Transfer Protocol, offre un service de courrier électronique ;

– TFTP, Trivial FTP, est une version allégée du protocole FTP,

– DNS, Domain Name System, est un système de bases de données réparties assurant la correspondance d'un nom symbolique et d'une adresse Internet (adresse IP) ;

– RIP, Routing Information Protocol, est le premier protocole de routage (vecteur distance) utilisé dans Internet ;

– SNMP, Simple Network Management Protocol, est devenu le standard des protocoles d'administration de réseau ;

– ICMP, Internet Control and error Message Protocol, assure un dialogue IP/IP et permet notamment : la signalisation de la congestion, la synchronisation des horloges et l'estimation des temps de transit... Il est utilisé par l'utilitaire Ping qui permet de tester la présence d'une station sur le réseau.

– ARP, Address Resolution Protocol, est utilisé pour associer une adresse logique IP à une adresse physique MAC (Medium Access Control, adresse de l'interface dans les réseaux locaux) ;

– RARP, Reverse Address Resolution Protocol, permet l'attribution d'une adresse IP à une station ;

– OSPF, Open Shortest Path First, est un protocole de routage du type état des liens, il a succédé à RIP ;

– SLIP, Serial Line Interface Protocol, protocole d'encapsulation des paquets IP, il n'assure que la délimitation des trames ;

– PPP, Point to Point Protocol, protocole d'encapsulation des datagrammes IP, il assure la délimitation des trames, identifie le protocole transporté et la détection d'erreurs.

Les mécanismes de base de TCP/IP

Le mode de mise en relation

Désirant alléger au maximum la couche inter réseau, les concepteurs de TCP/IP n'ont spécifié qu'une couche réseau en mode non connecté (mode datagramme). Ce mode de mise en relation optimise l'utilisation des ressources réseaux mais ne permet d'assurer ni un contrôle d'erreur, ni un contrôle de flux. Au niveau du réseau ces tâches sont reportées sur les réseaux physiques réels. En ce qui concerne les systèmes d'extrémité, c'est la couche TCP qui pallie les insuffisances de la couche inter réseau (Internet) en assurant le contrôle d'erreur et de flux de bout en bout (mode connecté). Cette approche est illustrée par la *Figure A1.02*.

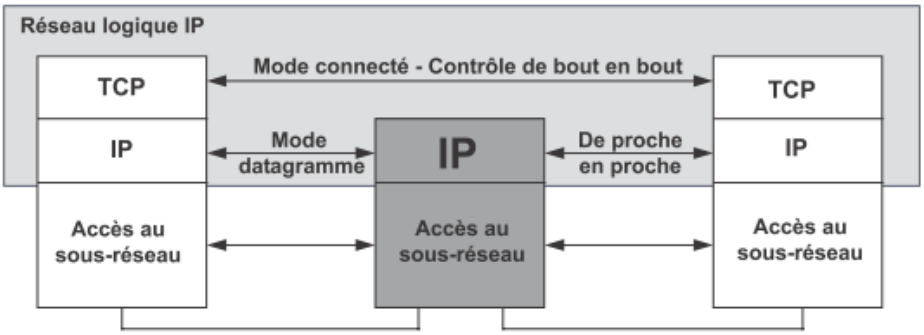


Figure A1.02 *: Réseau logique IP et modes de mise en relation.*

L'encapsulation des données

L'encapsulation consiste à transporter les données d'une couche dans une unité de données de la couche inférieure. Un en-tête contient les informations nécessaires à l'entité homologue pour extraire et traiter les données. Dans le modèle TCP/IP, les données de l'application constituent des messages, ceux-ci sont transportés dans des segments qui seront émis sur le réseau sous forme de datagrammes. L'unité de transport élémentaire est la trame qui constitue au niveau physique un train de bits (*Figure A1.03*).

