Ministère de L'Enseignement Supérieur et de la Recherche Scientifique.
Université Mouloud Mammeri de Tizi-Ouzou.
Faculté de Génie Electrique et D'informatique.
Département D'informatique.

MEMOIRE

De fin d'études

En vue de l'obtention de diplôme Master académique.

Domaine : *mathématiques et informatique*

Filière : *informatique*

Spécialité : *RMSE*

(LMD)

Thème

Réalisation d'un crypto-système basé sur l'algorithme TEA (Tiny Encryption Algortithme) pour les systèmes embarqués cas : Smartphones.

Réalisé par :

✓MECHHAT DJAMEL
✓ HOUFEL ABDELMALEK

Dirigé par :

✓M^{me} HADAOUI

Promotion : 2016/2017.

Remerciements

Nous rendons grâce à Dieu de nous avoir donné le courage, la volonté ainsi que la conscience nécessaires à l'élaboration de notre projet.

Ce travail n'a pas été possible à réaliser sans le support moral de nos familles.
Que ce travail soit un témoignage de notre gratitude envers elles.

Nous tenons à remercier notre promotrice madame ***Hadaoui*** *qui nous a accompagnées dès le début de ce projet et aussi durant notre formation.*

Nos sincères remerciements s'adressent aussi aux membres de jury qui nous avons fait l'honneur de juger notre travail.

Sans oublier de remercier toutes les personnes qui ont contribuées de près ou de loin à la réalisation de ce projet.

Dédicaces

Je dédié ce modeste travail à toutes personnes ayant participé de près ou de loin à sa réalisation.

A mes parents.

A mes frères et mes sœurs.

A mes amis (Djamel, Si lhewas, Farid, Younes, Mokrane, Fawzi, Ferhat,Lounes, Fares, Azzedine, Arezki, Boujemaa, Aissa, Aziz, Hakim, Lekhdar, Massi, Mokrane, Mohand).

A toutes la promotion RMSE2016/2017.

A tous ceux qui me sont chère

A tous ceux qui m'aiment

A tous ceux que j'aime

Je dédié ce travail.

Abdelmalek

Je dédié ce modeste travail à toutes personnes ayant participé de près ou de loin à sa réalisation.

A mes parents.

A mes frères et mes sœurs.

Je dédié ce travail particulièrement à une personne très chère que j'aime énormément.

A mes amis Malek, Farid, Hocine, Ferhat, Aziz, Azzedine, Kiki, Samir, Akli, Kadir, Boujemaa, Aissa, Arezki, Makhlouf, Hakim, lakhder, Mokrane, Fawzi…

A toute la section master 2 RMSE

Promotion (2016/2017)

Djamel

Sommaire :

Introduction générale 10
I.1 Introduction 11
I.2 Sécurité des réseaux: 11
I.2.1 Définition 11
II.2.2 Objectifs de la sécurité d'un réseau 11
I.3 Motivations d'un attaquant 12
I.4 Les Attaques 12
I.4.1 Définition d'une attaque 12
I.4.2 Types d'attaques 13
I.4.2.1 Attaques virales 13
I.4.2.2 Les attaques réseaux 13
I.4.2.3 Attaques d'accès 16
I.4.2.4 Attaques de Dénis de Service 16
I.5 Les vulnérabilités 17
I.6 Menaces 18
I.7 Risques 18
I.8 Mécanismes de la sécurité 19
I.8.1 Cryptographie, signature électronique et certificats 19
I.8.1.1 Cryptographie 19
I.8.1.2 La signature électronique 19
I.8.1.3 Le certificat: 19
I.8.2 VPN (Virtual Private Network): 20
I.8.2.1 Définition 20
I.8.2.2 Fonctionnement d'un VPN 20
I.8.2.3 Avantages et Inconvénients d'un VPN 20
I.8.3 Antivirus 21
I.8.4 Pare-feu (firewall): 21
I.8.5 Système de détection d'intrusion 23
I.8.6. Proxy 23
I.8.6.1 Principe de fonctionnement 23

I.8.6.2 Les proxys "anonymes" 24
I.9 Conclusion 25

Chapitre 2: La Cryptographie

I.1 Introduction 26
I.2 cryptologie 26
I.3 cryptographie 26
I.3.1 Notations 26
I.3.2 Les avantages de la cryptographie 27
I.3.3 Les grandes menaces 27
I.3.3.1 Les attaques passives 27
I.3.3.2 Les attaques actives 27
I.4 La cryptanalyse 28
I.4.1 Attaque par force brute 28
I.4.2 Attaque à texte chiffré seul 28
I.4.3 Attaque à texte clair connu 28
I.4.4 Attaque à texte clair choisi 28
I.4.5 Attaque par mot probable 29
I.5 Cryptographie moderne 29
I.5.1 Cryptographie symétrique 29
I.5.1.1 Caractéristiques 30
I.5.1.2 DES, le chiffrement à clé secrète 31
I.5.1.3 IDEA 32
I.5.1.4 AES (Advanced Encryption Standard 33
I.5.2 Cryptographie asymétrique (ou chiffrement à clés publiques) 35
I.5.2.1 le principe du chiffrement à clé publique 35
I.5.2.2 Protocole d'échange de clés de Diffie et Hellman 36
I.5.2.3 RSA (Rivest, Shamir et Adleman) 37
I.5.3 Signature électronique 38
I.5.3.1 Introduction à la notion de signature électronique 38
I.5.3.2 L'objectif des signatures électroniques 38
Tiny Encryption Algorithm (TEA) 39

II.1 Introduction 39
II.2 Présentation de l'algorithme TEA 39
II.3 Les caractéristiques de l'algorithme TEA 40
II.4 Principe de fonctionnement du TEA 40
II.4.1 Chiffrement 40
II.4.2 Déchiffrement 44
II.5 Sécurité du TEA 46
II.6 Comparaison avec d'autres algorithmes 47
II.7 Conclusion 48

Chapitre 3: Système Embarqués et Android

III.1.1 Introduction 50
III.1.2 Définition : 50
III.1.3 Brève histoire des systèmes embarqués 51
III.1.4 Architecture d'un système embarqué 51
III.1.5 Caractéristiques principales d'un système embarqué 52
III.1.6 Les avantages d'un système embarqué 52
III.1.7 Contraintes d'un système embarqué 53
III.1.8 Les contraintes de développement sur un système embarqué 54
III.1.9 Contraintes matérielles d'un système embarqué 54
III.1.10 Utilisation d'Android pour des systèmes embarqués et temps réel 55
III.1.11 Compétences pour la conception et développement de Système embarqué . 56
III.1.12 Conclusion 57
III.2.1 Historique d'Android 59
III.2.2 Listes des versions android 59
III.2.3 Architecture d'android 63
III.2.3.1 Le noyau Linux 64
III.2.3.2 Librairies 65
A) Bioniclibc 65
B) WebKit 66
C) Media Framework 66
III.2.3.3 Hardware Abstraction Layer 66

III.2.3.4 Android Runtime 67
III.2.3.4.1 Dalvik 67
III.2.3.4.2 librariesCore 68
III.2.3.5 La couche Framework 69
III.2.3.5.1 CorePlateform Services 69
III.2.3.5.2 Hardware Services (Les services matériels) 70
III.2.3.6 La couche Application 70
III.2.4 Composantes d'une application Android 71
III.2.4.1 Activities (Activités en Français) 71
III.2.4.2 Services 71
III.2.4.3 Broadcast and IntentReceivers 72
III.2.4.4 Content providers 72
III.2.4.5 Le cycle de vie d'une activité 72
III.2.5 Les outils de développement Android indispensables 75
III.2.5.1 Eclipse 75
III.2.5.2 Le JDK Java Development Kit (JDK) 75
III.2.5.3 LE SDK Software Development Kit 75
Conclusion 77

Chapitre 4: La Réalisation

IV.1. Introduction 78
IV.2 Environnement de travail 78
IV.2.1 Environnement logiciel 78
IV.2.2 Environnement de travail 78
IV.2.2.1 Langage de programmation 78
IV.2.2.2 Eclipse 79
IV.3 Les différentes interfaces de notre application 80
IV.3.1 Lancement de l'application 80
IV.3.2 Le menu principal de l'application 81
IV.3.3 L'interface de cryptage 82
IV.3.4 L'interface Résultat 84
IV.3.5 L'interface Décryptage 86

IV.4 Conclusion 87
Conclusion générale 88
Bibliographie 89

Introduction générale :

Le besoin de dissimuler les informations préoccupe l'homme depuis le début de la civilisation, La confidentialité apparait notamment nécessaire lors des luttes pour l'accès au pouvoir. Puis elle se développe énormément à des fins militaires et diplomatiques. Aujourd'hui, de plus en plus d'applications dites civiles nécessitent la sécurité des données transitant entre deux interlocuteurs ou plusieurs, via un vecteur d'information comme les réseaux de télécommunications actuels et futurs. Ainsi, les banques utilisent ces réseaux pour assurer la confidentialité des opérations avec leurs clients ; les laboratoires de recherche s'en servent pour échanger des informations dans le cadre d'un projet d'étude commun ; les chefs militaires pour donner leurs ordres de bataille, etc. De nos jours, la nécessité de cacher ou de casser une information rentre dans un vaste ensemble appelé cryptographie.

Le mot cryptographie est un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Le verbe crypter est parfois utilisé mais on lui préfèrera le verbe chiffré. La cryptographie est l'art du secret à celle de la piraterie, sert non seulement à préserver la confidentialité des données mais aussi à garantir leur intégrité et leur authenticité. L'objectif principal de notre projet de fin d'étude, est de développer une application qui va chiffrer des messages et qui assuré la confidentialité.

I.1 Introduction :

L'informatique et en particulier l'internet jouent un rôle grandissant dans le domaine des réseaux. Un grand nombre d'applications critiques d'un point de vue de leur sécurité sont déployées dans divers domaines comme le domaine militaire, la santé, le commerce électronique, etc. La sécurité des réseaux devient alors une problématique essentielle tant pour les individus que pour les entreprises ou les états. Il est donc important de définir une politique de sécurité pour protéger ces réseaux contre des attaques qui peuvent être réalisé par des hackers en exploitant leurs failles et leurs vulnérabilités.

Néanmoins les mécanismes de sécurité préventifs mis en place ne sont pas incontournables. Il est nécessaire de mettre en œuvre des outils permettant de détecter toute violation de la politique de sécurité, c'est-à-dire toute intrusion.

Tout au long de ce chapitre, nous parlerons sur les principales menaces pesant sur la sécurité des réseaux ainsi que les mécanismes de défense.

I.2 Sécurité des réseaux:

I.2.1 Définition:[1]

La sécurité d'un réseau est un ensemble de moyens techniques, organisationnels, juridiques et humains nécessaires et mis en place pour conserver, rétablir, et garantir sa sécurité. En général, la sécurité d'un réseau englobe celle du système informatique sur lequel il s'appuie.

II.2.2 Objectifs de la sécurité d'un réseau

La sécurité informatique vise généralement cinq principaux objectifs (critères) :

• ***Disponibilité*** : Elle consiste à garantir l'accès à un service ou à une ressource.

• ***Intégrité*** : Elle consiste à s'assurer que les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle).

• ***Confidentialité*** : Elle consiste à rendre l'information inintelligible à d'autres personnes que les seuls acteurs concernés.

• ***Authentification*** : Elle consiste à assurer l'identité d'un utilisateur, c'est-à dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit l'être.

• ***Non répudiation*** : Elle consiste à garantie qu'aucun des correspondants ne pourra nier la transaction.

I.3 Motivations d'un attaquant :

Un attaquant n'est pas forcément un "hacker" chevronné. Ça peut être n'importe quelle personne avec des motivations aussi banales que les suivantes :

- Le gain financier
    - Récupération de num de cartes bancaires, ...
- Vengeance
    - Site www.aljazeera.net lors de la couverture de la guerre d'irak
- Besoin de reconnaissance
    - Attaque contre le site du cerist avec un message sur les restrictions d'accès à Internet à Cuba.
- Curiosité
    - Attaques d'étudiants du MIT sur le premier ordinateur IBM 704 au MIT en 1959.
- Recherche d'émotions fortes
- Ignorance [8]

I.4 Les Attaques :

I.4.1 Définition d'une attaque :

Une « **attaque** » est l'exploitation d'une faille d'un système informatique (système d'exploitation, logiciel ou bien même de l'utilisateur) à des fins non connues par l'exploitant du systèmes et généralement préjudiciables.
Sur internet des attaques ont lieu en permanence, à raison de plusieurs attaques par minute sur chaque machine connectée. Ces attaques sont pour la plupart lancées automatiquement à partir de machines infectées (par des virus, chevaux de troie, vers, etc.), à l'insu de leur propriétaire. Plus rarement il s'agit de l'action de pirates informatiques. .
Afin de contrer ces attaques il est indispensable de connaître les principaux types d'attaques afin de mettre en œuvre des dispositions préventives. [2]

I.4.2 Types d'attaques :

I.4.2.1 Attaques virales :

Il existe principalement trois types de menaces distinctes :

I.4.2.1.1 Virus :

Un virus est un logiciel autoreproductible comportant du code informatique se copiant explicitement pouvant infecter d'autres programmes en les modifiant ou en modifiant leur environnement. Afin que l'accès à un logiciel infecté implique l'accès à une copie évoluée du virus. Les deux caractéristiques fondamentales des virus sont ainsi leur auto-reproductibilité et leur auto-propagation. [3]

I.4.2.1.2 Vers :

Un ver (ou *worm*) est un type de virus particulier qui se propage par le réseau. Le vers contrairement aux virus, une fois implantés et activés dans un ordinateur, sont des programmes capables de se propager d'un ordinateur à un autre via le réseau, sans intervention de l'utilisateur et sans exploiter le partage de fichiers.

I.4.2.1.3 Cheval de Troie :

Un cheval de Troie (*Trojan horse*) est un programme qui exécute des instructions sans l'autorisation de l'utilisateur. Ces instructions sont généralement nuisibles à l'utilisateur, et qui une fois installé sur un ordinateur y effectue des actions cachées et pernicieuses. [5]

Le cheval de Troie contrairement au ver ne se réplique pas.

I.4.2.2 Les attaques réseaux :

Les attaques réseaux les plus connues aujourd'hui sont :

- **Spoofing IP** [13]

Le spoofing IP (en français mystification) est une technique permettant à un pirate d'envoyer à une machine des paquets semblant provenir d'une adresse IP autre que celle de la machine du pirate. Le spoofing IP n'est pas pour autant un changement d'adresse IP. Plus exactement il s'agit d'une mascarade de l'adresse IP au niveau des paquets émis, c'est-à-dire une modification des paquets envoyés afin de faire croire au destinataire qu'ils proviennent d'une autre machine.

- **Spoofing ARP**

Le spoofing ARP est une technique qui modifie le cache ARP. Le cache ARP contient une association entre les adresses matérielles des machines et les adresses IP, l'objectif du pirate est de conserver son adresse matérielle, mais d'utiliser l'adresse IP d'un hôte approuvé. Ces informations sont simultanément envoyées vers la cible et vers le cache. A partir de cet instant, les paquets de la cible seront routés vers l'adresse matérielle du pirate.

- **Spoofing DNS**

Le système DNS (Domain Name System) a pour rôle de convertir un nom de domaine en son adresse IP et réciproquement, à savoir : convertir une adresse IP en un nom de domaine. Cette attaque consiste à faire parvenir de fausses réponses aux requêtes DNS émises par une victime. Il existe deux types de méthode: [14]

  - **DNS ID spoofing** L'attaquant essaie de répondre à un client en attente d'une réponse d'un serveur DNS, avec une fausse réponse et avant que le serveur DNS ne réponde.
  - **DNS Cache Poisoning** L'attaquant essaie d'empoisonner le cache (table de correspondance IP- nom _machine) du serveur DNS.

- **Désynchronisation TCP** [2]

Le protocole TCP permet d'assurer le transfert des données de façon fiable, bien qu'il utilise le protocole IP (qui n'intègre aucun contrôle de livraison de datagramme) grâce à un système d'accusés de réception (ACK) permettant au client et au serveur de s'assurer de la bonne réception mutuelle des données.

Lors de l'émission d'un segment, un numéro d'ordre (appelé aussi numéro de séquence) est associé, et un échange de segments contenant des champs particuliers (appelés *drapeaux*, en anglais *flags*) permet de synchroniser le client et le serveur. Ce dialogue (appelé *poignée de mains en trois temps*) permet d'initier la communication, il se déroule en trois temps, comme sa dénomination l'indique:

- Dans un premier temps, la machine émettrice (le client) transmet un segment dont le drapeau SYN est à 1 (pour signaler qu'il s'agit d'un segment de synchronisation), avec un numéro d'ordre C, que l'on appelle numéro d'ordre initial du client
- Dans un second temps la machine réceptrice (le serveur) reçoit le segment initial provenant du client, puis lui envoie un accusé de réception, c'est-à-dire un

segment dont le drapeau ACK est à 1 (accusé de réception) et le drapeau SYN est à 1 (car il s'agit là encore d'une synchronisation). Ce segment contient un numéro de séquence S. Le champ le plus important de ce segment est le champ accusé de réception (ACK) qui contient le numéro d'ordre initial du client, incrémenté de 1

- Enfin, le client transmet au serveur un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1, et dont le drapeau SYN est à zéro (il ne s'agit plus d'un segment de synchronisation). Son numéro d'ordre est incrémenté et le numéro d'accusé de réception représente le numéro de séquence initial du serveur incrémenté de 1

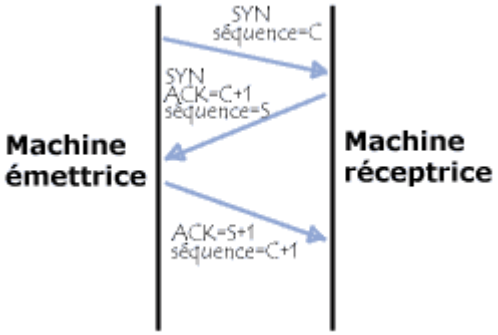


Figure I.1 : Désynchronisation TCP [2]

L'attaquant peut rediriger le trafic TCP, pour cela il envoie des paquets malformés au client avec une adresse IP correspondant à celle du serveur en y plaçant des mauvais numéros de séquences, le client va croire qu'il a perdu la connexion et stoppera ses échanges avec le serveur. Mais si l'attaquant envoie les bons numéros de séquences au serveur, il récupèrera la connexion pour lui.

I.4.2.3 Attaques d'accès :

Les attaques d'accès exploitent des vulnérabilités connues dans les services d'authentification, les services FTP et les services Web pour accéder à des comptes Web, à des bases de données confidentielles ou à toute autre information sensible.

On trouve Plusieurs types (Exemples):

I.4.2.3.1 Attaques de mot de passe :

Cette attaque vise à trouver le mot de passe d'un utilisateur sur un ordinateur ou sur un équipement réseau. Si l'intrus arrive à retrouver le mot de passe, il peut accéder à une ressource, ou ouvrir une porte dérobée pour des futurs accès. La technique la plus utilisée est la force brute (brut force) qui consiste à essayer toutes les combinaisons jusqu'à trouver le mot de passe, L'attaque par dictionnaire est une variante de la force brute qui consiste à essayer les mots prélevés dans un dictionnaire qui contient les mots susceptibles de constituer un mot de passe.

I.4.2.3.2 Trust exploitation :

Dans cette attaque, l'intrus utilise les privilèges d'une autre entité de confiance pour pénétrer dans un système sécurisé.

I.4.2.3.3 Port redirection :

La redirection de port est un type de Trust exploitation qui fait passer un flux sur un port non autorisé par le pare-feu en le faisant passer pour un flux d'un autre port autorisé.

I.4.2.3.4 Man-In-The-Middle:

L'intrus est positionné au milieu d'une communication entre deux parties dans le but de l'espionner ou de la modifier, Un intrus peut fournir des services de point d'accès wifi ou de passerelle réseau pour centraliser les communications à son niveau. La victime peut se connecter et utiliser normalement le réseau sans savoir que le flux passe par une tiers personne.

I.4.2.3.5 Buffer overflow :

Cette attaque utilise les failles des programmes et les propriétés des langages tels que C et C++. L'intrus injecte plus de données dans le buffer de l'application que ce qui a été prévu dans le but de le saturer et d'arriver à l'adresse de retour. Ensuite, l'intrus écrase l'adresse de retour dans le but de se brancher vers un programme malicieux.

I.4.2.4 Attaques de Dénis de Service

Une « attaque par déni de service » est une attaque ayant pour but de rendre indisponible un service, d'empêcher les utilisateurs légitimes d'un service de l'utiliser. Il peut s'agir de :

- L'inondation d'un réseau afin d'empêcher son fonctionnement ;
- La perturbation des connexions entre deux machines, empêchant l'accès à un service particulier.
- L'obstruction d'accès à un service à une personne en particulier ;
- Également le fait d'envoyer des millions de kilooctets à une box Wi-Fi.

L'attaque par déni de service peut ainsi bloquer un serveur de fichiers, rendre impossible l'accès à un serveur web ou empêcher la distribution de courriel dans une entreprise. L'attaquant hacker n'a pas forcément besoin de matériel sophistiqué. Ainsi, certaines attaques DoS (attaque par déni de service) peuvent être exécutées avec des ressources limitées contre un réseau beaucoup plus grand et moderne. On appelle parfois ce type d'attaque « attaque asymétrique » (en raison de la différence de ressources entre les protagonistes). Un hacker avec un ordinateur obsolète et un modem lent peut ainsi neutraliser des machines ou des réseaux beaucoup plus importants. [7]

I.5 Les vulnérabilités

Les vulnérabilités sont les faiblesses d'un système ou d'un logiciel. Ces faiblesses sont exploitées par des hackers pour obtenir un accès à une ressource (**CPU**, réseaux, etc.) ou à une information. Les faiblesses se trouvent au niveau des services, des applications et des utilisateurs:

- **Les services:**

En général, ces services sont l'e-mail, le Web ou toute autre application qui communique avec une autre application. On citera par exemple les attaques contre IIS en 2001 qui ont permis la diffusion du ver CodeRed. Les vulnérabilités des services sont très dangereuses car elles ne nécessitent pas une intervention humaine.

- **les applications:**

Les vulnérabilités des applications nécessitent souvent une intervention humaine: par exemple l'utilisateur qui active un virus par un clic de souris. Quoi de plus alléchant que des mails avec le sujet "I love you" ou avec un contenu informant que vous êtes l'heureux gagnant d'une énorme somme d'argent? Un simple clic et on se retrouve victime d'un virus, worm ou spyware.

- **les actions des utilisateurs:**

Les utilisateurs peuvent engendrer des vulnérabilités sur des systèmes ou des logiciels par des configurations mauvaises ou incorrectes. Un utilisateur peut reconfigurer un système ou un logiciel et ouvrir ainsi des portes à des hackers. Dans ce cas, même le système de sécurisation le plus performant, s'il est mal configuré, offre une brèche de sécurité. [9]

I.6 Menaces [2]

(En anglais threat) signe, indice qui laisse prévoir un danger. Action ou évènement susceptible de se produire, de se transformer en agression contre un environnement ou des ressources et de porter préjudices à leur sécurité. On peut également classer les menaces en deux catégories selon qu'elles ne changent rien (menaces *passives*) ou qu'elles perturbent effectivement le réseau (menaces *actives*).

a) *Les menaces passives* :

Consistent essentiellement à copier ou à écouter l'information sur le réseau, elles nuisent à la confidentialité des données. Dans ce cas, celui qui prélève une copie n'altère pas l'information elle-même.

b) *Les menaces actives* :

Sont de nature à modifier l'état du réseau.

I.7 Risques :

Les risques se mesurent en fonction de deux critères principaux : la *vulnérabilité* et la *sensibilité*. La vulnérabilité désigne le degré d'exposition à des dangers. Un des

points de vulnérabilité d'un réseau est un point facile à approcher. Un élément de ce réseau peut être très vulnérable tout en présentant un niveau de sensibilité très faible : le poste de travail de l'administrateur du réseau, par exemple, dans la mesure où celui-ci peut se connecter au système d'administration en tout point du réseau. [2]

La sensibilité désigne le caractère stratégique d'un composant du réseau. Celui-ci peut être très sensible, vu son caractère stratégique mais quasi invulnérable, grâce à toutes les mesures de protection qui ont été prises pour le prémunir contre la plupart des risques.

I.8 Mécanismes de la sécurité :

Nous avons constatés que les attaquants disposent de plusieurs moyens pour réussir leurs d'attaque, Il faut mettre en place des mécanismes pour s'assurer de la confidentialité, l'intégrité et la disponibilité des services. Parmi ces mécanismes, on peut citer:

I.8.1 Cryptographie, signature électronique et certificats :

I.8.1.1 Cryptographie [20]

Le mot cryptographie est un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages. Chiffrer un message consiste à le transformer au moyen d'un algorithme mathématique afin de le rendre inintelligible, sauf pour celui qui possède le moyen (une clé) de le déchiffrer. L'encryptions des informations électronique qui transitent par le réseau est utilisée pour assurer la confidentialité et l'authenticité des transactions. Le chiffrement se fait généralement à l'aide d'une clé de chiffrement, le déchiffrement nécessite quant à lui une clé de déchiffrement.

I.8.1.2 La signature électronique:

C'est un code digital (une réduction du document électronique à envoyer) qui, associé aux techniques de cryptage, garantit l'identité de la personne qui émet le message et assure la non-répudiation et l'intégrité de l'envoi.

I.8.1.3 Le certificat:

Document électronique (carte d'identité) émis par une autorité de certification. Il valide l'identité des interlocuteurs d'une transaction électronique, associe une identité

à une clé publique de cryptage et fournit des informations de gestion complémentaires sur le certificat et le détenteur.

I.8.2 VPN (Virtual Private Network) [10]:

I.8.2.1 Définition :

VPN pour Virtual Private Network ou réseau privé virtuel.

De plus en plus souvent les entreprises ont besoins d'échanger des données avec des succursales ou des employés travaillant à l'extérieur de l'entreprise. Le VPN est un moyen sécurisé pour échanger, à travers Internet, des données informatiques (fichiers, images, etc…).

Le VPN permet également de sécurisé des connexions de travail à distance.

I.8.2.2 Fonctionnement d'un VPN :

On peut représenter le VPN comme un tunnel sécurisé, dans lequel les données transitent dans des algorithmes cryptographiques. Il y a peu de contrainte à l'établissement d'un VPN. Une des rares contraintes est de disposer d'une IP fixe (Internet protocol, suite de chiffres servant à identifier une adresse). Une autre contrainte est qu'il faut paramétrer plusieurs périphériques qui participent à la sécurité de l'entreprise comme, par exemple, le pare-feu.

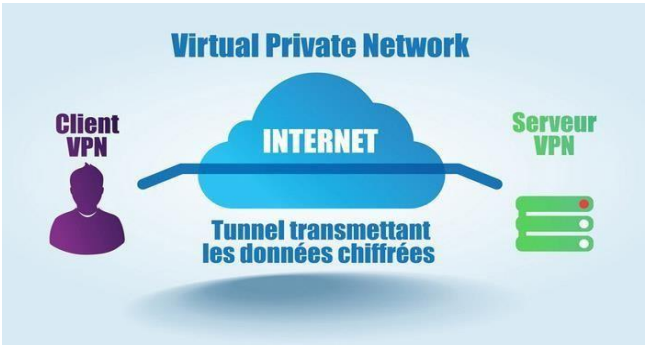


Figure I.2 : VPN [10]

I.8.2.3 Avantages et Inconvénients d'un VPN :

- ✓ **Avantages d'un VPN :**

On peut constater deux avantages principaux à cette méthode de communication :

- La sécurité : Le VPN offre un très bon compromis en matière de sécurité.
- Le coût : La mise en place d'un VPN est peu onéreuse, puisque celui-ci ne demande pas d'équipement spécifique.

✓ **Inconvénients d'un VPN :**

Le principal défaut d'un VPN est qu'un protocole, même sécurisé, qui emprunte un réseau publique pour se déplacer n'est pas la panacée en matière de sécurité.

I.8.3 Antivirus:

Les antivirus sont des logiciels conçus pour identifier, neutraliser et éliminer des logiciels malveillants. Ceux-ci peuvent se baser sur l'exploitation de failles de sécurité, mais il peut également s'agir de programmes modifiant ou supprimant des fichiers, que ce soit des documents de l'utilisateur de l'ordinateur infecté, ou des fichiers nécessaires au bon fonctionnement de l'ordinateur. [11]

Un antivirus vérifie les fichiers et courriers électroniques, les secteurs de boot (pour détecter les virus de boot), mais aussi la mémoire vive de l'ordinateur, les médias amovibles (clefs USB, CD, DVD, etc.), les données qui transitent sur les éventuels réseaux (dont internet), etc.

I.8.4 Pare-feu (firewall) [12]:

✓ **Définition :**

Un pare-feu, aussi appelé communément *coupe-feu*, *garde-barrière ou firewall*, est un système permettant de protéger un ordinateur, et/ou un réseau d'ordinateurs, des intrusions provenant d'un réseau tiers externe (habituellement internet). Le pare-feu est un système permettant de filtrer les paquets de données échangés avec ce réseau. Le pare-feu sert donc de "passerelle filtrante" comportant au minimum les interfaces réseau suivante :

- une interface pour le réseau à protéger (réseau interne) ;
- une interface pour le réseau externe.

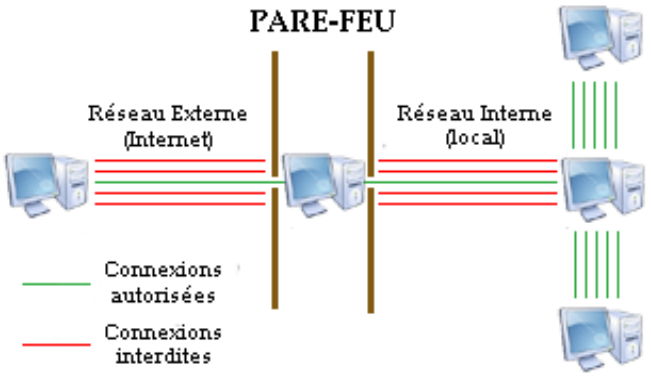


Figure I.3 : pare-feu [12]

Le système pare-feu est un système logiciel, reposant parfois sur un matériel réseau dédié, constituant un intermédiaire entre le réseau local et un ou plusieurs réseaux externes. Il est possible de mettre un système pare-feu sur n'importe quelle machine et avec n'importe quel système d'exploitation à condition que:

- la machine soit suffisamment puissante pour traiter le Traffic informatique.
- Le système soit sécurisé.
- aucun autre service que le service de filtrage de paquets ne fonctionne sur le serveur.

✓ **Fonctionnement d'un système pare-feu**

Un système pare-feu contient un ensemble de règles prédéfinies permettant :

- d'autoriser la connexion (*allow*).
- de bloquer la connexion (*deny*).
- de rejeter la demande de connexion sans avertir l'émetteur (*drop*).

Ces règles permettent de mettre en œuvre une méthode de filtrage se basant sur la politique de sécurité adoptée par l'équipe informatique. Il y a habituellement deux types de politiques de sécurité permettant :

- soit d'autoriser uniquement les communications ayant été explicitement autorisées, donc "*Tout ce qui n'est pas explicitement autorisé est interdit*".
- soit d'empêcher les échanges qui ont été explicitement interdits.

La première méthode est sans doute la plus sûre, mais elle impose toutefois une définition précise et contraignante et des besoins en communication.

I.8.5 Système de détection d'intrusion :

Comme Nous l'avons vu, les attaques utilisées par les pirates sont très variées. Certaines utilisent des failles réseau et d'autres des failles de programmation. Nous pouvons donc facilement comprendre que la détection d'intrusion doit se faire à plusieurs niveaux. Ainsi, il existe différents types de système de détection d'intrusion, Citons quelques types :

- ✓ **Les systèmes de détection d'intrusions (IDS) :**

IDS est un ensemble de composants logiciels et matériels dont la fonction principale est de détecter et analyser toute tentative d'effraction (volontaires ou non) ,et la détection des techniques de sondage (balayage de ports), des tentatives de compromission de systèmes, d'activités suspectes internes, des activités virales ou encore audit des fichiers de journaux(logs).

Certains termes sont souvent employés quand on parle d'IDS :

Faux positif : une alerte provenant d'un IDS, mais qui ne correspond pas à une attaque réelle ;

Faux négatif : une intrusion réelle qui n'a pas été détecté par l'IDS.

- ✓ **Les systèmes de détection d'intrusions « réseau » (NIDS) :**

NIDS est utilisé pour analyser de manière passive le flux en transit sur le réseau et détecter les intrusions en temps réel. Un NIDS écoute donc tout le trafic réseau, puis l'analyse et généré des alertes si des paquets semblent dangereux, les NIDS étant les IDS les plus intéressants et les plus utiles du fait de l'omniprésence des réseaux dans notre vie quotidienne.

I.8.6. Proxy :

Un serveur proxy est un ordinateur ou un module qui sert d'intermédiaire entre un navigateur Web et Internet. Le proxy participe à la sécurité du réseau.

I.8.6.1 Principe de fonctionnement

Les serveurs proxy permettent de sécuriser et d'améliorer l'accès à certaines pages Web en les stockant en cache (ou copie). Ainsi, lorsqu'un navigateur envoie une requête sur la demande d'une page Web qui a été précédemment stockée, la réponse et le temps d'affichage en sont améliorés. L'utilisateur accède plus rapidement au site et ne sature pas le proxy pour sortir. Les serveurs proxy renforcent également la sécurité en filtrant certains contenus Web et les logiciels malveillants.

- ✓ **Filtrage**

Le filtrage est appliqué en fonction de la politique de sécurité mise en place sur le réseau. Ceci permet de bloquer selon une liste noire, les sites considérés comme malveillants et/ou inutiles au contexte de travail de l'entreprise .

- ✓ **Authentification**

Afin de limiter l'accès au réseau extérieur, et de renforcer ainsi la sécurité du réseau local, il peut être nécessaire de mettre en place un système d'authentification pour accéder aux ressources extérieures. Ceci est assez dissuasif pour les utilisateurs souhaitant visiter des sites contraires à la charte de leur système d'information. Ils se sentent suivis et restent "sages" dans leurs recherches.

- ✓ **Stockage des Logs**

Le stockage, des logs des sites visités et des pages vues, permet à l'administrateur du réseau de redéfinir la politique de sécurité du réseau et/ou d'intervenir auprès d'un utilisateur qui visite fréquemment des sites malveillants ou sans rapport avec l'activité de l'entreprise.

I.8.6.2 Les proxys "anonymes" :

En plus de masquer l'adresse IP, un proxy anonyme peut supprimer :

- Cookies
- Pop-ups
- Bannières
- Scripts
- Informations confidentielles en zone de saisie (Identifiant et mot de passe)

I.9 Conclusion :

Dans ce chapitre, Nous avons présenté les principales notions et concepts de la sécurité des systèmes informatiques et des réseaux, dont nous avons décrit plus particulièrement les types d'attaques et de menaces, ainsi différentes méthodes et mécanismes connus pour sécuriser les réseaux. A travers les différentes sections que nous avons montrées, on conclut qu'aucun réseau n'est pas sûr à 100% et il est impossible de garantir la sécurité totale d'un réseau. Mais nous avons pu voir un peu comment sécuriser et protéger des systèmes informatiques et des réseaux, et dans le deuxième chapitre nous allons détailler l'un des moyens de sécuriser une information qui est la cryptographie, et nous allons nous intéresser plus précisément sur l'algorithme de crypthographie TEA (Tiny Encryption Algorihme) qui est l'objet de notre travail.

I.1 Introduction :

Les transactions faites à travers le réseau peuvent être interceptées, d'autant plus que les lois ont du mal à se mettre en place sur Internet, il faut donc garantir la sécurité de ces informations, c'est la cryptographie qui s'en charge. Le but de la cryptographie moderne est de traiter plus généralement des problèmes de sécurité des communications et de fournir un certain nombre de services de sécurité. L'objectif de ce chapitre, est de présenter les méthodes de la cryptographie. Il intéressera le lecteur qui connait peu ou pas au domaine et qui souhaiterait comprendre le fonctionnement et le mécanisme mis en œuvre en cryptographie.

I.2 cryptologie :

La cryptologie est la science du secret, ne peut être vraiment considéré comme science que depuis peu de temps. Cette science englobe la cryptographie, la cryptanalyse et la stéganographie

I.3 cryptographie : [15]

La cryptographie est l'étude des techniques mathématiques qui permettent d'assurer certain service de sécurité. Elle est défini comme étant une science permettant de convertir des informations (en clair) en information cryptées (codées), c'est-à-dire non compréhensible, et puis, à partir de ces information, de restituer les informations originales. La cryptographie est l'étude des méthodes donnant la possibilité d'envoyer des données de manière confidentielle sur un support donné.

I.3.1 Notations:

En cryptographie, la propriété de base est que M = D(E(M))

Où

– M représente le texte clair,

– C est le texte chiffré,

– K est la clé (dans le cas d'un algorithme à clé symétrique), Ek et Dk dans le cas d'algorithmes asymétriques,

– E(x) est la fonction de chiffrement, et

– D(x) est la fonction de déchiffrement.

Ainsi, avec un algorithme à clef symétrique,

$$M = D(C) \text{ si } C = E(M)$$

I.3.2 Les avantages de la cryptographie :

La cryptographie est un outil de sécurité de l'information essentielle. Il fournit les quatre services les plus élémentaires de la sécurité de l'information -

- **Confidentialité** – La technique de cryptage peut protéger l'information et la communication de la révélation non autorisée et l'accès à l'information.
- **Authentification** - La technique cryptographique telle que MAC et les signatures numériques peuvent protéger l'information contre l'usurpation d'identité.
- **Intégrité des données** - Les fonctions de hachage cryptographique jouent un rôle essentiel pour assurer les utilisateurs de l'intégrité des données.
- **Non-répudiation** - La signature numérique fournit le service non-répudiation pour se prémunir contre le différend qui pourrait survenir en raison de refus de transmettre un message par l'expéditeur.

Tous ces services fondamentaux offerts par la cryptographie ont permis à la conduite des affaires sur les réseaux utilisant les systèmes informatiques de manière extrêmement efficace et efficiente.

I.3.3 Les grandes menaces :

De façon générale, les grands types de menaces que peut subir un message lors de son échange peuvent être récapitulés à travers les points suivants :

I.3.3.1 Les attaques passives :

Avec ce type d'attaque et lors d'une communication élaborée entre deux personnes souvent nommées *Alice* et *Bob*, *Oscar* qui est l'opposant ou l'attaquant, se contente d'écouter le message tout en essayant de menacer sa confidentialité. Dans ce cas, il se peut qu'une information secrète parvienne également à une personne autre que son destinataire légal.

I.3.3.2 Les attaques actives :

Ici, *Oscar* peut menacer l'intégrité, qui sera présentée dans la section suivante. Ainsi, ces informations vont parvenir d'une personne autre que leur véritable auteur. Et comme exemple d'attaques actives, on peut citer [16]: l'usurpation d'identité (de l'émetteur ou du récepteur) ; l'altération / modification du contenu des messages ; la destruction de messages/ le retardement de la transmission ; la répétition de messages (jusqu'à engorgement) ; la répudiation de message : l'émetteur nie avoir envoyé le message.

I.4 La cryptanalyse [17] :

La cryptanalyse est une des disciplines de la cryptologie : la cryptographie a pour but de chiffrer une information en utilisant des secrets ou clés, la cryptanalyse a pour but de déchiffrer les informations. Le chiffrement existe depuis l'antiquité, la cryptanalyse existe donc aussi depuis longtemps. Le premier livre connu sur ce sujet (retrouvé en 1987) est Manuscrit sur le déchiffrement des messages cryptographiques, dû au savant arabe Al-Kindi. Où il présente la technique d'analyse fréquentielle des lettres du texte chiffré. C'est pendant la première guerre mondiale que la cryptanalyse a pris un essor important. Des méthodes nouvelles apparaissent, faisant de la cryptanalyse une science dans la deuxième partie du 20ème siècle. Le processus par lequel on tente de décrypter un message chiffré est appelé une attaque. Il existe plusieurs familles d'attaques cryptanalytiques. Un grand choix de méthodes est employé dans la cryptanalyse.

I.4.1 Attaque par force brute [18]:

Dans une attaque par force brute, un attaquant essaye chaque clé possible avec l'algorithme de décryptage sachant que par la suite l'un d'entre eux fonctionnera. Tous les algorithmes de chiffrement sont vulnérables à cette attaque. En moyenne, une attaque par force brute réussit environ 50 pour cent par le keyspace, qui est l'ensemble de toutes les clés possibles. L'objectif des cryptographes modernes est d'avoir un keyspace assez grand pour qu'il faille trop d'argent et beaucoup de temps pour accomplir une attaque par force brute.

I.4.2 Attaque à texte chiffré seul :

C'est le cas le plus difficile. On ne dispose que d'un ou de plusieurs messages chiffrés, sans avoir d'informations sur leur signification en clair. Ce cas n'est, en réalité, pas si fréquent, car on a souvent une idée du type de message que l'on attend.

I.4.3 Attaque à texte clair connu :

L'attaquant possède plusieurs paires du type message clair/message codé l'attaque est plus fréquente qu'on ne pourrait le penser. Par exemple, pendant la Guerre Mondiale, les Alliés minaient certains ports, sachant que les auteurs allemands envoyaient alors toujours le même formulaire à leur marine.

I.4.4 Attaque à texte clair choisi :

L'attaquant choisit lui-même le message à décoder. Cela peut arriver si on a un espion qui fait office d'opérateur dans le camp ennemi. Une autre possibilité est de remettre un message important à un ambassadeur. Il en avisera immédiatement son gouvernement par un message chiffré. Les algorithmes à clé publique sont un autre exemple d'attaque à texte clair choisi, puisque l'algorithme pour chiffrer est public.

I.4.5 Attaque par mot probable :

On ne connait pas tout le message clair, mais au moins une partie, par exemple, la signature, ou bien le début, etc… Ainsi, le déchiffrement de la machine Enigma pendant la Seconde Guerre Mondiale utilisa beaucoup les bulletins météo envoyés par les Allemands. Ils commençaient en effet toujours par les mêmes mots. Dans ce cas, la rigueur allemande fut bien pénalisante...

I.5 Cryptographie moderne :

La cryptographie moderne s'intéresse généralement aux problèmes de sécurité des communications, Le but est d'offrir certain nombre de services de sécurité comme la confidentialité, l'intégrité, l'authentification des données transmises.

La cryptographie moderne ce compose de deux grandes parties : La cryptographie symétrique et la cryptographie asymétrique à base de clés.

Avant de les aborder, nous allons d'abord définir la notion de **clé** qui nous sera utile tout au long de ce chapitre :

Une clé : Paramètre constitué d'une séquence de symbole et utilisé par un algorithme cryptographique, pour transformer, valider, authentifier, chiffrer ou déchiffrer des données.

On distingue généralement deux types de clés :

Les clés symétriques: il s'agit de clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.

Les clés asymétriques: il s'agit de clés utilisées dans le cas du chiffrement asymétrique (aussi appelé chiffrement à clé publique). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement.