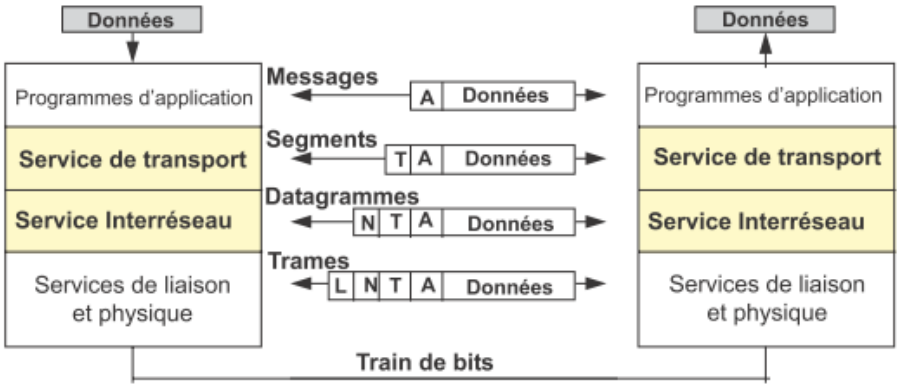


Figure A1.03 *: L'encapsulation des données dans TCP/IP.*

Contrairement à ISO, TCP/IP utilise un format unique d'en-tête de taille fixe. Cette approche, en nécessitant des en-têtes relativement importants, pénalise le débit (overhead protocolaire), mais optimise le traitement des blocs de données dans les systèmes intermédiaires.

La figure 10.5 illustre ce processus. La terminologie utilisée pour désigner les différents blocs de données diffère quelque peu de celle du monde OSI.

Identification des protocoles

À l'instar d'ISO avec la notion de SAP (Service Access Point), un adressage de couche organise le dialogue vertical. Chaque unité protocolaire de TCP/IP identifie le protocole ou l'application supérieure. L'EtherType des trames « Ethernet » identifie le protocole du niveau réseau.

L'identifiant de protocole dans le datagramme IP désigne le protocole de transport utilisé et la notion de port dans le segment TCP détermine l'instance locale de l'application. La *Figure A1.04* illustre ce principe et donne quelques exemples d'identifiants normalisés.

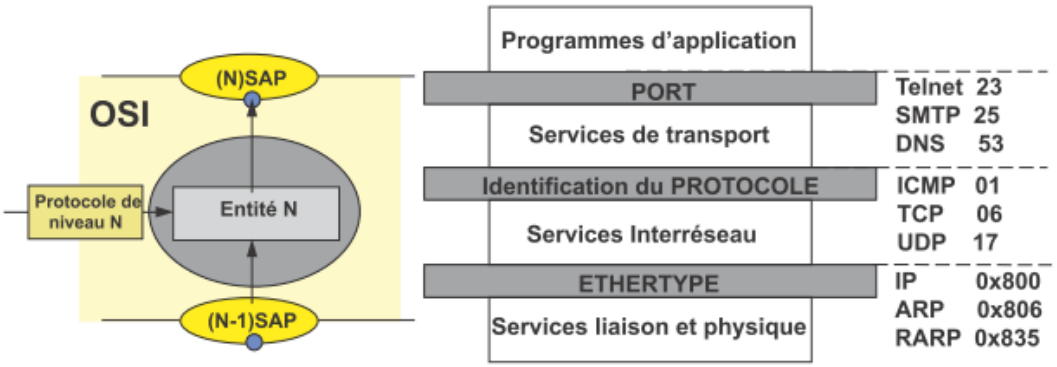


Figure A1.04 *: Identification des protocoles dans TCP/IP.*

Taille du segment de données échangé

Chaque réseau, en fonction de ses caractéristiques spécifiques admet des unités de données de taille plus ou moins grande (MTU, Maximum Transfer Unit). Pour certains réseaux, cette taille est normalisée. C'est le cas, par exemple, pour les réseaux de type Ethernet où la MTU est fixée à 1 500 octets. Dans les réseaux étendus (WAN), la taille est déterminée par l'opérateur en fonction des caractéristiques de ses éléments actifs (buffers...). Un datagramme peut, pour atteindre sa destination, traverser plusieurs réseaux dont les MTU sont différentes. Si le datagramme à transférer a une taille supérieure à la MTU du réseau, le commutateur d'accès devra fractionner (segmenter) l'unité de données pour la rendre compatible avec les capacités de transport du réseau (*Figure A1.05*).