I.5.1 Cryptographie symétrique [19]

Si la clé est unique, elle sert à chiffrer et à déchiffrer le message; on parle alors de chiffrement symétrique. Les algorithmes symétriques actuels utilisent une succession de transpositions et de substitutions complexes des valeurs du message, basées sur des opérations mathématiques et réalisées en plusieurs passes. La clé faisant partie intégrante de la fonction, il est impossible d'inverser l'algorithme sans elle, et les seules attaques envisageables consistent souvent à essayer toutes les valeurs de clés possibles. C'est la raison pour laquelle une clé symétrique suffisamment importante

(128 bits) et bien choisie est considérée comme sûre. L'algorithme symétrique le plus célèbre est le DES (Data Encryption Standard, qui fonctionnait avec des clés de 64 bits) remplacé depuis par l'AES (Advanced Encryption System, qui fonctionne avec des clés allant jusqu'à 256 bits). Les chiffrements symétriques exigent toutefois que les deux correspondants échangent au préalable la clé secrète par un canal sûr, ce qui est quasiment impossible à grande échelle Les cryptosystémes dit classiques ou à clé secrète ou encore symétrique se comporte comme une boite fermée par une serrure, les deux personnes qui veulent l'utiliser pour s'envoyer des messages doivent chacun en posséder une clé mais ils peuvent tous deux ouvrir et fermer cetteboite.

I.5.1.1 Caractéristiques : [20]

– Les clés sont identiques : KE = KD = K,

– La clé doit rester secrète,

– Les algorithmes les plus répandus sont le DES, AES, 3DES, …

– Au niveau de la génération des clés, elle est choisie aléatoirement dans l'espace des clés,

– Ces algorithmes sont basés sur des opérations de transposition et de substitution des bits du texte clair en fonction de la clé,

– La taille des clés est souvent de l'ordre de 128 bits. Le DES en utilise 56, mais l'AES peut aller jusque 256,

– L'avantage principal de ce mode de chiffrement est sa rapidité,

– Le principal désavantage réside dans la distribution des clés : pour une meilleure sécurité, on préfèrera l'échange manuel.

Malheureusement, pour de grands systèmes, le nombre de clés peut devenir conséquent. C'est pourquoi on utilisera souvent des échanges sécurisés pour transmettre les clés. En effet, pour un système à N utilisateurs, il y aura N. (N − 1)/2 paires de clés.

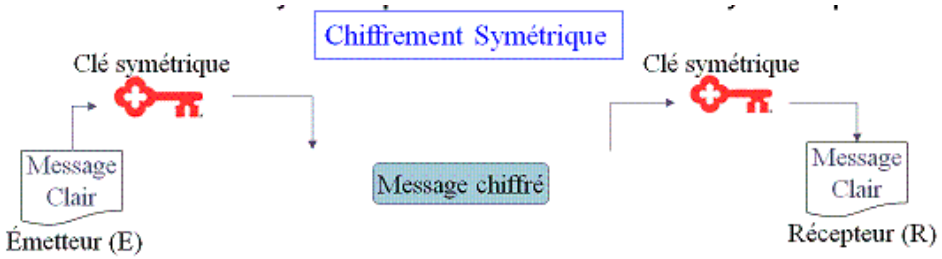


Figure 2.1 : Chiffrement symétrique [24]

I.5.1.2 DES, le chiffrement à clé secrète [21]

Le 15 mai 1973 le **NBS** (*National Bureau of Standards*, aujourd'hui appelé *NIST - National Institute of Standards and Technology*) a lancé un appel dans le *Federal Register* (l'équivalent aux Etats-Unix du *Journal Officiel* en France) pour la création d'un algorithme de chiffrement répondant aux critères suivants :

- Posséder un haut niveau de sécurité lié à une clé de petite taille servant au chiffrement et au déchiffrement
- Être compréhensible
- Ne pas dépendre de la confidentialité de l'algorithme
- Être adaptable et économique
- Être efficace et exportable

Fin 1974, IBM propose « Lucifer », qui, grâce à la NSA (National Security Agency), est modifié le 23 novembre 1976 pour donner le **DES** (*Data Encryption Standard*). Le DES a finalement été approuvé en 1978 par le NBS. Le DES fut normalisé par l'*ANSI* (*American National Standard Institute*) sous le nom de *ANSI X3.92*, plus connu sous la dénomination *DEA* (*Data Encryption Algorithm*).

I.5.1.2.1 Principe du DES

Il s'agit d'un système de chiffrement symétrique par blocs de 64 bits, dont 8 bits (un octet) servent de test de parité (pour vérifier l'intégrité de la clé). Chaque bit de parité de la clé (1 tous les 8 bits) sert à tester un des octets de la clé par parité impaire, c'est-à-dire que chacun des bits de parité est ajusté de façon à avoir un nombre impair de '1' dans l'octet à qui il appartient. La clé possède donc une longueur « utile » de 56 bits, ce qui signifie que seuls 56 bits servent réellement dans l'algorithme.

L'algorithme consiste à effectuer des combinaisons, des substitutions et des permutations entre le texte à chiffrer et la clé, en faisant en sorte que les opérations puissent se faire dans les deux sens (pour le déchiffrement). La combinaison entre substitutions et permutations est appelée **code produit**.

La clé est codée sur 64 bits et formée de 16 blocs de 4 bits, généralement notés *k1* à *k16*. Etant donné que « seuls » 56 bits servent effectivement à chiffrer, il peut exister 256 (soit 7.2*1016) clés différentes !

I.5.1.2.2 L'algorithme du DES

Les grandes lignes de l'algorithme sont les suivantes :

- Fractionnement du texte en blocs de 64 bits (8 octets) ;
- Permutation initiale des blocs ;

- Découpage des blocs en deux parties: gauche et droite, nommées G et D;
- Etapes de permutation et de substitution répétées 16 fois (appelées rondes);
- Recollement des parties gauche et droite puis permutation initiale inverse.

I.5.1.3 IDEA [22]

IDEA pour International Data Encryption Algorithm est un algorithme de chiffrement conçu par Xuejia Lai et James Massey et présenté pour la première fois en 1991. L'algorithme IDEA est un algorithme symétrique de chiffrement par bloc de 64 bits (soient 8 octets) fonctionnant avec une clé de 128 bits. La particularité de cet algorithme est que les opérations qu'il utilise en interne s'adaptent très facilement à une programmation informatique sur machine de 16 bits.

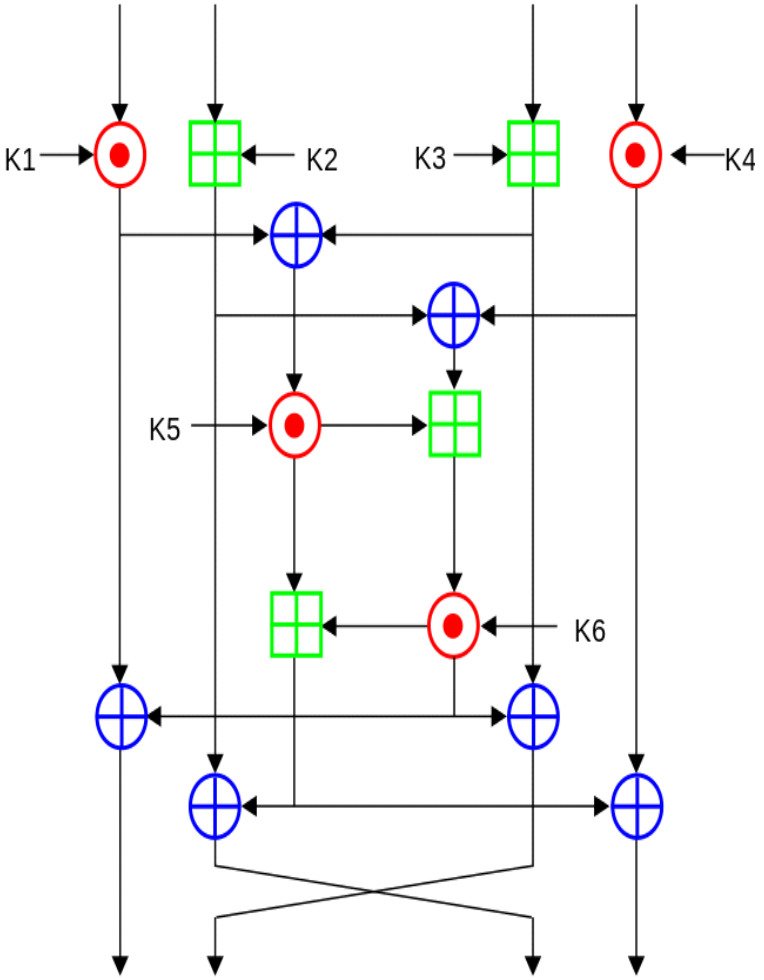


Figure 2.2 : Diagramme de fonctionnement IDEA

I.5.1.4 AES (Advanced Encryption Standard [23]

Avec le temps, et les progrès de l'informatique, les 2^{56} clés possibles du DES n'ont plus représenté une barrière infranchissable. Il est désormais possible, même avec des moyens modestes, de percer les messages chiffrés par DES en un temps raisonnable. En janvier 1997, le NIST (*National Institute of Standards and Technologies*) des Etats-Unis lance un appel d'offres pour élaborer l'AES, *Advanced Encryption System*. Le cahier des charges comportait les points suivants :

- évidemment, une grande sécurité.
- une large portabibilité : l'algorithme devant remplacer le DES, il est destiné à servir aussi bien dans les cartes à puces, aux processeurs 8 bits peu puissants, que dans des processeurs spécialisés pour chiffrer des miliers de télécommunications à la volée.
- la rapidité.
- une lecture facile de l'algorithme, puisqu'il est destiné à être rendu public.
- techniquement, le chiffrement doit se faire par blocs de 128 bits, les clés comportant 128,192 ou 256 bits.

Au 15 juin 1998, date de la fin des candidatures, 21 projets ont été déposés. Certains sont l'oeuvre d'entreprises (IBM,...), d'autres regroupent des universitaires (CNRS,...), les derniers sont écrits par à peine quelques personnes. Pendant deux ans, les algorithmes ont été évalués par des experts, avec forum de discussion sur Internet, et organisation de conférences. Le 2 octobre 2000, le NIST donne sa réponse : c'est le Rijndael qui est choisi, un algorithme mis au point par 2 belges, Joan Daemen et Vincent Rijmen. Depuis, le Rijndael, devenu AES, a été largement déployé et a remplacé progressivement le DES.

I.5.1.4.1 Principe de fonctionnement de l'AES

Le Rijndael procède par blocs de 128 bits, avec une clé de 128 bits également. Chaque bloc subit une séquence de 5 transformations répétées 10 fois:

- Addition de la clé secrète (par un ou exclusif).
- Transformation non linéaire d'octets : les 128 bits sont répartis en 16 blocs de 8 bits (8 bits=un octet), eux-même dispatchés dans un tableau 4×4. Chaque octet est transformé par une fonction non linéaire S. S peut être simplement vu comme une substitution sur les entiers compris entre 1 et 256. En particulier, elle peut être implantée sur ordinateur par un simple tableau.

- Décalage de lignes : les 3 dernières lignes sont décalées cycliquement vers la gauche : la 2ème ligne est décalée d'une colonne, la 3ème ligne de 2 colonnes, et la 4ème ligne de 3 colonnes.
- Brouillage des colonnes : Chaque colonne est transformée par combinaisons linéaires des différents éléments de la colonne (ce qui revient à multiplier la matrice 4×4 par une autre matrice 4×4). Les calculs sur les octets de 8 bits sont réalisés dans le corps à 2^8 éléments.
- Addition de la clé de tour : A chaque tour, une clé de tour est générée à partir de la clé secrète par un sous-algorithme (dit de cadencement). Cette clé de tour est ajoutée par un ou exclusif au dernier bloc obtenu.

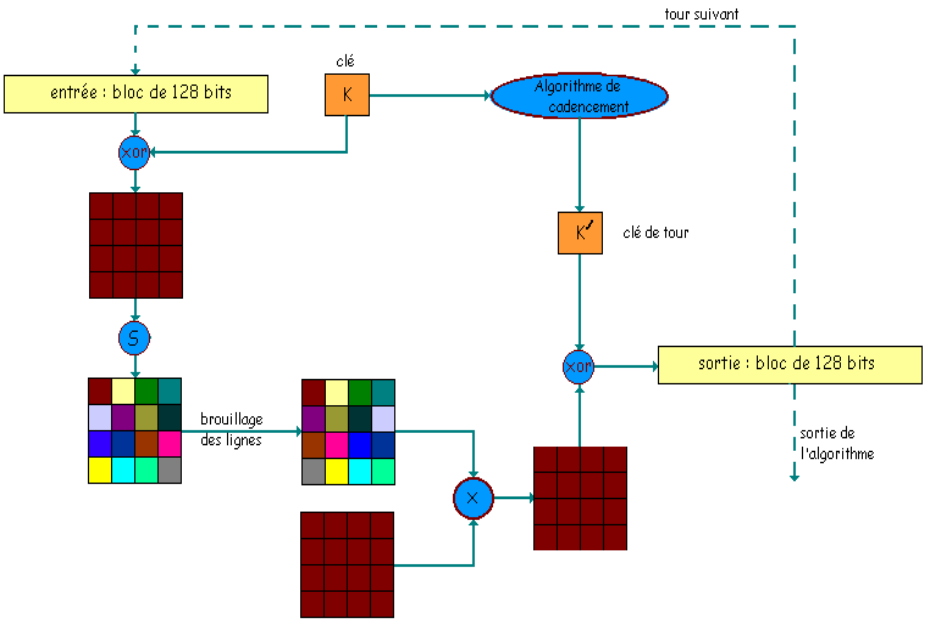


Figure 2.3 : Diagramme de fonctionnement AES [23]

La description précise des opérations effectuées dans l'AES nécessite certaines connaissances d'algèbre, notamment de l'arithmétique des polynômes dans le corps à 2^8 éléments!

I.5.2 Cryptographie asymétrique (ou chiffrement à clés publiques) [2]

I.5.2.1 le principe du chiffrement à clé publique

Le principe de chiffrement asymétrique (appelé aussi chiffrement à clés publiques) est apparu en 1976, avec la publication d'un ouvrage sur la cryptographie par Whitfield Diffie et Martin Hellman. Dans un crypto système asymétrique (ou crypto système à clés publiques), les clés existent par paires (le terme de bi-clés est généralement employé) :

- Une clé publique pour le chiffrement ; Martin Hellman.
- Une clé secrète pour le déchiffrement.

Ainsi, dans un système de chiffrement à clé publique, les utilisateurs choisissent une clé aléatoire qu'ils sont seuls à connaître (il s'agit de la clé privée). A partir de cette clé, ils déduisent chacun automatiquement un algorithme (il s'agit de la clé publique). Les utilisateurs s'échangent cette clé publique au travers d'un canal non sécurisé. Lorsqu'un utilisateur désire envoyer un message à un autre utilisateur, il lui suffit de chiffrer le message à envoyer au moyen de la clé publique du destinataire (qu'il trouvera par exemple dans un serveur de clés tel qu'un annuaire LDAP). Ce dernier sera en mesure de déchiffrer le message à l'aide de sa clé privée (qu'il est seul à connaître).

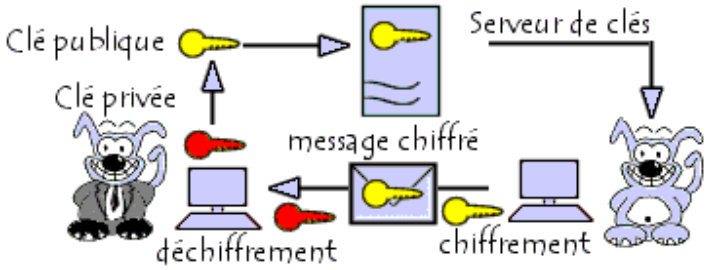


Figure 2.4 : chiffrement à clé publique [2]

Ce système est basé sur une fonction facile à calculer dans un sens (appelée *fonction à trappe à sens unique* ou en anglais *one-way trapdoor function*) et mathématiquement très difficile à inverser sans la clé privée (appelée *trappe*).

- **Avantages et inconvénients**

Le problème consistant à se communiquer la clé de déchiffrement n'existe plus, dans la mesure où les clés publiques peuvent être envoyées librement. Le chiffrement par clés publiques permet donc à des personnes d'échanger des messages chiffrés sans pour autant posséder de secret en commun. En contrepartie, tout le challenge consiste à (s') assurer que la clé publique que l'on récupère est bien celle de la personne à qui l'on souhaite faire parvenir l'information chiffrée !

I.5.2.2 Protocole d'échange de clés de Diffie et Hellman [23]

Parallèlement à leur découverte du principe de la cryptographie à clé publique, Diffie et Hellman ont proposé en 1976 un protocole d'échange de clés totalement sécurisé.

Le problème est le suivant. Alice et Bob veulent s'échanger un message crypté en utilisant un algorithme nécessitant une clé K. Ils veulent s'échanger cette clé K, mais ils ne disposent pas de canal sécurisé pour cela. Le protocole d'échange de clés de Diffie et Hellman répond à ce problème lorsque K est un nombre entier. Il repose sur l'arithmétique modulaire, et sur le postulat suivant:

Étant donnés des entiers p, a, x, avec p premier et $1 \leq a \leq p-1$:

- il est facile de calculer l'entier $y = a^x \pmod{p}$.
- si on connait $y = a^x \pmod{p}$, a et p, il est très difficile de retrouver x, pourvu que p soit assez grand.

Retrouver x connaissant a^x (mod p), a et p s'appelle résoudre le problème du **logarithme discret**. Comme pour la factorisation d'entiers, c'est un problème pour lequel on ne dispose pas d'algorithme efficace. Expliquons maintenant comment Alice et Bob peuvent s'échanger une clé secrète par le protocole de Diffie-Hellman. Ils font des actions en parallèle, que l'on décrit dans le tableau suivant :

	Alice	Bob
Étape 1 :	Alice et Bob choisissent ensemble un grand nombre premier p et un entier $1 \leq a \leq p-1$. Cet échange n'a pas besoin d'être sécurisé.	
Étape 2 :	Alice choisit secrètement x1.	Bob choisit secrètement x2.
Étape 3 :	Alice calcule $y1=a^{x1}$ (mod p).	Bob calcule $y2=a^{x2}$ (mod p).
Étape 4 :	Alice et Bob s'échangent les valeurs d'y1 et y2. Cet échange n'a pas besoin d'être sécurisé.	
Étape 5 :	Alice calcule $Y_2^{x1} = (a^{x2})^{x1} = a^{x1x2}$ (mod p)et appelle ce nombre K, la clé secrète à partager avec Bob.	Bob calcule yx21= (ax1) x2=ax1x2 (mod p) y1x2= (ax1) x2=ax1x2 (mod p) et appelle ce nombre KK, la clé secrète à partager avec Alice.

Tab II : protocole de Diffie-Hellman.

À la fin du protocole, Alice et Bob sont donc en possession d'une même clé secrète K, qu'ils ne se sont pas échangés directement. Si quelqu'un a espionné leurs conversations, il connait p, a, y_1p, a, y_1 et y_2y_2. Il ne peut pas retrouver K comme le font Alice ou Bob, car il lui manque toujours l'une des informations nécessaires, à savoir x_1 ou x_2. Et il ne peut pas retrouver x1 connaissant $y=a^{x1}$ (modp) $y=a^{x1}$, a et p, puisque la résolution du logarithme discret est un problème difficile.

Cette découverte de Diffie et Hellman est une vraie révolution dans l'histoire de la cryptographie. Le problème de l'échange des clés est en effet résolu.

Ce protocole a cependant un défaut : il exige la simultanéité des actions d'Alice et de Bob. Si Alice veut envoyer un e-mail à Bob alors que celui dort ou n'est simplement pas connecté, elle ne pourra pas le faire immédiatement. C'est pourquoi ce protocole fut en réalité très vite supplanté par les méthodes de chiffrement à clé publique de type RSA, pour lesquels on met à la disposition de tout le monde une clé publique. Toutefois, il est utilisé pour les problèmes d'appariement de deux objets dans la technologie Bluetooth.

I.5.2.3 RSA (Rivest, Shamir et Adleman)

La méthode de cryptographie RSA a été inventée en 1977 par Ron Rivest, Adi Shamir et Len Adleman, à la suite de la découverte de la cryptographie à clé publique par Diffie et Hellman. Le RSA est encore le système cryptographique à clé publique le plus utilisé de nos jours. Il est intéressant de remarquer que son invention est fortuite : au départ, Rivest, Shamir et Adleman voulaient prouver que tout système à clé publique possède une faille.

I.5.2.3.1 Principe de fonctionnement :

Si Bob souhaite recevoir des messages en utilisant le RSA, il procède de la façon suivante :

Création des clés :

Bob crée 4 nombres p,q, e et d :

p et q sont deux grands nombres premiers distincts. Leur génération se fait au hasard, en utilisant un algorithme de test de primalité probabiliste.

e est un entier premier avec le produit (p-1)(q-1).

d est tel que e.d=1 modulo (p-1)(q-1).

Autrement dit, e.d-1 est un multiple de (p-1)(q-1). On peut fabriquer d à partir de e, p et q, en utilisant l'algorithme d'Euclide.

Distribution des clés :

Le couple (n,e) constitue la clé publique de Bob. Il la rend disponible par exemple en la mettant dans un annuaire. Le couple (n,d) constitue sa clé privée. Il la garde secrète.

Envoi du message codé :

Alice veut envoyer un message codé à Bob. Elle le représente sous la forme d'un ou plusieurs entiers M compris entre 0 et n-1. Alice possède la clé publique (n,e) de Bob. Elle calcule C=M.e mod n. C'est ce dernier nombre qu'elle envoie à Bob.

Réception du message codé :

Bob reçoit C, et il calcule grâce à sa clé privée D=C.d (mod n). D'après un théorème du mathématicien Euler, D=M.d.e=M (mod n). Il a donc reconstitué le message initial.

I.5.3 Signature électronique

I.5.3.1 Introduction à la notion de signature électronique [2]

Le paradigme de **signature électronique** (appelé aussi signature numérique) est un procédé permettant de garantir l'authenticité de l'expéditeur (fonction d'authentification) et de vérifier l'intégrité du message reçu.

La signature électronique assure également une fonction de non-répudiation, c'est-à-dire qu'elle permet d'assurer que l'expéditeur a bien envoyé le message (autrement dit elle empêche l'expéditeur de nier avoir expédié le message).

I.5.3.2 L'objectif des signatures électroniques [23]

Les signatures électroniques sont l'analogue électronique des signatures sur papier. Un bon protocole de signature électronique doit avoir les propriétés suivantes :

- il doit garantir l'**authenticité** : la signature ne doit pas pouvoir être imitée. Autrement dit, si un document porte la signature électronique d'Alice, on doit pouvoir être sûr que c'est effectivement Alice qui l'a signé.
- il doit garantir la **non-répudiation** : si Alice a signé un document, elle ne doit pas pouvoir se rétracter et pouvoir prétendre ensuite qu'elle ne l'a pas signé.
- il doit garantir l'**intégrité** : en comparant la signature et le message, on doit pouvoir être sûr que le message n'a pas été altéré lors de l'envoi.

Tiny Encryption Algorithm (TEA)

II.1 Introduction

Après avoir présenté les notions générales de la cryptographie, à présent nous nous intéresserons à l'algorithme de chiffrement symétrique TEA (Tiny Encryption Algorithm) Nous soulignerons le fonctionnement et les motivations qui nous ont poussé à choisir cette algorithme

II.2 Présentation de l'algorithme TEA [25]

TEA (Le Tiny Encryption Algorithm) est l'un des algorithmes cryptographiques les plus rapides et les plus efficaces dans L'existence. Il a été développé par David Wheeler et Roger Needham au Computer Laboratory de Cambridge University.

L'algorithme de cryptage Tiny Encryption est un chiffrement **Feistel** à 32 tours qui fonctionne sur des blocs 64 bits et Utilise une clé de 128 bits.

En 1996, Kelsey a trouvé que la taille effective de la clé TEA était de 126 bits et ce résultat a entraîné une attaque sur la console de jeu Xbox de Microsoft où TEA a été utilisé comme une fonction hash

En 1998. Pour corriger les faiblesses de Block TEA, Needham et Wheeler ont conçu le TEA corrigé ou XXTEA. Ce chiffre utilise un réseau Feistel non équilibré et fonctionne sur des messages à longueur variable. Le nombre de tours est déterminé par la taille du bloc, mais il est au moins six.

Il Crypte 64 bits de données à la fois en utilisant une clé de 128 bits. Il semble très résistant à la cryptanalyse différentielle, et atteint une diffusion complète (où une différence d'un bit dans le texte en clair entraînera environ 32 bits de différences dans le texte chiffré) après seulement six tours.

La performance sur un ordinateur de bureau ou un poste de travail moderne est très impressionnant.

En 2002-2010. Un certain nombre de résultats de cryptalyse sur la famille TEA a été signalée dans cette période.

II.3 Les caractéristiques de l'algorithme TEA [26]

Comme son nom l'indique, l'algorithme de chiffrement TEA est petit dans la taille. Tous dits, il peut être mis en application dans quelques lignes de code de programmation. C'est important parce qu'il signifie qu'il peut être inclus dans presque n'importe quel type de package de logiciel, même ceux avec des contraintes sérieuses de l'espace. Par exemple, le logiciel à votre téléphone portable, ou le logiciel pour GPS dans votre voiture. Il est également facile à comprendre les étapes de chiffrage, la rendant simple pour le mettre en application et le maintenir.

II.4 Principe de fonctionnement du TEA

Avant de présenter le principe de fonctionnement de l'algorithme TEA, nous allons d'abord définir quelque notions qui nous serons utiles tout au long de cette partie.

Delta : c'est une constante entière non signé égal à 2654435769 (0x9E3779B9 en hexadécimale). Ce nombre qui est dérivé du nombre d'or

Delta= (√5 -1)*2 ^ 32

Delta$_i$: est un entier non signé égal à **(Delta * i) mod 2^{32}**, tel que i est le numéro du tour

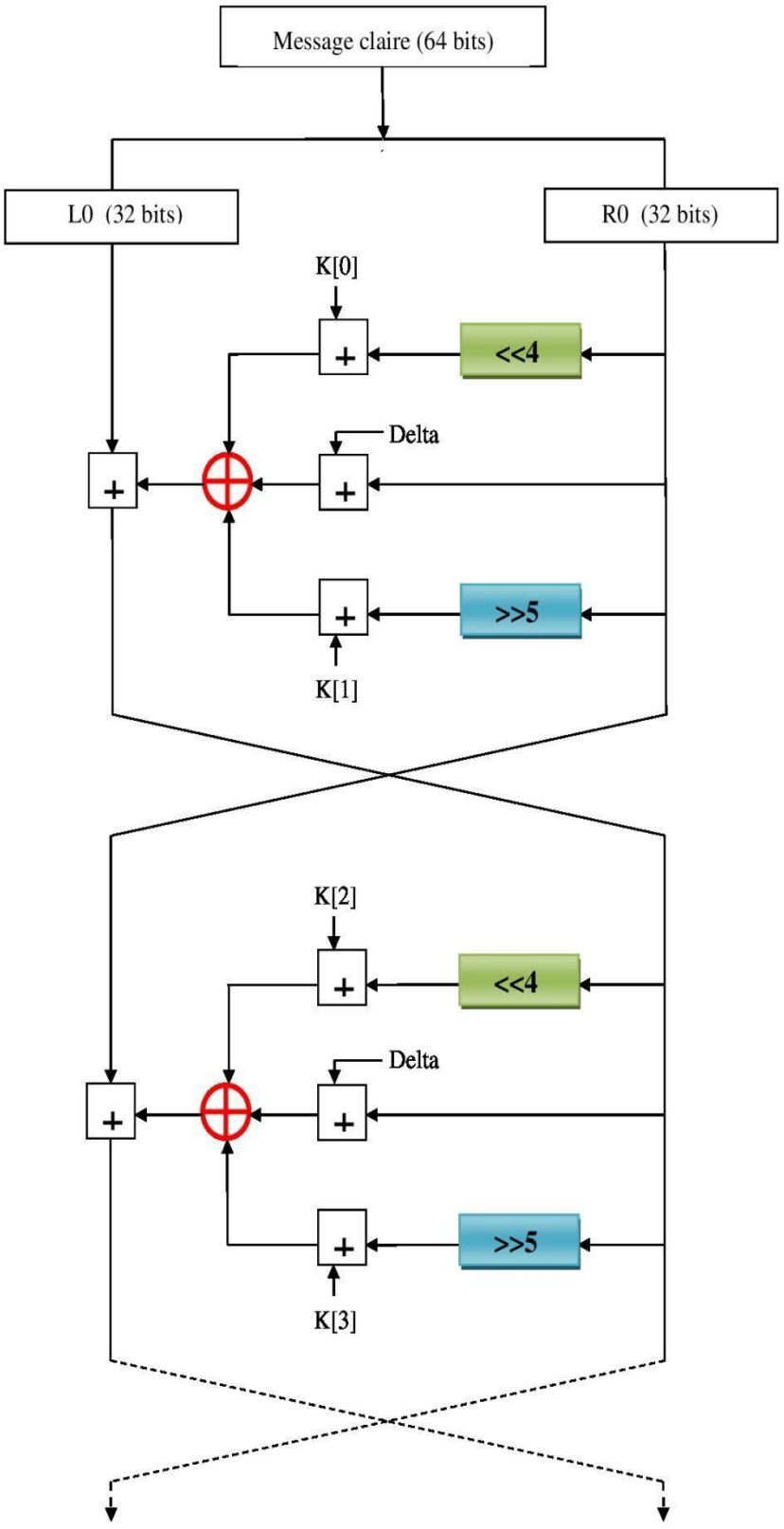
II.4.1 Chiffrement

C'est un type de structure de Feistel bien que l'addition et la soustraction soient employées en tant qu'opérateurs réversibles plutôt que XOR

L'algorithme se fonde sur l'utilisation alternative de XOR et ADD, et un double décalage de tous les bites de la clé et des donnés.

TEA a une clé de 128 bits qui est divisée en quatre sous-clés de 32 bits, qui peut être considérée comme K [0-3]

Des différents multiples de delta (Delta$_i$) sont utilisés dans chaque tour (mod 2 ^ 32), de sorte qu'aucun bit du multiple ne changera fréquemment


Message claire (64 bits)
L0 (32 bits)
R0 (32 bits)
K[0]
<<4
Delta
>>5
K[1]
K[2]
<<4
Delta
>>5
K[3]

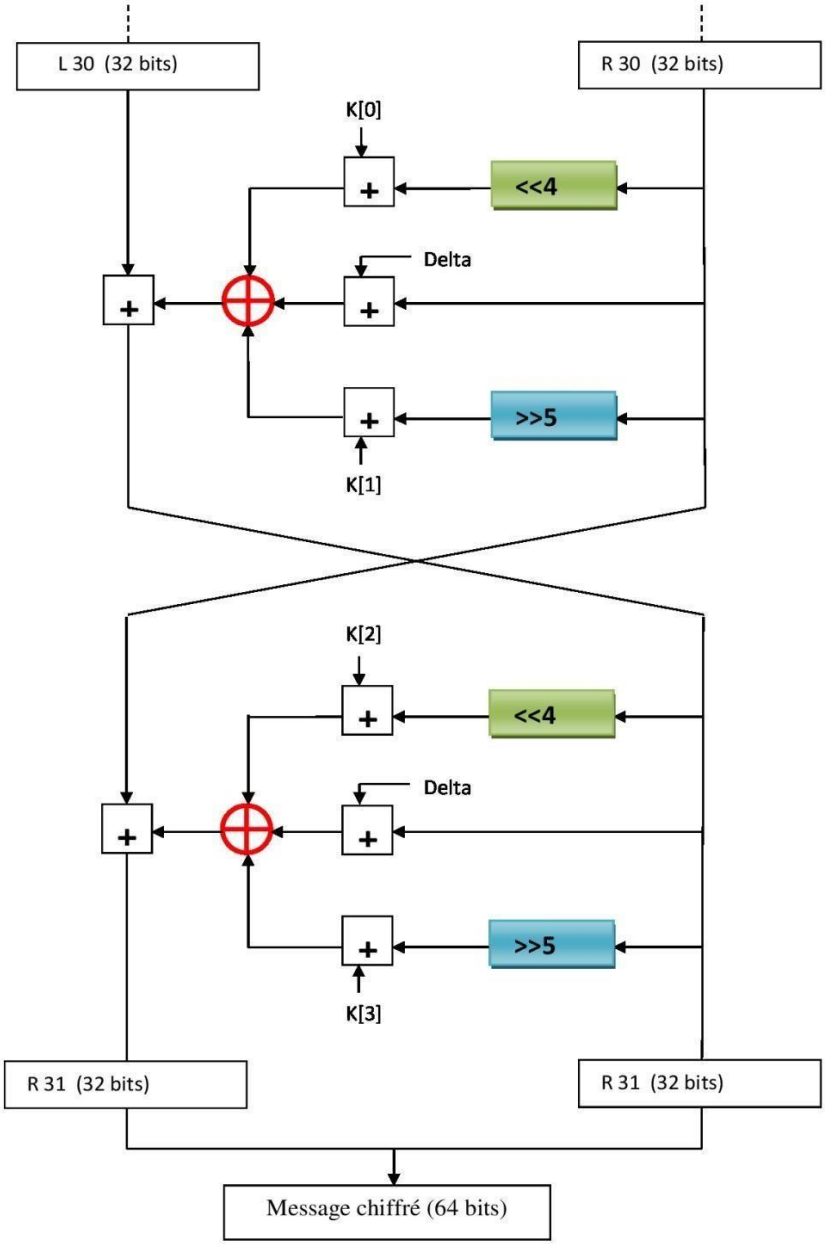


Figure 2.5 : Diagramme de fonctionnement chiffrement TEA

Le diagramme ci-dessus montre 2 rondes Feistel de TEA. Ces deux rondes en composent un tour du cycle TEA.

Le chiffrement commence par un bloc de données de 64 bits qui se divise en deux blocs de 32 bits que nous appellerons L et R. L est le côté gauche du bloc (représenté en haut à gauche du diagramme), et R est le côté droit (représenté en haut à droite du diagramme).

Ces blocs sont échangés par Round, on peut voir cet échange là où les deux lignes se croisent au milieu du diagramme.

Chaque tour se divise en deux rounds

- Dans le premier round, R est utilisé comme un input à plusieurs opérations.

1. R passe par un décalage à gauche de 4, puis il est ajouté à K[0]

2. R est ajouté à $Delta_i$

3. R passe par un décalage à droite de 5 puis est ajouté à K[1]

4. Une opération XOR est ensuite appliquée au résultat de ces dernières trois opérations

Et enfin, le résultat de l'opération XOR est ajouté à L

5. Ce résultat devient alors R pour le deuxième round

- Dans le deuxième round, après l'échange des deux blocs :

- Le résultat du premier round devient R pour le deuxième round
- R du premier round devient L pour le deuxième round

1. R passe par un décalage à gauche de 4, puis il est ajouté à K[2]

2. R est ajouté à $Delta_i$

3. R passe par un décalage à droite de 5 puis est ajouté à K[3]

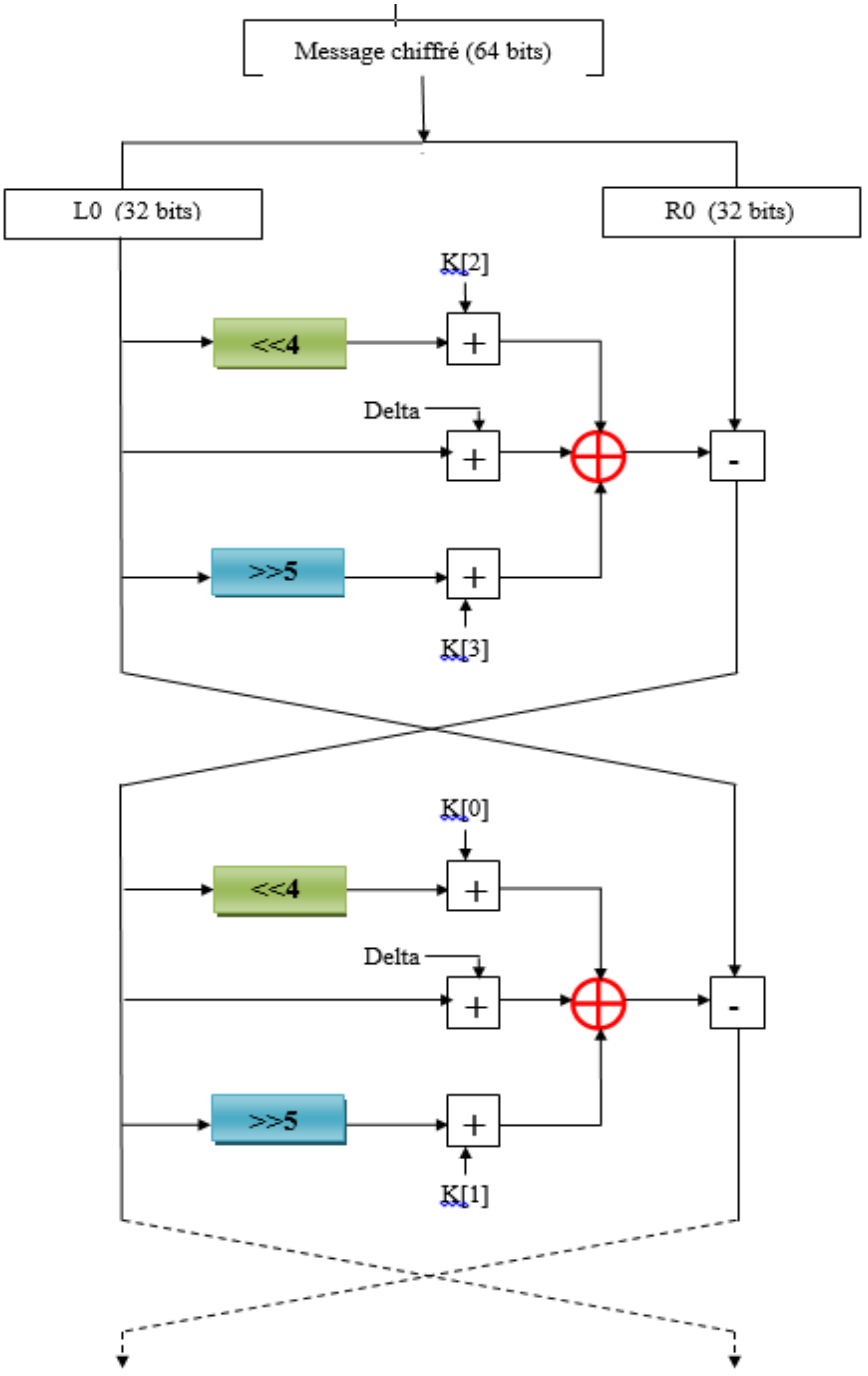
4. Une opération XOR est ensuite appliquée au résultat de ces dernières trois opérations

Et enfin, le résultat de l'opération XOR est ajouté à L

5. Ce résultat devient alors R pour le prochain tour.

II.4.2 Déchiffrement :

Le Déchiffrement est essentiellement le même que le processus de chiffrement; le texte chiffré est utilisé comme entrée de l'algorithme, mais les sous-clés K [i] sont utilisées dans l'ordre inverse.



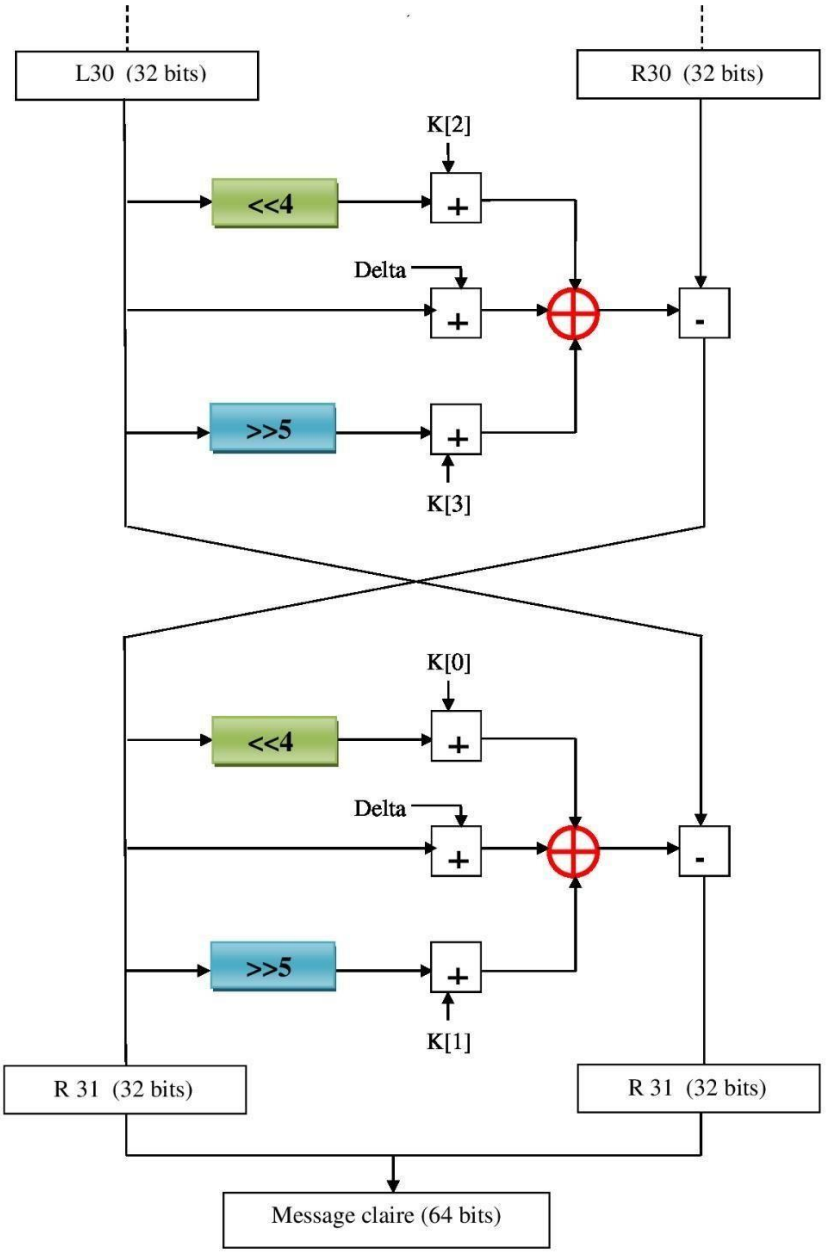


Figure 2.6 : Diagramme de fonctionnement Déchiffrement TEA

- Dans le premier round, cette fois c'est L qui est utilisé comme un input à plusieurs opérations.