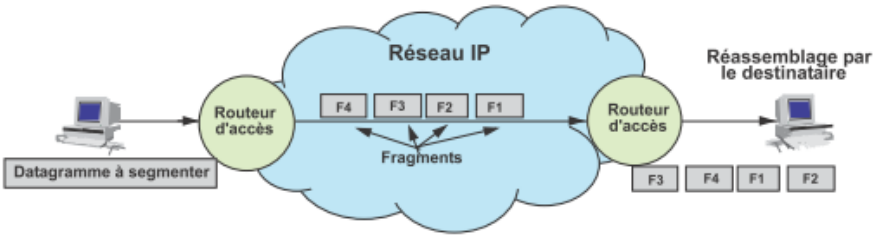


Figure A1.05 *: Principe de la segmentation sous IP.*

Le routage de chaque fragment étant indépendant du précédent et du suivant, aucun nœud n'a la certitude de recevoir tous les fragments, dans ces conditions, seul le destinataire a la capacité de réassembler les différents fragments.

En mode datagramme la perte d'un seul fragment implique une retransmission complète du segment TCP d'origine. Si la connexion est locale, afin d'éviter la reprise d'un segment complet, on cherche à définir une taille de segment correspondant à la taille maximale que peut supporter le réseau. Pour l'interconnexion, via des réseaux de transport, cette taille est fixée à 576 octets, dont 536 utiles. Les passerelles inter réseaux doivent être capables de traiter des segments de 576 octets sans avoir à les fragmenter. Néanmoins, cette taille n'est pas nécessairement compatible avec celle admissible par tous les sous-réseaux physiques traversés. Dans ces conditions, la couche IP fragmentera le bloc de données (datagramme IP), chaque fragment constituant un nouveau datagramme IP.

Lors de l'établissement de la connexion de transport, une option de TCP permet l'annonce, et non la négociation, de la taille maximale de segment que le système d'extrémité peut admettre (MSS, Maximum Segment Size). La *Figure A1.06* illustre la relation entre MSS et MTU pour la valeur par défaut de 576 octets de MTU. La MTU de 576 octets garantit une charge utile minimale de 512 octets aux données de l'application.

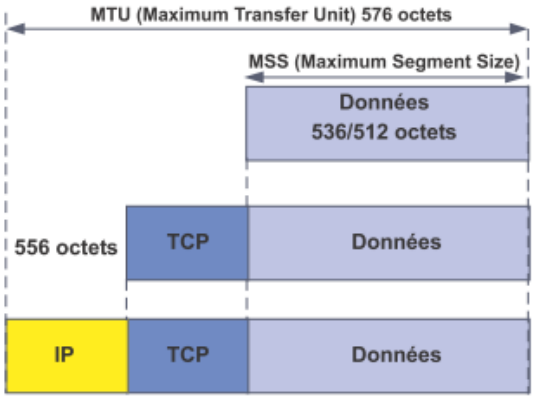


Figure A1.06 *: Relation entre MTU et MSS (Valeur implicite).*

ANNEXE 2: BASES DU CHIFFREMENT SYMETRIQUE

Le chiffrement d'un message consiste à le coder (c'est le terme commun, mais en cryptographie, on dit plutôt "chiffrer" que "coder") pour le rendre incompréhensible pour quiconque n'est pas doté d'une clé de déchiffrement K_D (qui doit donc impérativement être gardée secrète):

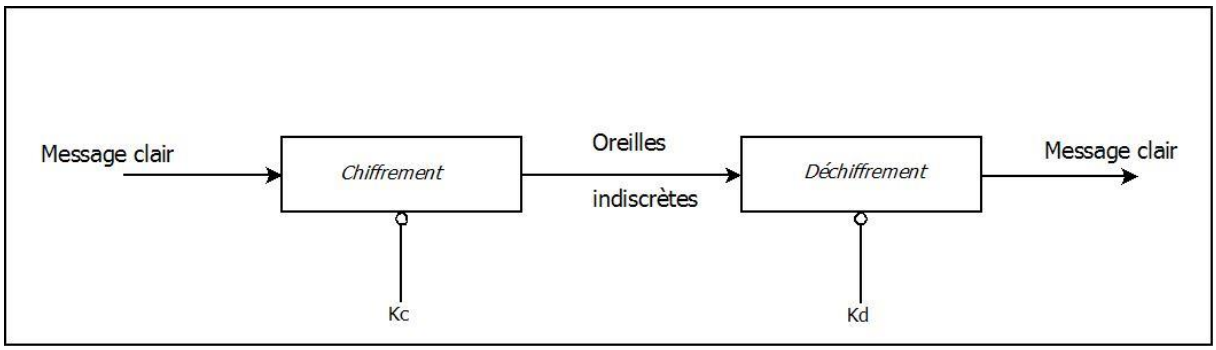


Figure A2.01 *: Principe de cryptage*

Le chiffrement est une activité cryptographique très ancienne qui remonte à l'antiquité (6ème siècle avant J.C.).