1. L passe par un décalage à gauche de 4, puis il est ajouté à K[2]

2. L est ajouté à $Delta_i$

3. L passe par un décalage à droite de 5 puis est ajouté à K[3]

4. Une opération XOR est ensuite appliquée au résultat de ces dernières trois opérations

Et enfin, le résultat de l'opération XOR est soustrait de R

5. Ce résultat devient alors L pour le deuxième round

- Dans le deuxième round, après l'échange des deux blocs :

- le résultat du premier round devient L pour le deuxième round
- R du premier round devient R pour le deuxième round

1. L passe par un décalage à gauche de 4, puis il est ajouté à K[0]

2. L est ajouté à $Delta_i$

3. L passe par un décalage à droite de 5 puis est ajouté à K[1]

4. Une opération XOR est ensuite appliquée au résultat de ces dernières trois opérations

Et enfin, le résultat de l'opération XOR est soustrait de R

5. Ce résultat devient alors L pour le prochain tour.

II.5 Sécurité du TEA [27]

Très sécurisé. Il n'y a pas eu de cryptanalyses réussies de TEA. On pense qu'il est aussi sûr que l'Algorithme IDEA, conçu par Massey et Xuejia Lai.

Il utilise la même technique de groupes algébriques mixtes que l'IDEA, mais c'est beaucoup plus simple, donc plus rapide. De plus, il est de domaine public, alors qu'IDEA est breveté par Ascom-Tech AG en Suisse.

Don Coppersmith et Massey d'IBM ont montré indépendamment que les opérations de mixage à partir des groupes algébriques orthogonaux effectuent les fonctions de diffusion et de confusion qu'un code de bloc traditionnel mettra en œuvre avec des boîtes P et S. En tant que routine simple de cryptage enfichable, c'est génial. Le code est léger et assez portable à utiliser n'importe où. Il constitue même un générateur de nombres aléatoires génial pour les simulations Monte Carlo et Les faiblesses mineures identifiées par David Wagner à Berkeley sont peu susceptibles d'avoir un impact sur le Monde réel, et vous pouvez toujours mettre en œuvre la nouvelle variante de TEA qui les traite.

II.6 Comparaison avec d'autres algorithmes :

	Propriétés	**Tiny Encryption Algorithm (TEA)**	**Data Encryption Standard (DES)**	**Rivest-Shamir-Adlema n (RSA)**	**Advance Encryption Standard (AES)**
1	**Taille de la clé**	128 bits	56 bits	[1,024 -4,096] bit	128, 192 ou 256 bits
2	**Nbr de round**	64 (32 tours)	16	1	10, 12 ou 14 (dépendant de la clé)
3	**Taille du block**	64bits	64bits	Pas de block	128 bits
4	**La structure**	Réseau Feistel	Réseau Feistel	Sans structure	Réseau de substitution/permutation
5	**Point forts**	Fournit à la fois un cryptage, et une transmission de données sécurisée	Les implémentations matérielles de DES sont très rapides.	Le plus grand avantage de RSA est qu'il utilise des clés asymétriques	Facilité de calcul, besoin en ressources et mémoire très faibles

Tab.II.2 : Comparaison avec d'autres algorithmes

II.7 Conclusion

Dans cette première partie nous avons vu quelques notions de base sur la cryptographie, Ainsi nous avons étudié la cryptographie moderne avec ses deux parties cryptographie symétrique cryptographie asymétrique et leurs différents algorithmes nécessaires pour la protection des informations personnelles ou privés.

Ensuite, nous avons détaillé le fonctionnement de l'algorithme TEA (Tiny Encryption Algorithm), et nous avons vu toutes les étapes du déroulement de cet algorithme pour le chiffrement et pour le déchiffrement, et finalement nous avons fait une petite comparaison entre le TEA et chacun des algorithmes DES, RSA, et AES

Dans le chapitre à suivre nous allons introduire quelques généralités sur les systèmes embarqués, en suite nous allons présenter le système d'exploitation Android.

PARTIE I : SYSTEME EMBARQUE

III.1.1 Introduction :

Les systèmes embarquées sont de plus en plus nombreux et de plus en plus complexes, le rôle des systèmes embarqués est primordial au quotidien.

Un système embarqué se compose d'une partie " matériel " (hard) et d'une partie " logiciel " (soft). Ces deux parties communiquent en permanence en utilisant des composants actifs (qui dialoguent en temps réel avec une base qui alerte l'utilisateur en cas de problème) et passifs (qui attendent que l'utilisateur les interroge pour obtenir une information de leur part).

La machine et le logiciel sont intimement liés et noyés dans le matériel et ne sont pas aussi facilement discernables comme dans un environnement de travail classique de type PC. On mentionne les contraintes physiques fortes: dimensions, poids, taille, autonomie, consommation, fiabilité, contraintes temporelles (temps réels).

III.1.2 Définition : [28]

Un système embarqué peut être défini comme un système électronique et informatique autonome, qui est dédié à une tâche bien précise, possédant des ressources d'ordre spatial (taille limitée) et énergétique (consommation restreinte) limitées. Le terme de « Système Embarqué » désigne aussi bien le matériel que le logiciel utilise. Les systèmes embarqués exécutent des tâches prédéfinies et ont un cahier des charges contraignant à remplir, qui peut être d'ordre : - D'espace compté, avec un espace mémoire limité de l'ordre de quelques Mo maximum. Ils font de plus très souvent appel à l'informatique et aussi aux systèmes « temps réel ». - De consommation énergétique le plus faible possible, due à 'utilisation de sources autonomes, batteries, panneaux solaires - Temporelle, dont le temps d'exécution de tâches est déterminé, Des compétences pluridisciplinaires en termes d'électronique (de commande et de puissance), d'automatique et d'informatique industrielle sont donc nécessaires pour concevoir de tels systèmes.

III.1.3 Brève histoire des systèmes embarqués :

1967 : Apollo Guidance Computer, premier système embarqué. Environ un millier de circuits intégrés identiques (portes NAND).

1960-1970 : Missile Minuteman, guidé par des circuits intégrés.

1971 : Intel produit le 4004, premier microprocesseur, à la demande de Busicom. Premier circuit générique, personnalisable par logiciel.

1972 : lancement de l'Intel 8008, premier microprocesseur 8 bits (48 instructions, 800kHz).

1974 : lancement du 8080, premier microprocesseur largement diffusé. 8 bits, (64KB d'espace adressable, 2MHz - 3MHz).

1978 : création du Z80, processeur 8 bits.

1979 : création du MC68000, processeur 16/32 bits.

III.1.4 Architecture d'un système embarqué [29]

L'architecture d'un système embarqué se définie par le schéma suivant:

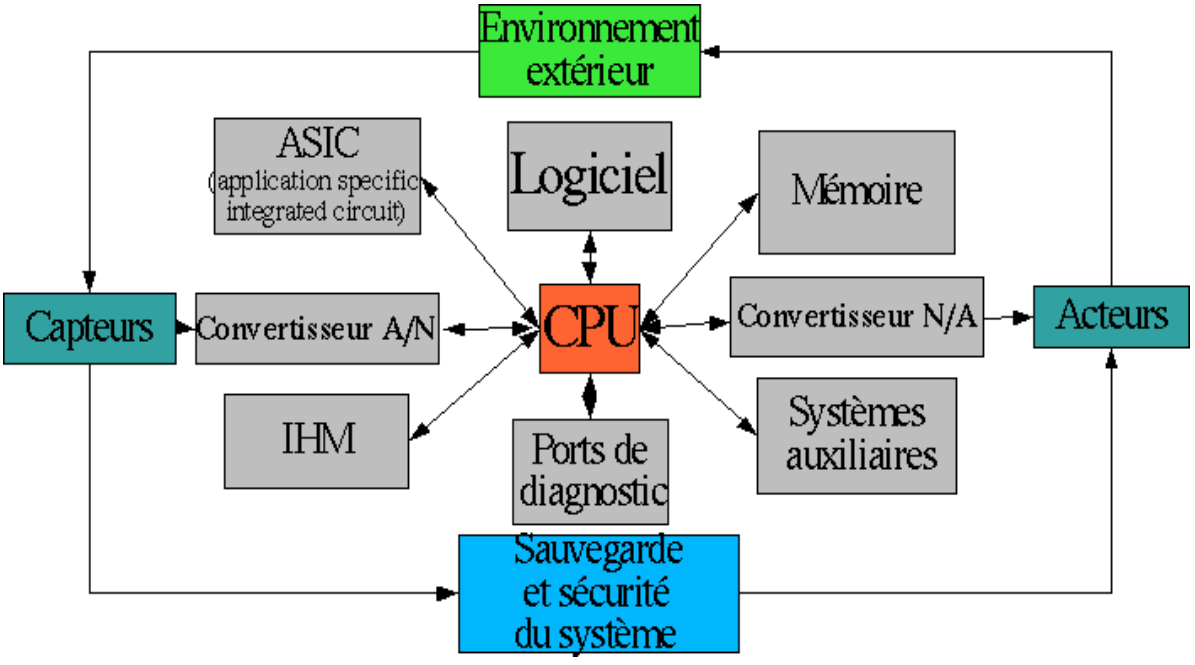


Figure 3.1 : Architecture d'un système embarqué [29]

Cette architecture peut varier selon les systèmes: on peut par exemple, ne pas trouver de systèmes auxiliaires dans de nombreux systèmes embarqués autonome et indépendants. En revanche, l'architecture de base est la plupart du temps composée d'une unité centrale de traitement (CPU), d'un système d'exploitation qui réside parfois

uniquement en un logiciel spécifique (ex: routeur), ou une boucle d'exécution (ex: ABS). De même l'interface IHM n'est pas souvent existante, mais est souvent utile pour reconfigurer le système ou vérifier son comportement. Le fonctionnement du système se résume ainsi:

- Il reçoit des informations de l'environnement extérieur qu'il converti en signal numérique
- L'unité de traitement composée du CPU, de la mémoire, du logiciel, de l'ASIC et éventuellement de système externes traite l'information
- Le traitement génère éventuellement une sortie qui est envoyée vers la sortie, les systèmes auxiliaire, les ports de monitoring ou l'IHM

III.1.5 Caractéristiques principales d'un système embarqué : [29]

Les principales caractéristiques d'un système embarqué sont les suivantes :
• C'est un système principalement numérique.
• Il met en œuvre généralement un processeur.
• Il exécute une application logicielle dédiée pour réaliser une fonctionnalité précise et n'exécute donc pas une application scientifique ou grand public traditionnelle.
• Il n'a pas réellement de clavier standard (Bouton Poussoir, clavier matriciel...). L'affichage est limité (écran LCD...) ou n'existe pas du tout.
• Ce n'est pas un PC en général mais des architectures similaires (x86) basse consommation sont de plus en plus utilisées pour certaines applications embarquées. De ce constat, on peut voir :
• Qu'un PC standard peut exécuter tout type d'applications car il est généraliste alors qu'un système embarqué n'exécute qu'une application dédiée.
• Que l'interface IHM peut être aussi simple qu'une led qui clignote ou aussi complexe qu'un cockpit d'avion de ligne.
• Que des circuits numériques ou des circuits analogiques sont utilisés en plus pour augmenter les performances du système embarqué ou sa fiabilité.

III.1.6 Les avantages d'un système embarqué [30]

Les systèmes embarqués se réfèrent à des systèmes informatiques qui sont créés pour exécuter un nombre défini de tâches ou fonctions. Ils sont noyés dans le sens où le système informatique est incorporé dans un appareil à côté matériel nécessaire.

- ✓ **Conception et efficacité**

Le noyau central de traitement dans les systèmes embarqués est généralement moins complexe, ce qui rend plus facile à entretenir. La fonction limitée nécessaire des systèmes embarqués leur permet d'être conçus pour exécuter plus efficacement leurs fonctions.

- ✓ **Coût**

La rationalisation de maquillage de la plupart des systèmes embarqués permet à leurs parties à être plus petits moins cher à produire.

- ✓ **Accessibilité**

Les systèmes embarqués sont difficiles à entretenir, car ils sont à l'intérieur d'une autre machine, donc un plus grand effort est fait pour développer les soigneusement. Cependant, si quelque chose ne va pas avec certains systèmes embarqués, ils peuvent être trop inaccessibles pour réparer. Cette préoccupation est parfois abordée dans la phase de conception, comme en programmant un système embarqué de sorte qu'il n'affecte pas les systèmes connexes négativement lorsque dysfonctionnement des systèmes embarqués.

- ✓ **Maintenance**

Sont plus faciles à entretenir car la puissance fournie est intégré dans le système et ne nécessite pas de maintenance à distance. Licenciements de systèmes embarqués n'impliquent pas la programmation redondante et l'entretien impliqués dans d'autres modèles de systèmes.

III.1.7 Contraintes d'un système embarqué :

Secteur d'activité	Contraintes
Équipements scientifiques	Performances, fiabilité, coût
Équipements militaires et aérospatiaux	Performances, fiabilité, pérennité, intégration
Transports	Fiabilité, coût, interactivité
Informatique industrielle	Fiabilité, coût, pérennité
Matériel de bureau	Performance, coût, standardisation
Réseau et télécommunications	Performance, fiabilité, intégration
Électronique grand public	Performance, coût, design / intégration

Tab.III.1 : Contraintes d'un système embarqué

III.1.8 Les contraintes de développement sur un système embarqué :

Les contraintes de développements sur un système embarqué sont nombreuses et se rapprochent assez de celles rencontrées en informatique en général:

- La complexité des algorithmes utilisés doit être relativement faible d'autant plus que les performances et la puissance des systèmes est moindre.
- L'absence de MMU limite la taille des applications.
- L'absence de media de stockage permanent...
- De nombreuses applications des systèmes embarquées sont des applications temps réels: On doit souvent ajouter une contrainte temps-réel.
- L'utilisation d'outils de Génie Logiciel est fortement conseillée
- La conception orientée objet est préconisée dès que cela est possible

III.1.9 Contraintes matérielles d'un système embarqué

Le tableau suivant résume les contraintes matérielles liées aux systèmes embarquées: Généralement, ce sont des contraintes de taille. La taille du système impose des contraintes de performances des systèmes.

Besoin	Miniature	Petit	Moyen	Haut de gamme	PC embarqué	PC haute disponibilité
Taille de la RAM	<0.1 Mo	0.1 Mo-4Mo	2Mo-8Mo	8Mo-32Mo	>64Mo	>xMo
Taille de la ROM/Flash	<0.1 Mo	0.1 Mo-4Mo	2Mo-8Mo	8Mo-32Mo	>64Mo	>xMo
Processeur	ARM, ColdFire	ARM, ColdFire, x86, PowerPC	x86, Power P C	x86, PowerPC	x86, PowerPC	Pentium, PowerPC
Caracterisitiques	MMU* optionnelle	MMU* optionnelle, Carte unité centrale, System on Chip (SoC)	Carte unité centrale, System on Chip (SoC)	Carte unité centrale, System on Chip (SoC)	Carte unité centrale, System on Chip (SoC)	Compact PCI
Exemple d'application	Caméra numérique, PDA, téléphone	Caméra numérique, PDA, téléphone, décodeur, équipement réseau, imprimant	Décodeur, équipement réseau, imprim ante	Décodeur, équipement réseau, imprimante	Décodeur, équipement réseau, imprimante	Commutateur téléphonique, routeur multiprotocole, serveur central

Tab.III.2 : Contraintes matérielles d'un système embarqué

III.1.10 Utilisation d'Android pour des systèmes embarqués et temps réel

Le système d'exploitation Android a su conquérir en quelques années le monde de la téléphonie mobile. Son architecture est proche d'un système GNU/Linux même s'il existe des différences notables. Si l'on se place du coté applicatif, sa simplicité d'utilisation a bien entendu tenter les industriels car le coût de développement d'une application graphique Android est souvent moins élevé que dans le cas de GNU/Linux avec Qt. Dans un avenir proche, Android pourrait donc être la référence des systèmes « Linux embarqué » avec IHM. Dans cet article nous avons donc évalué dans l'environnement Android les différentes fonctionnalités utilisées sous GNU/Linux dans un cadre industriel, en particulier l'approche temps réel en utilisant les patch PREEMPT-RT et Xenomai. La majorité des tests a été réalisé sur x86 (Atom) avec Android-x86. [35]

III.1.11 Compétences pour la conception et développement de Système embarqué

Domaine métier: médical, loisirs, transport...

Ingénierie: électricité, électronique, chimie, mécanique, robotique, ..., informatique

Sciences: mathématiques, statistiques, probabilités, recherche opérationnelle

Informatique

- Algorithmique
- Programmation
- Architectures (CPU, mémoires, périphériques)
- Gestion de l'énergie des processeurs et périphériques
- Génie logiciel, co-design, méthodes formelles, automates...
- Evaluation de performances, test, simulation, vérification
- Systèmes d'exploitation (ordonnancement...)
- Sécurité et robustesse
- Réseaux, mobilité
- Capteurs et actionneurs
- Vision par ordinateur, caméra

III.1.12 Conclusion :

Les systèmes embarqués font aujourd'hui partie intégrante de notre vie et tendent à se généraliser à tous les domaines où la miniaturisation, la mobilité, la puissance de calcul sont nécessaires.

Les systèmes embarqués ne sont pas des systèmes ordinaires: Ils requièrent une fiabilité irréprochable du fait du coût élevé de leur fabrication et de leur production de masse.

Dans cette partie nous avons définis le système embarqué et son architecture, comme nous avons vu la diversité des systèmes embarqués, ce qui implique une diversité de contraintes et d'exigences, ainsi nous avons présenté un aperçu de certaines de ces contraintes.

PARTIE II : ANDROID

III.2.1 Historique d'Android [31]

La naissance d'Android : tout a débuté avec une société américaine du nom d'Android fondée en 2003 qui a été ensuite rachetée par Google deux ans plus tard (en 2005).

L'objectif premier était de développer un système d'exploitation qui permettrait à l'utilisateur d'interagir avec ce dernier. Car auparavant chaque constructeur développait son propre système embarqué. Dès lors, il était impossible de concevoir une application compatible sur tous les appareils, sans même parler des bibliothèques de développement fournies qui s'en trouvaient bridées afin que les secrets de fabrication des marques ne soient pas divulgués.

Au premier mois de l'année 2007, la marque à la pomme a présenté une véritable révolution : l'iPhone. C'est là que tout a basculé. Le système iOS se voulait moderne, bien en avance sur la technologie actuelle et l'annonce faite par Apple a été une grande claque pour les concurrents.

Pour la première fois, elles se sont mises d'accord, et de cet accord est né l'**Open Handset Alliance**, au mois de novembre de la même année. Concrètement, elle regroupait pas moins de 35 entreprises dont Google qui avaient suggéré le développement d'un système d'exploitation open-source, pour révolutionner le marché du mobile en proposant quelque chose de nouveau, et balayer la concurrence (Windows Mobile à l'époque et iOS surtout).

À l'heure actuelle, le projet Android est un **grand succès** avec près de 85% de parts de marché sur le secteur des nouveaux smartphones vendus (premier trimestre 20

III.2.2 Listes des versions android [32]

Le système de Google n'aurait pas connu un tel succès s'il était resté le même. C'est là que l'on voit la puissance d'un tel OS qui a su s'adapter aux besoins des utilisateurs à chaque version majeure et qui s'enrichit de nouveautés.

Voici les différentes versions que se sont succédées et un bref aperçu de chacune d'elles :

Version	Nom de la version	Caractéristiques majeures ajoutées	Date de sortie	API Level
Android 1.0	Apple pie	Téléchargement et mise à jour des applications via Android Market Navigateur supportant les sites web en HTML et XHTML Support de l'appareil photo Support des dossiers d'applicatons Accès aux serveurs e-mail POP3, IMAP4 et SMTP	23 sept. 2008	1
Android 1.1	Banana bread	"Afficher" & "Cacher" le pavé numérique, inclus dans le menu d'appel Support pour sauvegarder les fichiers attachés aux MMS	9 févr. 2009	2
Android 1.5	Cupcake	Support pour les claviers virtuels en trois parties avec prédiction des mots et dictionnaire personnalisé Support pour les Widgets, qui permettent d'accéder rapidement à certaines informations de l'application à laquelle ils sont rattachés Enregistrement vidéo dans les formats MPEG-4 et 3GP	30 avr. 2009	3
Android 1.6	Donut	Possibilité pour les développeurs d'intégrer leurs contenus dans les résultats de recherche Interface de l'Android Market améliorée Interface native pour l'appareil photo, la camera et la galerie Galerie : autorise les utilisateurs à sélectionner plusieurs photos pour suppression Support des écrans avec une résolution WVGA	15 sept. 2009	4
Android 2.0	Eclair	Performance matériel optimisée Support de plus de taille d'écran et résolutions Réorganisation de l'interface utilisateur Nouvelle interface du navigateur et support de l'HTML5 Nouvelle liste de contact Meilleur contraste pour les arrière-plan Amélioration de Google Maps 3.1.2	26 oct. 2009	5
Android 2.2	Froyo	Performance matériel optimisée Support de plus de taille d'écran et résolutions Réorganisation de l'interface utilisateur Nouvelle interface du navigateur et support de l'HTML5	20 mai 2010	8

		Nouvelle liste de contact Meilleur contraste pour les arrière-plans Amélioration de Google Maps 3.1.2		
Android 2.3	Gingerbread	Interface utilisateur mise à jour Support des grands écrans à résolutions extra-larges (WXGA et plus) Support de la VoIP et SIP Support des formats vidéo WebM/VP8, et l'encodage audio AAC	6 déc. 2010	9
Android 3.0	Honeycomb	Meilleur support des tablettes Bureau tridimensionnel avec widgets améliorés Ajout Google Talk video chat Ajout Google Livre Ajout "Navigation Privée"	22 févr. 2011	11
Android 4.0	IceCream Sandwich		18 oct. 2011	14
Android 4.1	Jelly Bean	Assistant Vocale Amélioration de la vitesse d'exécution Amélioration gestion appareil photo Accessibilité : mode gestuel, prise en charge clavier externe "Braille".	9 juil. 2012	16
Android 4.4	KitKat	Nouvelle interface translucide Enregistrement séquence vidéo de l'écran Amélioration du système de notification Gestion système des sous-titres Amélioration des performances	31 oct. 2013	19
Android 5.0	Lollipop	Nouvelle interface / design ("Material design") Amélioration de la rapidité Amélioration de la gestion de la batterie	17 oct. 2014	21
Android 6	Marshmallow	Support de l'USB Type-C Support de l'authentification par empreinte digitale Amélioration de la durée de la batterie avec un mode "deepsleep" Panneau pour controler les permissions des applications Android Pay Améliorations de Google Now	5 oct. 2015	23
Android 7.0	Nougat	Unicode 9.0 emoji Meilleur support du multitâche Multi-fenêtrage (PIP...)	22 août 2016	24

		Mises à jour systèmes amélioré (grâce à une double partition système) **Amélioration des performances et de la taille du code grâce à un nouveau compilateur JIT**		

Tab.III.3 Listes des versions Android

Tous les mois, Google publie de nouvelles statistiques qui rendent compte de la répartition des différentes versions de son système d'exploitation d'Android.