Dans un crypto système, on distingue:

1 - l'espace des messages clairs M sur un alphabet A (qui peut être l'alphabet romain, mais qui sera dans la pratique {0, 1} car tout message sera codé en binaire pour pouvoir être traité par ordinateur);

2 - l'espace des messages chiffrés C sur un alphabet B (en général égal à A);

3 - l'espace des clés K

4 - un ensemble E de transformations de chiffrement (chaque transformation étant indexée par une clé):

$$E_K : M \in \mathrm{M} \rightarrow \mathrm{C} \in \mathrm{C}$$

5 - un ensemble D de transformations de déchiffrement (chaque transformation étant indexée par une clé):

$$D_K : \mathrm{C} \in \mathrm{C} \rightarrow M \in \mathrm{M}$$

Evidemment, la condition $D_{K_D}\left(E_{K_C}(M)\right) = M$ doit être réalisée.

On notera un tel crypto système (M, C, K, E, D).

Chiffrement par flot, par blocs

Les schémas de chiffrement par flot, traitent l'information bit à bit, et sont très rapides. Ils peuvent être traités avec une m´mémoire limitée et la propagation des erreurs de transmission du chiffré au clair est limitée. Ils sont parfaitement adaptés à des moyens de calcul, de m´mémoire et de transmission limités (cryptographie en temps réel) comme la cryptographie militaire, ou la cryptographie entre le téléphone portable GSM et son réseau (système A51).

Leur principe est d'effectuer un chiffrement de Vernam en utilisant une clé pseudo-aléatoire, c'est à dire une clé qui ne soit pas choisie aléatoirement parmi tous les mots binaires de longueur n. Cette clé (qu'on appellera suite pseudo-aléatoire) est générée par différents procédés à partir d'une clé aléatoire d'une longueur juste suffisante pour résister aux attaques exhaustives (les performances à venir des ordinateurs demandent donc de la prendre d'au moins 80 bits), qu'on appellera clé courte.

La sécurité d'un tel système contre une attaque à clair connu est équivalente à la sécurité du générateur pseudo-aléatoire. On demande bien entendu que la connaissance de N bits consécutifs de la clé ne permette pas de calculer facilement la clé courte. On est plus exigeant en demandant que la connaissance de N bits consécutifs ne permette pas de prévoir facilement (i.e. en temps polynomial en N) la valeur du bit suivant avec probabilité $p > 1/2$. En particulier cela demande que la suite des bits de la clé satisfasse aux propriétés de régularité d'une suite véritablement aléatoire. Les schémas par blocs sont plus lents et nécessitent plus de moyens informatiques que les schémas par flots. Mais ils sont bien adaptés à la cryptographie civile comme celle des banques.

Dans un système par blocs, chaque clair est découpé en blocs de même longueur et chiffré bloc par bloc. La longueur l des clés doit être suffisante pour que l'attaque exhaustive (attaque à chiffré seul si on a un moyen de différencier les clairs des 14messages aléatoires, et attaque à clair connu sinon) consistant à déchiffrer le chiffré avec toutes les clés possibles jusqu'à l'obtention du clair, soit irréaliste ($l \geq 80$).

La longueur n des blocs doit également être suffisante pour éviter les attaques dites par dictionnaire consistant `a s'aider d'un pré-calcul (partiel) des chiffrés des 2 n blocs possibles.

La sécurité des schémas par blocs retenus comme standards (DES puis AES) repose sur le fait qu'avant d'être retenus (et après!), ils ont été massivement attaqués par la communauté cryptographique. Ces schémas qui ont résisté sont alors considérés comme sûrs. Le DES (Data Encryptions Standard) date des années 70 et a résisté à toutes les techniques de cryptanalyse (l'attaque qui reste la plus efficace en pratique est l'attaque exhaustive). Il a dû cependant être remplacé récemment comme standard par l'AES (Advanced Encryptions Standard), car sa clé de 56 bits était trop courte (et de longueur non modifiable!) pour assurer de nos jours une résistance suffisante à l'attaque exhaustive.

Les exemples historiques de chiffrement (par transposition et par substitution) vus en début de cours sont des chiffrements par blocs. La substitution ajoute de la confusion au procédé de chiffrement et la transposition ajoute de la diffusion en éparpillant l'influence moyenne (selon les différentes clés) de chaque bit du clair, sur les bits du chiffré. Mais aucun de ces deux procédés ne produit `a la fois de la confusion et de la diffusion, et c'est une raison pour laquelle ils ne peuvent pas à assurer une réelle sécurité. Les systèmes modernes, pour assurer une véritable sécurité, doivent produire à la fois de la confusion et de la diffusion, faute de quoi ils ne résistent pas aux attaques que nous d´écrirons plus loin.

L'AES

L'Advanced Encryptions Standard a fait l'objet d'un appel d'offre datant de 1997. Il s'agissait de remplacer le DES dont la taille des clés était devenue trop petite pour les performances des ordinateurs modernes. Les spécifications étaient une longueur de blocs de 128 bits (ou de 256 bits) et une longueur de clé paramétrable: 128 ou 192 ou 256 bits. Parmi les 15 candidats, le candidat retenu (en 2000) se nomme RIJNDAEL (mais on l'appelle simplement l'AES). Il est dû à deux chercheurs Belges, Rijmen et Daemen. C'est un chiffrement itératif, mais contrairement à 9 autres candidats, ce n'est pas un chiffrement de Feistel. C'est un chiffrement par substitution-permutation: à chaque tour, le chiffré produit par le tour précèdent subit une substitution non-linéaire qui assure la confusion puis une permutation linéaire qui assure la diffusion, puis la clé du tour est ajoutée bit à bit pour donner. Le nombre de tours est 10 pour une clé de 128 bits et de 14 pour une clé de 256 bits.

La fonction de diffusion agit sur les 16 octets de l'entrée en les permutants puis en appliquant la même application linéaire sur chaque bloc de 4 octets consécutifs.

Chaque octet est identifié à un élément du corps $F_{2^8} = F_{256}$ à 256 éléments. Le polynôme irréductible utilisé pour la construction de ce corps est

$$P(X) = X^8 + X^4 + X^3 + X + 1$$

Ce n'est pas un polynôme primitif mais les concepteurs l'ont choisi pour accélérer les calculs. L'identification entre les éléments de ce corps et les octets se fait classiquement: on choisit une base du corps vu comme espace vectoriel de dimension 8 sur F_2 ; chaque élément du corps peut ainsi être identifié à un vecteur binaire de longueur 8, c'est à dire à un octet. L'application linéaire est définie par sa matrice 4 × 4 à coefficients dans le corps F_{256} :

$$\begin{bmatrix} \propto & \propto+1 & 1 & 1 \\ 1 & \propto & \propto+1 & 1 \\ 1 & 1 & \propto & \propto+1 \\ \propto+1 & 1 & 1 & \propto \end{bmatrix}$$

où α est un élément primitif. Elle agit donc principalement au niveau global des octets (mais elle agit quand même au niveau des bits via les multiplications par les éléments de ce corps).

La fonction de substitution (la boite S): chaque octet est considéré comme un élément du corps F_{256}. La boite S est constituée de 16 boites identiques consécutives agissant chacune sur un octet. Chaque boite Si consiste en l'application F : $x \in F_{256}$ 7→ $x^{254} \in F_{256}$. Notons que si x = 0 alors x^{254} = 0 et si $x \neq 0$ alors $x^{254} = \frac{1}{x}$

Le résultat de cette transformation subit ensuite une application affine. Cette boite S permet au système de résister à l'attaque différentielle et à l'attaque linéaire. La raison de ce choix est que la fonction F : $x \in F_{256}$ 7→ $x^{254} \in F_{256}$ est telle que l'équation F (x) + F (x + a) = b admet au plus 4 solutions (cela évite l'existence de différentielles de probabilités élevées) et que l'équation $a \cdot X \oplus b \cdot F(X) = 0$ admet un nombre de solutions proche de 2^{128} pour tout $a \neq 0$ et tout $b \neq 0$ (cela rend couteuse l'attaque linéaire).