Mis à jour le 3 avril, le tableau révèle une évolution notable dans la distribution des différentes versions du système d'exploitation, en particulier la nouvelle branche d'Android, baptisée **Nougat** (version 7.x du système d'exploitation). Celle-ci frôle la barre des 5 % du fait d'une forte progression de sa branche 7.0, dont la part de marché à bien augmenté sur le mois (+2,1 points), tandis que celle numérotée 7,1 est restée stable.

Pour sa part, **Lollipop** se trouve désormais sur un peu moins un tiers des terminaux en circulation (32 %) : la version 5.0, est en léger recul, avec une baisse de 9,4 à 9 %, tandis que la branche 5.1, qui figure à elle seule sur plus d'un cinquième des smartphones détectés sur Google Play, est restée quasiment stable, à 23 %, ne permettant que 0,1 point.

Ces deux branches permettent aujourd'hui à Lollipop d'être la version Android la plus répandue auprès des utilisateurs, mais sa place est très proche d'une autre branche de l'OS, **Marshmallow**.

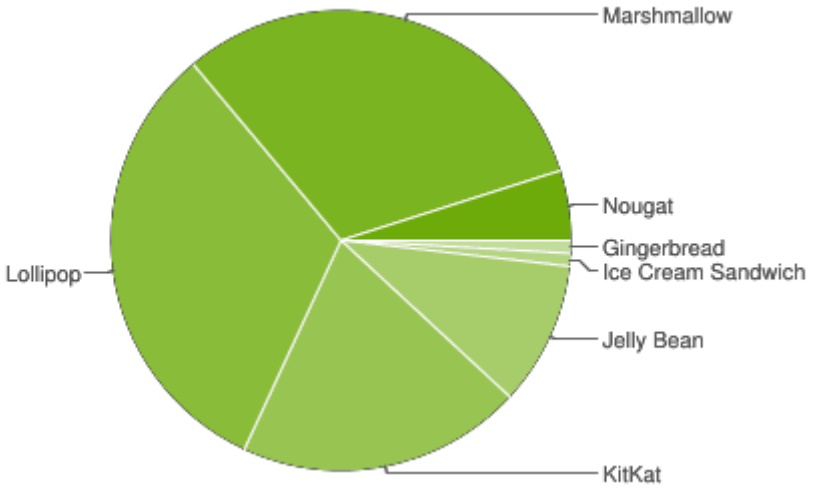


Figure 3.2 : Situation début avril 2017. [32]

Cela dit, cette mise à jour du système d'exploitation vient peut-être de toucher les limites de son développement. Alors qu'elle ne cessait de progresser mois après mois, se taillant une part de marché de 31,3 % début mars, elle vient pour la première fois

de reculer, certes de 0,1 point, mais la réalité est là : la progression constante de Marshmallow vient de prendre fin.

Concernant **KitKat**, dont la présence figure désormais sur un téléphone sur quatre (20 %), le mois de mars est un mois de baisse avec un recul de 0,8 point. Du côté de **Jelly Bean**, le recul continue aussi pour ses trois versions (4.1.x ; 4.2.x et 4.3) sont passées à 10,1 % début avril contre 10,6 % en mars. Elle reste toutefois la quatrième branche d'Android la plus courante.

Quant aux autres moutures du système d'exploitation, elles se contentent de quelques miettes : **Gingerbread** se situe à 0,9 %, avec une légère baisse tout au long de mars. C'est aussi à ce niveau que se trouve **IceCream Sandwich**, puisque sa part de marché a aussi perdu 0,1 point, à 0,9 %

III.2.3 Architecture d'android [33] :

Android est basé sur un kernel linux 2.6.xx, au-dessus du kernel il y a "le hardware abstraction layer" qui permet de séparer la plateforme logique du matériel. Au- dessus de cette couche d'abstraction on retrouve les librairies C/C++ utilisées par un certain nombre de composants du système Android. Au-dessus des librairies on retrouve l'Android Runtime, cette couche contient les librairies cœurs du Framework ainsi que la machine virtuelle exécutant les applications. Au-dessus la couche "Android Runtime" et des librairies cœurs on retrouve le Framework permettant au développeur de créer des applications. Enfin au-dessus du Framework il y a les applications.

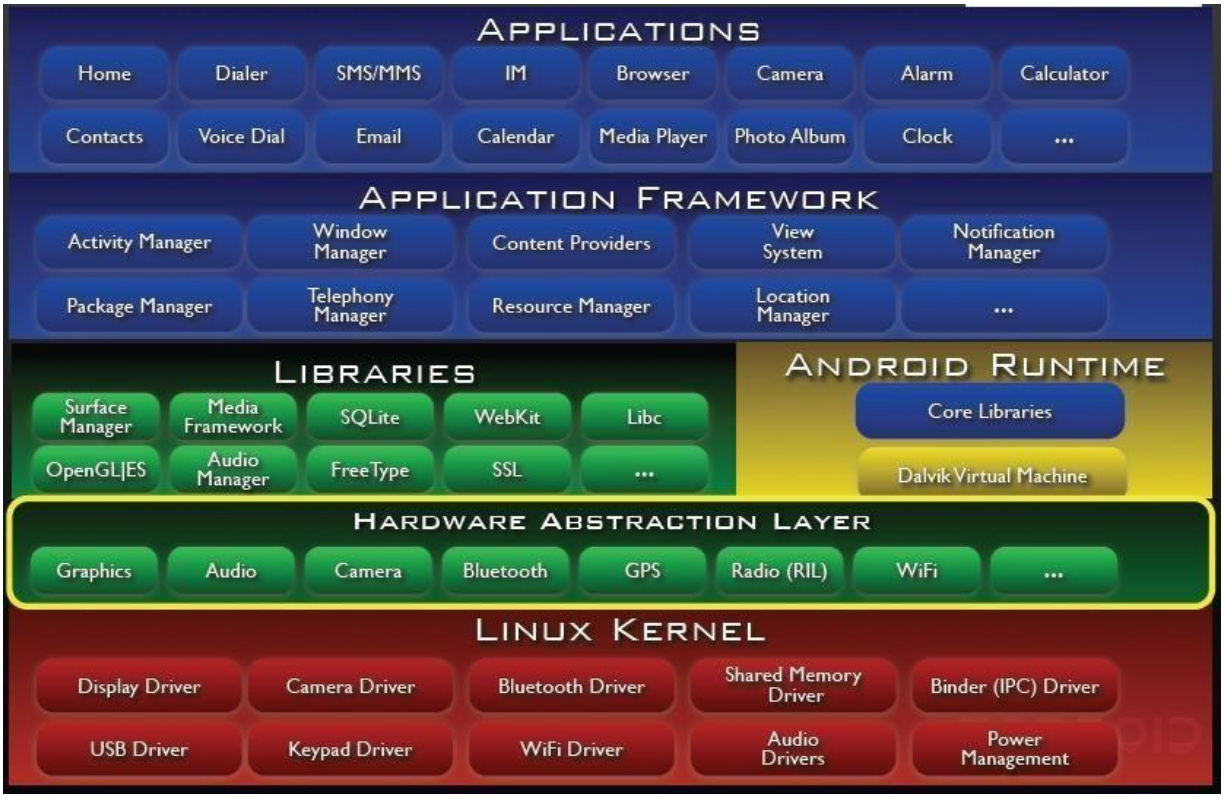


Figure 3.3 : Architecture d'android [33]

III.2.3.1 Le noyau Linux

Android est basé sur un kernel linux 2.6 mais ce n'est pas linux. Il ne possède pas de système de fenêtrage natif (X window system), la glibc n'est pas supporté, Android utilise une libc customisé appelé Bioniclibc.

Figure 3.3.1: Le noyau Linux

Android utilise un kernel avec différents patches pour la gestion de l'alimentation, le partage mémoire, etc. permettant une meilleurs gestion de ces caractéristiques pour les appareils mobiles. Android n'est pas linux mais il est basé sur un kernel linux. Pourquoi sur un kernel linux ?

- Le kernel linux a un système de gestion mémoire et de processus reconnu pour sa stabilité et ses performances.

- Le model de sécurité utilisé par linux, basé sur un système de permission, connu pour être robuste et performant. Il n'a pas changé depuis les années 70
- Le kernel linux fournit un système de driver permettant une abstraction avec le matériel. Il permet également le partage de librairies entre différent processus, le chargement et le déchargement de modules à chaud.
- le kernel linux est entièrement open source et il y a une communauté de développeurs qui l'améliorèrent et rajoute des drivers.

III.2.3.2 Librairies

Au-dessus du kernel, il y a les librairies natives. Ces librairies sont écrites en C/C++. Elles fournissent les fonctionnalités de bas niveau d'Android.

Figure 3.3.2 : Librairies

A) Bioniclibc :

Comme cela a été dit précédemment, Android ne supporte pas la glibc, donc les ingénieurs d'Android ont développé une librairie C (libc) nommé Bioniclibc. Elle est optimisée pour les appareils mobiles et a été développé spécialement pour Android.

Les ingénieurs d'Android ont décidé de développer une libc propre à la plateforme Android car ils avaient besoin d'une libc légère (la libc sera chargé dans chaque processus) et rapide (les appareils mobiles ne disposent de CPU puissant). La Bioniclibc a été écrit pour supporter les CPU ARM, bien que le support x86 soit présent. Il n'y pas de support pour les autres architecture CPU tel que PowerPC ou MIPS. Néanmoins, pour le marché des appareils mobiles, seulement l'architecture ARM est importante. Cette libc est sous licence BSD, elle reprend une grande partie du code des glibc issue d'OpenBSD, FreeBSD et NetBSD.

➢ **Caractéristique importante :**

- Elle pèse environ 200Ko soit la moitié de la glibc
- L'implémentation des pthreads (POSIX thread) a été complètement réécrit pour supporter les threads de la machine virtuelle Dalvik. De ce fait la Bioniclibc ne supporte les threads POSIX

- Les exceptions C++ et les "wide char" ne sont pas supportés
- Il n'y a pas de "Standard Template Library" (STI)

B) WebKit

Le navigateur web présent dans Android est basé sur le moteur de rendu sous licence BSD WebKit. WebKit est moteur de rendu, qui fournit une "fondation" sur lequel on peut développer un navigateur web. Il a été originellement dérivé par Apple du moteur de rendu KHTML pour être utilisé par le navigateur web Safari et maintenant il est développé par KDE project, Apple, Nokia, Google et d'autres. WebKit est composé de deux librairies : WebCore et JavascriptCore qui sont disponible sous licence GPL. WebKit supporte le CSS, Javascript, DOM, AJAX. La dernière version a obtenu 100% au test Acid 3. La version de WebKit présent dans Android a été légèrement modifiée pour s'adapter aux appareils mobiles. Ainsi le moteur de rendu basé sur WebKit présent dans Android supporte l'affichage sur une colonne.

C) Media Framework

La librairie Media Framework est basée sur OpenCore de PacketVideo. Elle permet le support des standards audio/vidéo/images. Le schemaci dessousdecrit toutes les fonctionnalitées fournit par cette librarie :

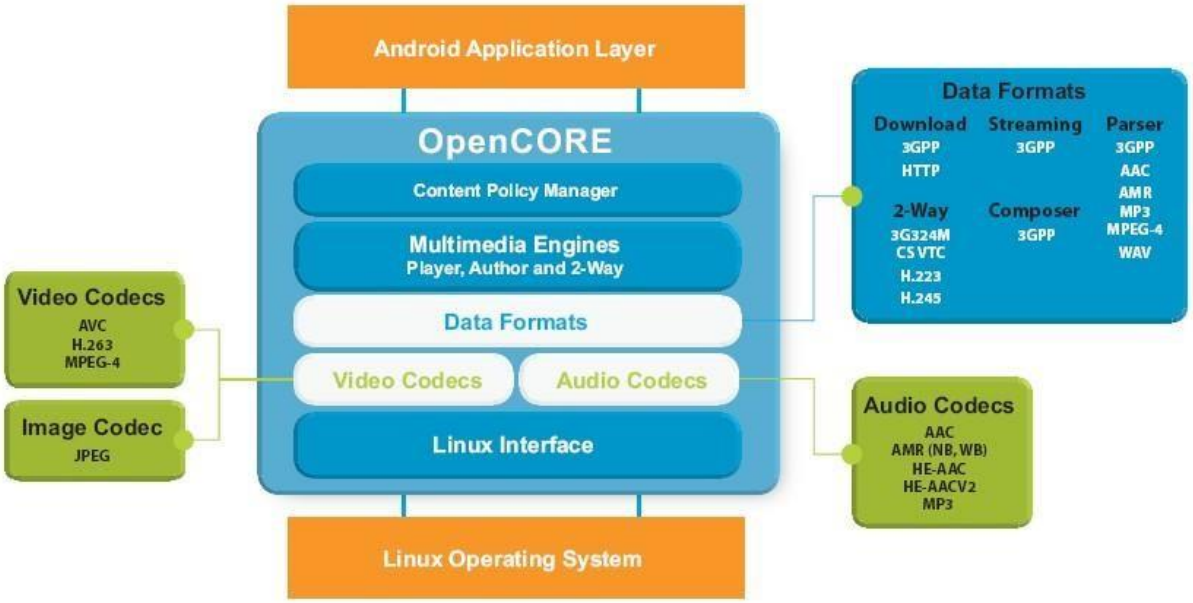


Figure 3.3.2.1: Media Framework [33]

III.2.3.3 Hardware Abstraction Layer

Figure 3.3.3 : Hardware Abstraction Layer

Cette couche se situe entre les librairies et le kernel linux, elle fournit les interfaces que doivent implémenter les drivers kernel. Cette couche sépare la plateforme logique des interfaces matérielles. Le but de cette couche est de faciliter le portage des librairies sur différents matériels.
Les ingénieurs d'Android ont décidé de faire cette couche car :

- pas tous les drivers kernel n'ont des interfaces standardisées.
- les drivers kernel sont sous licence GPL ce qui exposerait les interfaces propriétaires des fabricants. Les fabricants veulent pouvoir garder ces interfaces en "closed source"
- Android a des besoins spécifiques pour les drivers kernel.

III.2.3.4 Android Runtime :

Cette couche se situe au-dessus des libraires C/C++, elle se compose du "cœur" du Framework et de la machine virtuel dalvik.

Figure 3.3.4 : III.4 Android Runtime

III.2.3.4.1 Dalvik

La machine virtuelle Dalvik est basée sur une architecture de registre à l'instar de beaucoup de machine virtuel et de la machine virtuel Java qui ont une architecture de pile. Utilisé une architecture de pile ou de registre dépends des stratégies de compilation et d'interprétation choisit. Généralement, les machines basées sur une architecture de pile, doivent utiliser des instructions pour charger les données sur la pile et manipuler ces données. Ce qui rajoute des instructions dans le code machine, et donc il y a plus de code que pour une machine basé sur une architecture de registre. Cependant, les instructions pour une machine basée sur une architecture de registres doivent être encodées pour les registres sources et destinations, ce qui prend également de la place dans le code machine résultant. La différence est essentiellement importante suivant l'interpréteur de code machine présent dans la VM.

Les applications Java développées pour Android doivent être compilées au format dalvik exécutable (.dex) avec l'outil dx. Cet outil compile les .java en .class et ensuite il convertit ces .class en .dex. Un .dex peut contenir plusieurs classes. Les strings dupliqués et autre constantes utilisées dans de multiples classes sont regroupées dans un .dex. Le bytecode utilisé dans les .dex est le Dalvikbytecode et non le java Bytecode.

Pour comparaison un .dex décompressé est un peu plus petit en taille qu'un .jar compressé dérivé des mêmes fichiers .class.

Etant optimisé pour utiliser une quantité de mémoire minimale, la VM Dalvik a quelques caractéristiques spécifiques par rapport aux autres VM:

- la VM a été "dégraissée" pour utiliser moins d'espace mémoire
- pas compilation à la volé (JIT)
- Elle utilise son propre bytecode et pas le Java bytecode
- La table des constantes a été modifié pour n'utiliser que des indexes de 32 bit afin de simplifier l'interpréteur.

III.2.3.4.2 librariesCore

Les librariesCore fournissent le langage Java disponible pour les applications. Le langage Java fournit avec Android reprend en grande partie l'API JSE 1.5. Il y a des choses qui ont été mis de côté car cela n'avait pas de sens pour Android (comme les imprimantes, swing, etc.) et d'autres par ce que des APIs spécifiques sont requises pour Android.

Packages JSE 1.5 supportés par Android :	Packages JSE 1.5 non supportés par Android :	Librairies spécifiques ajoutées dans les Core Libraries d'Android :
java.io java.lang (sauf java.lang.management) support java.math	java.applet java.awt java.beans java.lang.management java.rmi	org.apache.commons.codec org.apache.commons.httpclient org.bluez org.json

java.net java.nio java.security java.sql java.text java.util javax.crypto javax.net javax.security (sauf javax.security.auth.kerberos, javax.security.auth.spi, and javax.security.sasl) javax.sound javax.sql (sauf javax.sql.rowset) javax.xml.parsers org.w3c.dom org.xml.sax	javax.accessibility javax.activity javax.imageio javax.managemen t javax.naming javax.print javax.rmi javax.security.auth.kerbe ros javax.security.auth.spi javax.security.sasl javax.swing javax.transaction javax.xml (sauf javax.xml.parsers) org.ietf.* org.omg.* org.w3c.dom. *	

Tab.III.3 : Récapitulatif des différentes librairies Java, rajoutées ou non supportées par Android

III.2.3.5 La couche Framework

Le Framework est situé au-dessus de l'Android Runtime et des librairies. Il fournit des API permettant aux développeurs de créer des applications riches.

Figure 3.3.5 : La couche Framework

III.2.3.5.1 CorePlateform Services :

Android introduit la notion de services. Un service est une application qui n'a aucune interaction avec l'utilisateur et qui tourne en arrière-plan pendant un temps indéfini. Les services cœurs de la plateforme (CorePlateform Services) fournissent des services essentiels au fonctionnement de la plateforme :

- **Activity Manager** : gère le cycle de vie des applications et maintient une "pile de navigation" (navigation backstack) permettant d'aller d'une application à une autre et de revenir à la précédente quand la dernière application ouverte est fermée.

- **Package Manager** : utiliser par l'Activity Manager pour charger les informations provenant des fichiers .apk (androidpackaga file)
- **Window Manager** : juste au-dessus du Surface Flinger (lien), il gère les fenêtres des applications --> quelle fenêtre doit être affiché devant une autre à l'écran.
- **Resouce Manage** : gère tous ce qui n'est pas du code, toutes les ressources --> images, fichier audio, etc.
- **Content Provider** : gère le partage de données entre applications, comme par exemple la base de données de contact, qui peut être consultée par d'autres applications que l'application Contact. Les Données peuvent partager à travers une base de données (SQLite), des fichiers, le réseau, etc.
- **View System** : fournit tous les composants graphiques : listes, grille, text box, buttons et même un navigateur web embarqué.

III.2.3.5.2 Hardware Services (Les services matériels)

Les services matériels (Hardware Services) founissent un accès vers les API matérielles de bas niveau :

- **Telephony Service** : permet d'accéder aux interfaces "téléphonique" (gsm, 3G, etc.)
- **Location Service** : permet d'accéder au GPS.
- **Bluetooth Service** : permet d'accéder à l'interface bluetooth.
- **WiFi Service** : permet d'accéder à l'interface Wifi.
- **USB Service** : permet d'accéder aux interfaces USB.
- **Sensor Service** : permet d'accéder aux détecteurs (détecteurs de luminosité, etc.)

III.2.3.6 La couche Application

Android est fourni avec un ensemble de programmes de base (également nommés applications natives) permettant d'accéder à des fonctionnalités comme les courriels, les SMS, le téléphone, le calendrier, les photos, les cartes géographiques, le Web, pour ne citer que quelques exemples. Ces applications sont développées à l'aide du langage

de programmation Java. Pour l'utilisateur final, c'est la seule couche accessible et visible [9]

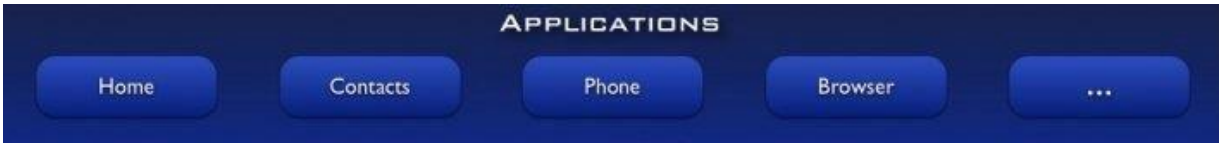


Figure 3.3.6 : Couche Application

III.2.4 Composantes d'une application Android

Une application Android est composée d'éléments de base :

III.2.4.1 Activities (Activités en Français)

Une activité est la composante principale pour une application Android. Elle représente l'implémentation et les interactions de vos interfaces.