BIBLIOGRAPHIE

[1] R. Pandya, *"Emerging mobile and personal communication systems,"* IEEE Communications Magazine, Vol. 33,pp. 44--52, juin 1995.

[2] Schulzrinne, H., Casner, S., Frederick, R. et V. Jacobson, "*RTP: A Transport Protocol for Real-Time Applications"*, RFC 1889, janvier 1996.

[3] Schulzrinne, H., Rao, R. et R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, avril 1998.

[4] Cuervo, F., Greene, N., Rayhan, A., Huitema, C., Rosen, B. et J. Segers, *"Megaco Protocol Version 1.0"*, RFC 3015, novembre 2000.

[5] Handley, M. et V. Jacobson, *"SDP: Session Description Protocol*", RFC 2327, avril 1998.

[6] Boucadair Mohamed *"Etude des problèmes de sécurité liés au protocole SIP (Session Initiation Protocol)"*

[7] Mathieu Sengelé *"Le protocole SIP et la gestion des sources dynamiques dans une session de groupe"* (Rapport)

[8] J. Rosenberg ; H. Schulzrinne ; G. Camarillo ; A. Johnston ; J. Peterson ; R. Sparks ; M. Handley ; E. Schooler *"SIP : Protocole d'initialisation de session»*, RFC 3261, juin 2002

[9] Emonet jean-Bruno, *"Algorithmes de chiffrement: Mesures de performances réseaux"*, Dernière mise à jour le 24 juin 2005;

[10] Joan Daemen, Vincent Rijmen, *"The Rijndael Block Cipher"*, Document version 2, Date: 03/09/99

[11] Charles CHEBLI, *" Signature et Chiffrement "*, Diplôme d'Etudes Approfondies Réseaux de télécommunications, 22/04/2003

[12*] "Manuel Windows Trust 3.0"*, 2008-2009

[13] Julien CHAMELOT, *"Asterisk Base"*, Dernière modification 10/11/2010

FICHE DE RENSEIGNEMENT

Nom : RADIASON

Prénoms : Tsiry Andriamampianina

Adresse : Lot IPN 48 bis Ambanilalana Itaosy

Antananarivo 102

Téléphone : +261 033 25 432 41

E-mail : **rtsiry@live.fr**

Titre du mémoire :

Développement D'un Logiciel De Messagerie Instantanée

Avec Un Module De Cryptage De Message

Nombre de pages : 70

Nombre de tableaux : 2

Nombres de figures : 44

Mots clés : Messagerie, cryptage, SIP, AES Rijndael, programmation, sécurité, tchat.

Directeur de mémoire : Monsieur Ndaohialy Manda-Vy RAVONIMANANTSO

RESUME

L'objectif de ce projet, après avoir établi des études sur le protocole SIP, est de sécuriser un réseau VoIP. L'étude consiste à évaluer les risques d'attaques sur le réseau et voir quelles sont les vulnérabilités existantes afin de sécuriser le réseau.

Dans une première étape, nous nous sommes intéressés à l'étude du protocole SIP avec ces structures et ses mécanismes. Dans une seconde étape, nous avons étudié les services de sécurité interne au protocole, puis les problèmes de sécurité du système, les vulnérabilités et les attaques possibles. Dans une troisième étape, nous avons étudié le fonctionnement des systèmes de chiffrement symétriques, leurs exigences et leurs capacités. Dans la dernière étape nous avons installé un programme de messagerie instantanée et un serveur ASTERISK déjà configuré pour tester la sécurité installée.

Dans cet ouvrage nous avons présenté une proposition pour sécuriser les messages des voix sur IP en générale et plus particulièrement le protocole SIP en particulier en utilisant la cryptographie.

ABSTRACT

This memory has enabled us to develop an instant messaging program with ability to encrypt transmissions.

The first part is to present the protocol and message encryption mode, detail the mechanisms of its past and ability.

The second part is dedicated to the elements required for program development and applications useful for testing. Pre configurations are presented for the server and the functionality of each application is also cited.

In the fourth chapter we will talk about the actual construction of the application, we will subsequently test results that we have inflicted in simulation and real job situations; we will reap the real facts of the use application in the workplace or private.