Prenant l'exemple d'une application qui liste toutes les images présentes dans votre téléphone, le projet pourrait se décomposer comme çi-dessous :

- Une vue pour afficher la liste des images.
- Une activité pour gérer le remplissage et l'affichage de la liste.
- Si l'on veut pouvoir rajouter, supprimer des images, on pourrait rajouter d'autres activités.

III.2.4.2 Services

Un service, à la différence d'une activité, ne possède pas de vue mais permet l'exécution d'un algorithme sur un temps indéfini. Il ne s'arrêtera que lorsque la tâche est finie ou que son exécution est arrêtée.

Il peut être lancé à différents moments :

- Au démarrage du téléphone.
- Au moment d'un événement (arrivée d'un appel, SMS, mail, etc...).
- Lancement de votre application.
- Action particulière dans votre application.

III.2.4.3 Broadcast and IntentReceivers :

Un BroadcastReceiver comme son nom l'indique permet d'écouter ce qui se passe sur le système ou sur votre application et déclencher une action que vous aurez prédéfinie. C'est souvent par ce mécanisme que les services sont lancés.

III.2.4.4 Content providers

Les "content providers"servent à accéder à des données depuis votre application. Vous pouvez accéder :

- Aux contacts stockés dans le téléphone.
- A l'agenda.
- Aux photos.
- Ainsi que d'autres données depuis votre application grâce aux content providers.

III.2.4.5 Le cycle de vie d'une activité

Le cycle de vie d'une activité correspond aux différents états d'une activité lors de sa gestion par le système Android. Il est très important car il va vous permettre de suivre l'état de votre activité au fur et mesure de son existence dans le système Android.

Le voici, il fait un peu peur comme ça, mais on le commente ensemble ensuite !

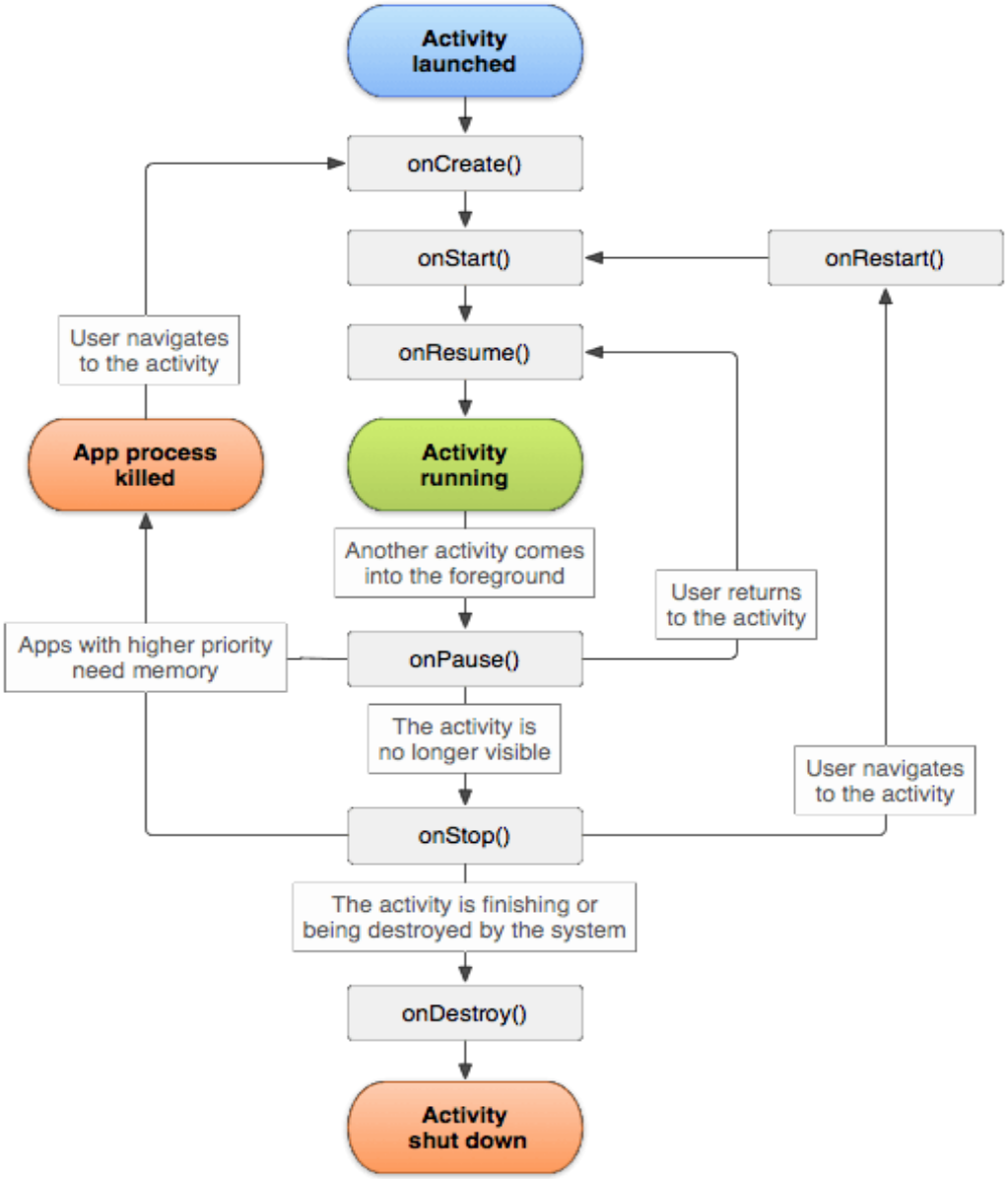


Figure 3.4 : Le cycle de vie d'une activité

- onCreate() :
  - Exécuté quand l'utilisateur clique sur l'icône de l'application pour une première fois
  - Utilisé pour l'initialisation :
  - vue XML;
  - des fichiers/données temporaires;
- onRestart() :
  - Exécuté lors l'activité arrêtée via un stop () redémarre (repasse en premier plan);

- onStart() :
  - Exécuté après chaque onCreate() ou onRestart();
  - Chargement des données sauvegardées durant le dernier arrêt;
- onResume() :
  - Exécuté après chaque onStart();
  - Exécuté a chaque passage en premier plan de l'activité;
  - Initialisation :
  - Connexion DB;
  - Mise à jour des données qui auraient pû être modifées entre temps (avant le onResume);
- onPause() :
  - Exécuté avant chaque onStop();
  - Exécuté chaque fois que l'utilisateur passe à une autre activité, ou bien lorsqu'il demande un « finish() » sur cette activité, ou bien encore si le système à besoin de libérer de la mémoire;
  - Libération des resources :
  - sauvegarde des données qui seront perdues après l'arrêt si elles ne sont pas sauvegardées;
  - connexion DB;
  - Attention : l'exécution de cette fonction doit être rapide car la prochaine activité ne démarrera pas tant que l'exécution de cette fonction n'est pas terminée;
- onStop() :
  - Exécuté avant chaque mise en sommeil;
  - Exécuté avant chaque onDestroy:
  - Libération des ressources;
- onDestroy() :
  - Exécuté lors du kill / de l'arrêt de l'activité;
  - onCreate() devra à nouveau être exécuté pour obtenir à nouveau l'activité.
  - libération des resources
  - fichiers temporaires

III.2.5 Les outils de développement Android indispensables

Afin de programmer pour android, plusieurs outils sont nécessaires, certains sont même fournis par Google.

III.2.5.1 Eclipse

Eclipse IDE est un environnement de développement intégré libre (le terme *Eclipse* désigne également le projet correspondant, lancé par IBM) extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions. La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plug-in (en conformité avec la norme OSGi) : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plug-in.

III.2.5.2 Le JDK Java Development Kit (JDK)

Le **Java Development Kit** (**JDK**) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en bytecode destiné à la machine virtuelle Java.

Il existe plusieurs éditions de JDK, selon la plate-forme Java[1] considérée (et bien évidemment la version de Java ciblée) :

- JSE pour la Java 2 Standard Edition également désignée J2SE ;
- JEE, sigle de Java Enterprise Edition également désignée J2EE ;
- JME 'Micro Edition', destinée au marché mobiles ;
- etc.

À chacune de ces plateformes correspond une base commune de Development Kits, plus des bibliothèques additionnelles spécifiques selon la plate-forme Java que le JDK cible, mais le terme de JDK est appliqué indistinctement à n'importe laquelle de ces plates-formes.

III.2.5.3 LE SDK Software Development Kit [34]

SDK signifie Software Development Kit, c'est **un ensemble d'outils d'aide à la programmation** pour concevoir des logiciels, jeux, applications mobiles, etc. pour un terminal et/ou un système d'exploitation spécifique.

En quelques mots, un SDK contient **du code**, permettant de **concevoir une interface ou une partie d'une interface numérique** (web, mobile, jeux, logiciels de recherches, widget météo…). Ce code est conçu avec le langage de programmation correspondant au terminal (ordinateur, téléphone, tablette…) et au système de navigation ciblés. Par exemple, pour développer une application mobile (logiciel) pour iPhone (terminal mobile avec système d'exploitation iOS), il faut utiliser le SDK iOS mobile.

Ce code est organisé sous forme de **librairies (ou "bibliothèques logicielles")** c'est-à-dire des collections de fonctions prédéfinies, de points d'accès à du matériel et à des fonctionnalités système (ou "natives") d'un terminal. Autrement dit, pour réaliser une application mobile pour iPhone par exemple, il faudra utiliser le SDK mobile iOS qui contient des fichiers de code entiers "tout prêts" qui permettent d'utiliser les fonctions natives du téléphone (GPS, bluetooth, appareil photo…) et qui proposent des éléments graphiques standards iOS.

Attention, ce code ne peut pas être dupliqué et intégré dans le code source. Le développeur doit mettre en place des « ponts » dans son code qui vont **communiquer avec le SDK**, pour **appeler** les fonctionnalités nécessaires au développement du logiciel.

En parallèle de ces librairies, de nombreuses ressources peuvent accompagner un SDK :

- documentation expliquant comment utiliser le SDK et éviter certaines erreurs, des conseils et suggestions pour produire un code solide et performant, etc.;
- un **émulateur pour faire des tests** en simulant un terminal;
- des **éléments graphiques** tels que des boutons ou une banque de pictogrammes ;
- etc.

Chaque SDK est différent et plus ou moins d'outils seront fournis avec lui.

Un SDK peut être gratuit ou payant, en téléchargement libre sur internet ou distribué seulement sur demande.

Conclusion :

Tout au long de cette partie, nous avons abordé quelques généralités sur Android, et nous avons introduit quelques outils de développement Android qui vont nous servir à réaliser notre application Android d'un crypto-système basé sur l'algorithme TEA.

IV.1. Introduction :

Dans ce chapitre nous allons présenter l'environnement du développement de notre application ainsi que les différents outils utilisés pour sa réalisation puis expliquer son fonctionnement en présentant quelques interfaces illustratives

IV.2 Environnement de travail :

Dans notre projet, l'environnement de travail disponible pour le développement de l'application est le suivant :

- Pc : Intel Core i3-4005U, 1.7 GHZ, 4,00 GO Mémoire vive, Windows 8.1

IV.2.1 Environnement logiciel :

Notre application est destinée pour tous les smartphones équipé d'un système Android 4.0.3 dite IceCream Sandwich et plus

IV.2.2 Environnement de travail :

Avant de pouvoir faire du développement, il faut nécessairement préparer l'environnement de travail. Dans notre cas, on a choisi de faire notre développement mobile sur la platforme Eclipse. Pour la préparer, on a effectué différentes tâches qui sont :

- Installation JDK
- Installation Eclipse
- Installation SDK Android
- Intégration SDK Android sous Eclipse pour pouvoir créer des projets Android.

IV.2.2.1 Langage de programmation :

Le langage java est un langage de programmation orienté objet mis au point par Sun Microsystème. Sa caractéristique principale est qu'il est indépendant de toute plateforme, il est possible d'exécuter des programme java surtout les environnements que possèdent une « Java Virtuelle Machine » (JVM).

Ce concept est à la base de slogan de Sun pour java : WORA (Write Once Run Anywhere : Ecrire une fois, exécuter partout).Sun fournit aussi gratuitement un ensemble d'outils et d'API pour Permettre de développer des programmes avec ce langage, Ce Kit est nommé JDK (Java Développement Kit).

Java est caractérisée aussi par la réutilisation de son code ainsi que la simplicité de sa mise en œuvre.

IV.2.2.2 Eclipse :

Eclipse est un IDE qui permet de programmer dans différents langages grâce a ces nombreux plug-ins et notamment les plug-ins d'Android. Une interface spécifique permet de gérer des fichiers java et de compiler ses programmes. Les fichiers sont organisés selon une arborescence qui correspond au paquetage java défini. L'analyse syntaxique permet de mettre en valeur les mots clés dans les fichiers java.

Eclipse dispose aussi d'un système d'auto complétion des fonctions, de détection des erreurs syntaxiques en temps réel sans oublier un système de débogage permettant d'exécuter ses programmes pas à pas.

Figure IV : LOGO Eclipse

IV.3 Les différentes interfaces de notre application :

IV.3.1 Lancement de l'application

En premier lieu l'utilisateur lance l'application en appuyant sur l'icône qui lui correspond dans le Menu du Smartphone

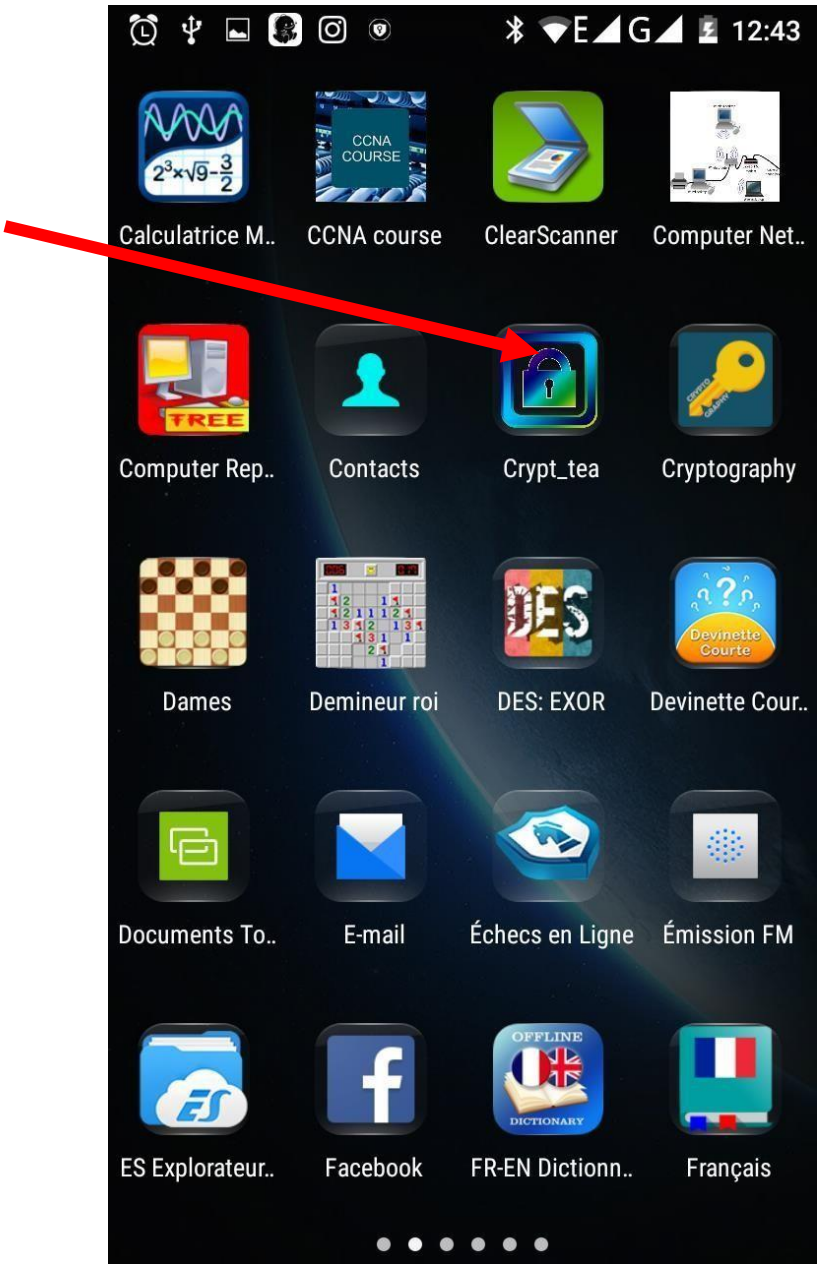


Figure IV: L'icône de l'application

IV.3.2 Le menu principal de l'application :

Apres l'appui sur l'icône, on aura cette interface suivant :

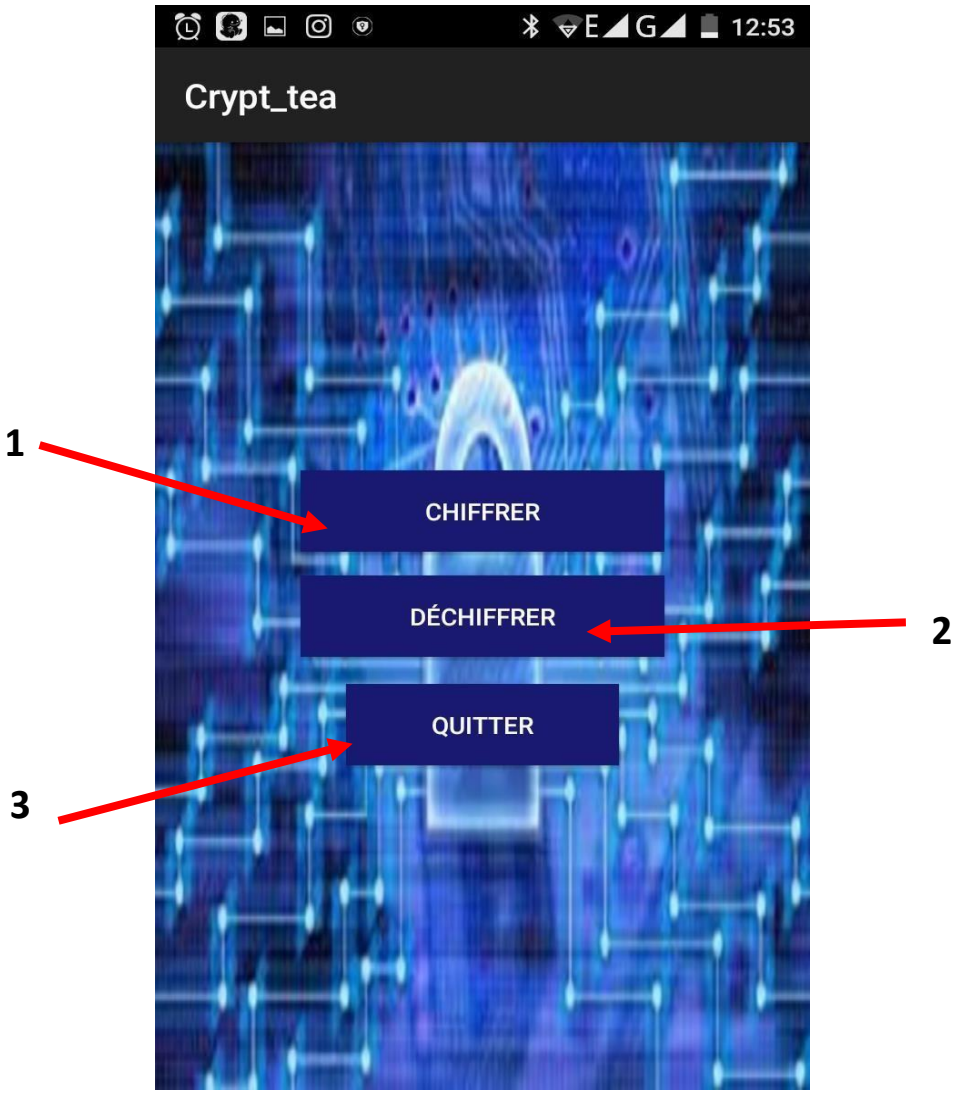


Figure IV : Menu principal de l'application

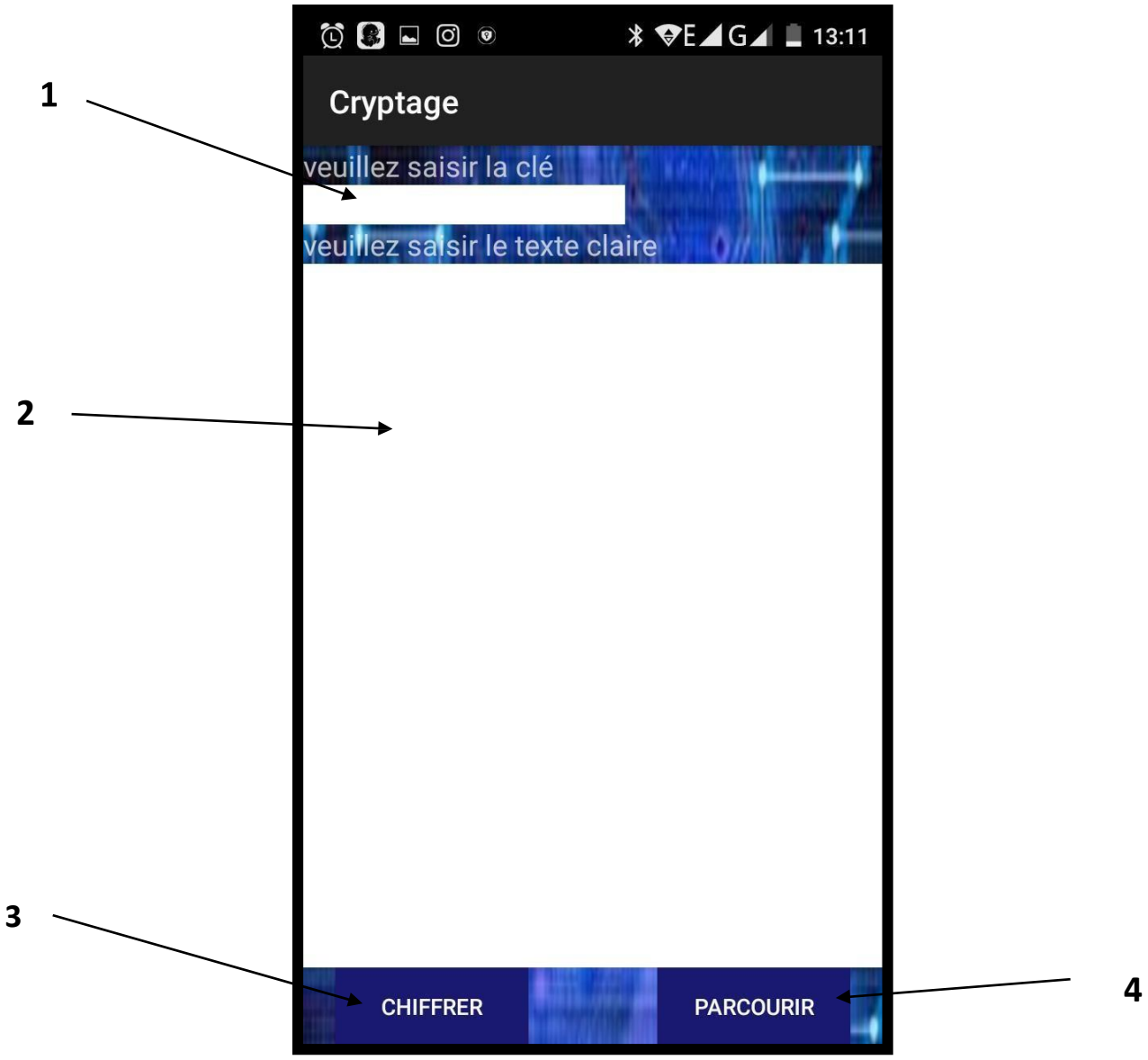
(1) Accéder à l'interface Cryptage

(2) Accéder à l'interface Décryptage

(3) Quitter l'application

IV.3.3 L'interface de cryptage :

Apres avoir cliquer sur le bouton chiffrement l'interface de cryptage apparaitra, elle demandera de saisir une clé de 16 caractères et de saisir le texte à chiffrer



(1) Saisir la clé

(2) saisir le texte en claire

(3) chiffrer le texte

(4) parcourir des fichiers

Pour crypter un fichier, il faut cliquer sur le bouton parcourir et le sélectionner

Après avoir cliqué sur le bouton parcourir l'interface suivante s'affichera.

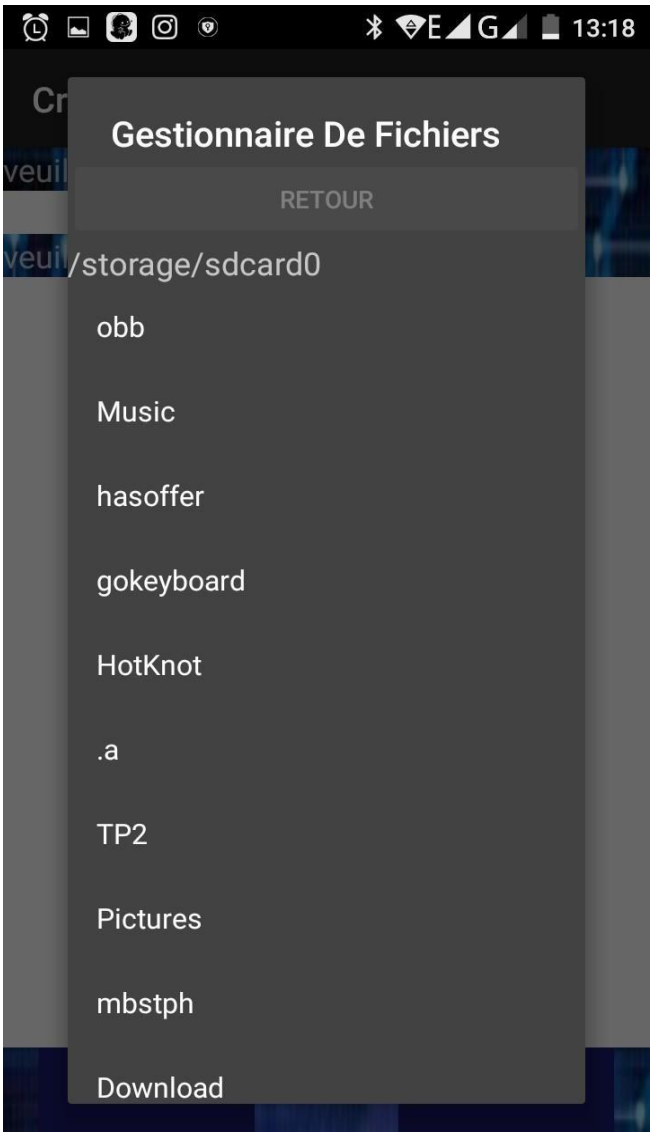


Figure IV : Gestionnaire de fichiers

Lorsqu'on trouve le fichier on clique dessus et le contenu du fichier sera exporter automatiquement vers l'interface Cryptage ensuite on peut le chiffrer

IV.3.4 L'interface Résultat :

Apres avoir cliqué sur chiffrer le texte crypté s'affichera dans l'interface Résultat

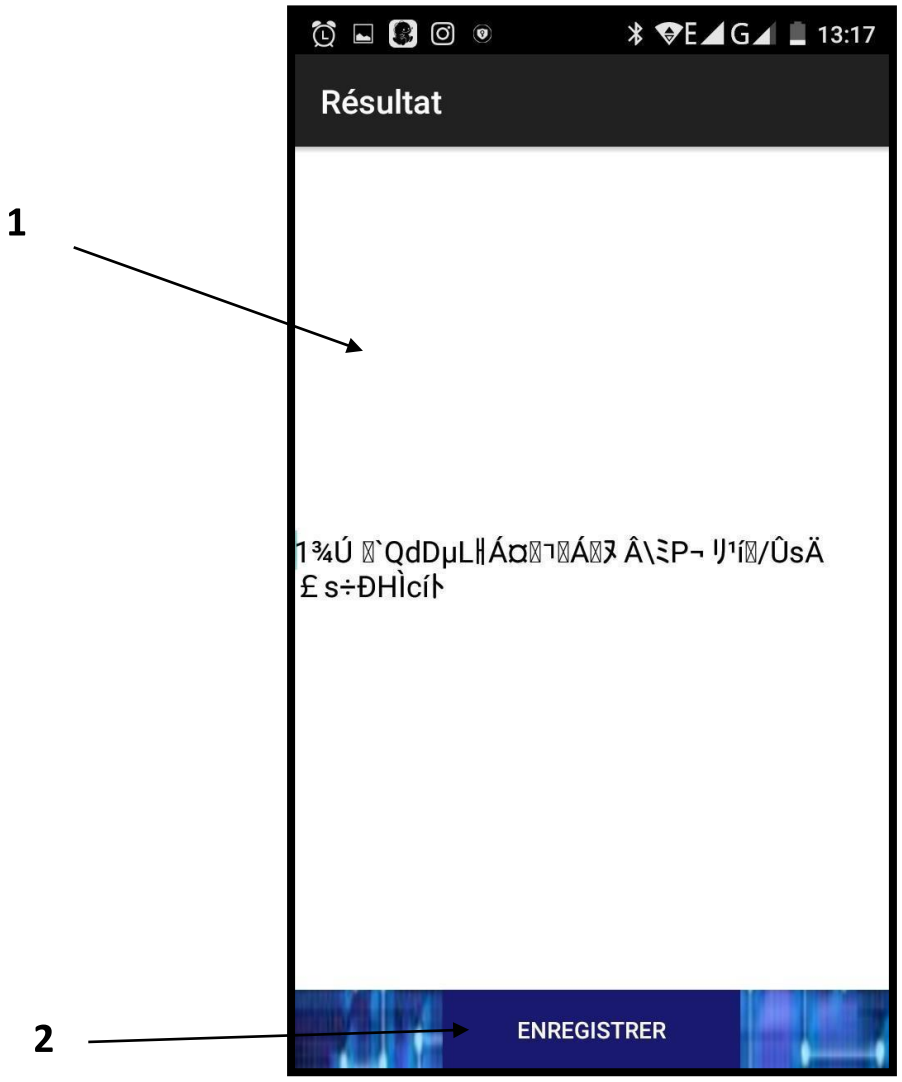


Figure IV : l'interface Résultat

(1) Affichage du texte crypté

(2) Enregistrer le texte dans un fichier txt

Une fois que le texte soit chiffré, on peut l'enregistrer en cliquant sur enregistrer.

Après avoir cliqué sur le bouton enregistrer l'interface de dialogue suivante s'affichera :

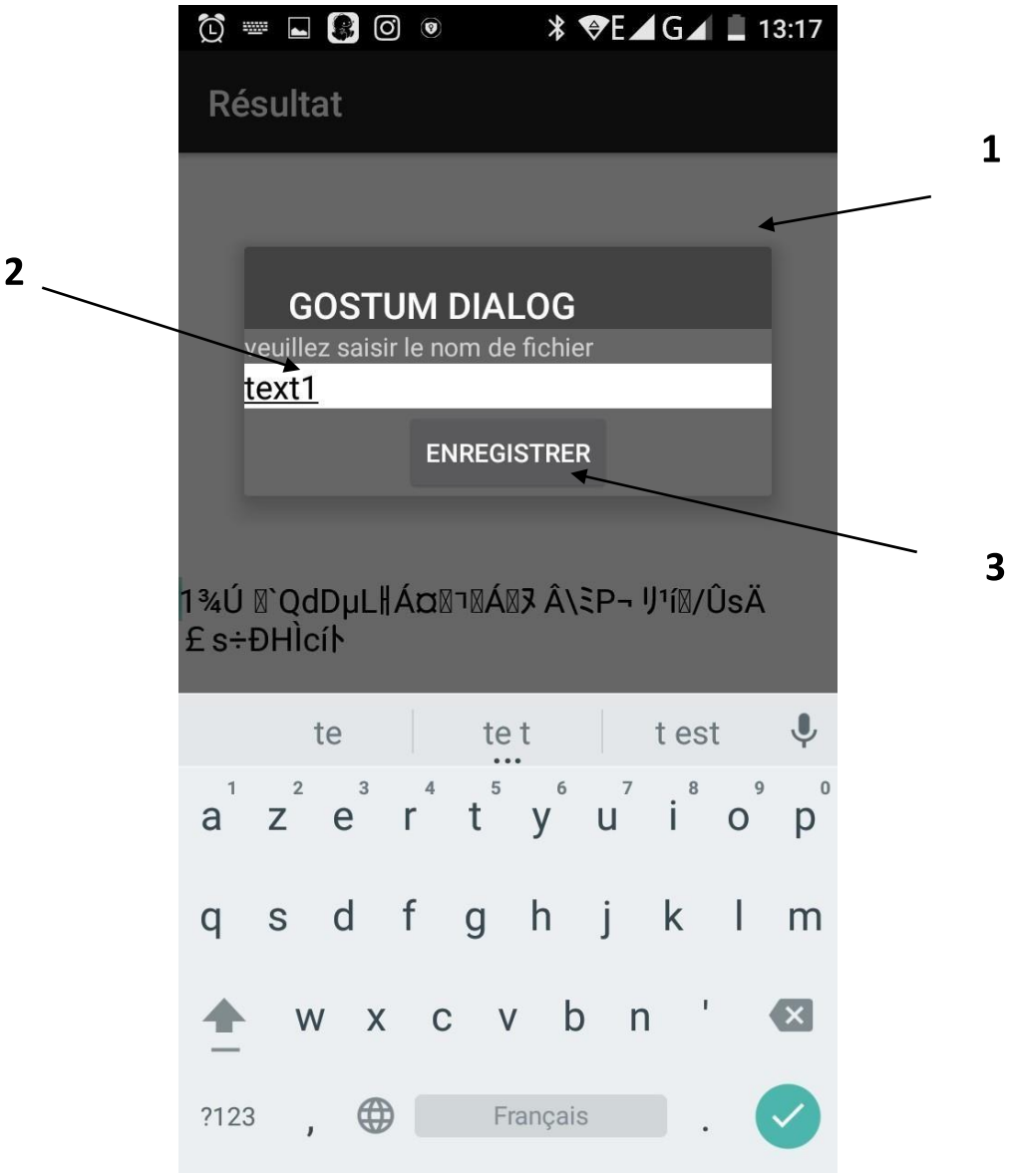


Figure IV : La boite Dialogue

(1) La boite dialogue

(2) Saisir le nom du fichier

(3) Enregistrer le fichier

IV.3.5 L'interface Décryptage :

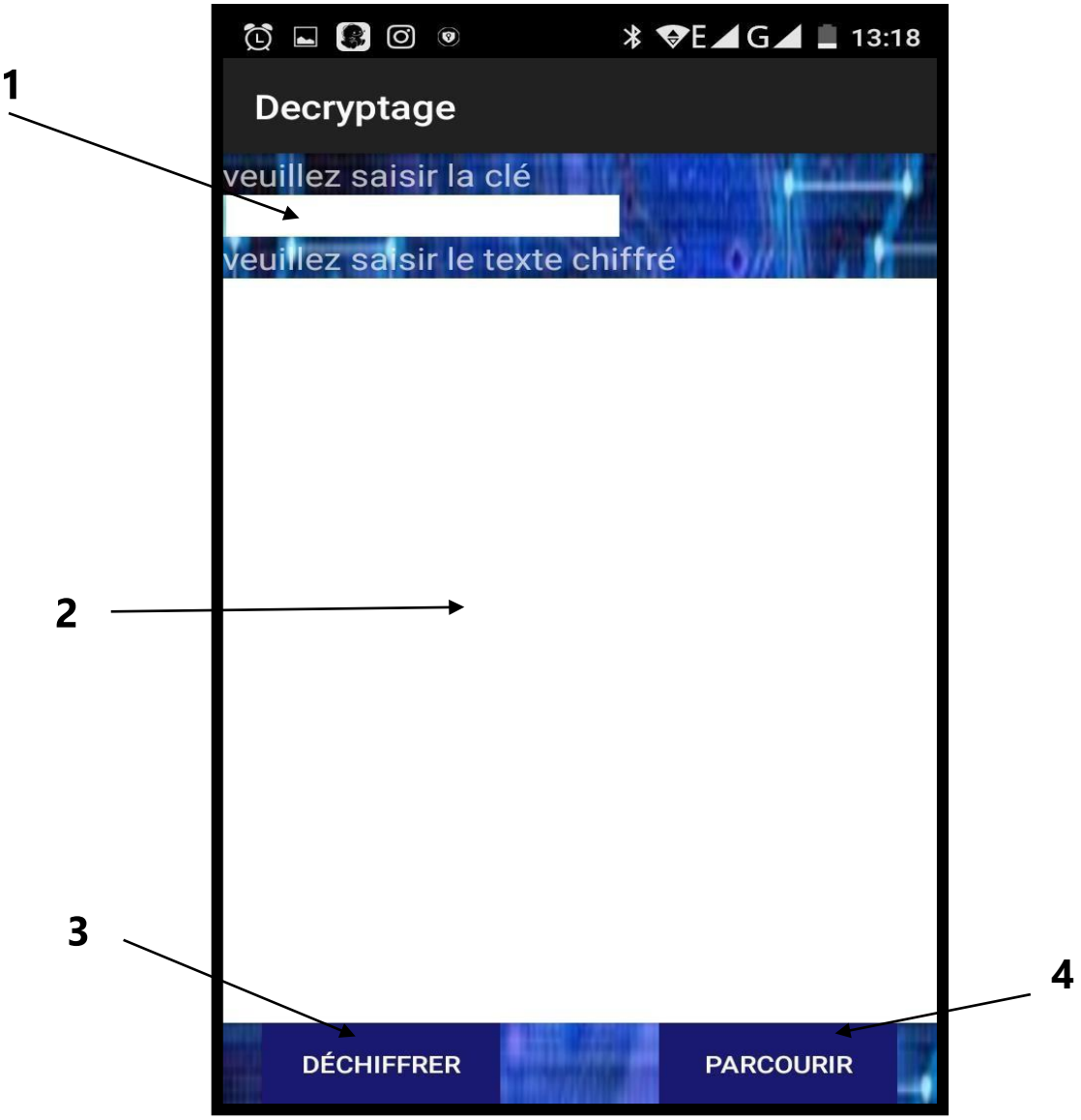


Figure IV : L'interface décryptage

(1) Saisir la clé (16 caractères obligatoire)
(2) saisir le texte en claire
(3) Déchiffrer le texte
(4) parcourir des fichiers

IV.4 Conclusion :

Ce quatrième chapitre a été consacré à la phase réalisation de notre application. Nous avons tout d'abord défini l'environnement de développement, puis illustré les différentes interfaces qui la composent.

Conclusion générale :

Nous avons au cours de ce mémoire, réalisé un crypto système basé sur l'algorithme de TEA (Tiny Encryption Algorithm), pour les systèmes embarqués cas : Android.

Pour cela nous avons, dans un premier temps commencé par présenter des généralités sur la sécurité dans les réseaux informatiques, ensuite nous avons vu quelques notions de base sur la cryptographie, ainsi nous avons détaillé le fonctionnement de l'algorithme TEA, et dans la partie suivante nous avons définis le système embarqué et son architecture, ensuite nous avons abordé quelques généralités sur android et son architecture, et nous avons introduit quelques outils de développement android qui vont nous servir à réaliser notre application.

Au cours de la réalisation de ce projet nous avons enrichis notre savoir et développé nous connaissances informatique notamment dans le domaine de la programmation et la cryptographie. En effet l'application a exigé des connaissances du langage JAVA et des outils de développement indispensables à sa réalisation.la mise en œuvre de notre travaille a exigé des connaissances très approfondies en la matière ainsi qu'une bonne maitrise de la configuration d'éclipse et de l'environnement Android.

Et enfin, notre travaille ainsi présenté reste un prototype sur lequel nous espérant apporter plus de connaissances pour le rendre meilleur et plus fonctionnel, ainsi nous aimerons améliorer cette application pour qu'elle puisse crypter tous types de donnés (images, Vidéos … etc.).

Bibliographie :

[1] : Elie MABO, La sécurité des systèmes informatiques (Théorie), support de cours, 2010

[2] : http://www.commentcamarche.net/contents/47-piratage-et-attaques-informatiques

[3] : Nicolas W. Vermeys, Virus informatiques : responsables et responsabilité 2008

[4]: Anderson, J. Computer security threat monitoring and surveillance. 1980

[5] : Anlauf, J.K. et Biehl, M. The adatron : an adaptive perceptron algorithm. *Europhysics Letters*. Vol. 10, pp. 687-692, Décembre 1989

[.6] :http://aidesecurite.blogspot.com/2013/03/types-dattaques-dun-reseau.html

[7] :https://tpesecuriteinformatique.wordpress.com/les-differentes-attaques-informatiques/

[8] :http://moodle.utc.fr/file.php/498/SupportIntroSecu/co/CoursSecurite_8.html

[9] :http://www.awt.be/web/sec/index.aspx?page=sec,fr,100,010,001

[10] :http://www.helpcomputer.ch/vpn.html

[11] : Vincent Erceau & Romain Colombier, « GMSI Informatique », Projet SAS, 2011.

[12] :http://www2.cegep-ste-foy.qc.ca/freesite/index.php?id=14735

[13] : David Bugermeister, Jonathan Krier « systéme de détection d'intrusion »

[14] : Karim Tamine, « Sécurité dans les réseaux », Cours Master 2 (recherche)-Informatique –Decembre 2004

[15] :www.securiteinfo.com,(Le Grand Livre de SecuriteInfo.com),19 février 2004

[16] Sébastien Varrette, « Cryptographie: Principe et évolution au cours des âges », Cours de Cryptographie CUT I3-011, Université du Luxembourg, septembre 2004.

[17]http://publimath.irem.univ-mrs.fr/glossaire/CR011.htm

[18] : Rabah LEBSIR, « Evaluation des performances de l'algorithme AES dans les systèmes embarqués : Cas des Smartphones sous Android », Université Amar Télidji,2013

[19]http://www.academia.edu/11459156/Rapport_sur_létude_et_limplémentation_de_quelques_algorithmes_de_chi_rement_et_de_signature

[20]Cryptographie et Sécurité informatique ,Université de Liège Faculté des Sciences Appliquées 2009 - 2010 Renaud Dumont

[21] RAHMOUNI SAMIA, Développement d'une application pour l'échange des messages sécurisés, Université Abou Bakr Belkaid– Tlemcen ,2015

[22] :http://ram-0000.developpez.com/tutoriels/cryptographie/#L7.4

[23] ;http://www.bibmath.net/crypto/index.php?action=affiche&quoi=moderne/aes

[24] :http://moodle.utc.fr/file.php/498/SupportIntroSecu/co/CoursSecurite_32.html

[25] :https://fr.slideshare.net/foofiM/tiny-encryption-algorithm

[26] :http://study.com/academy/lesson/tiny-encryption-algorithm-tea.html

[27] : Simon Shepherd, Professor of Computational Mathematics Director of the Cryptography and Computer Security Laboratory, « The Tiny Encryption Algorithm » Bradford University, England.

[28] : Université paulsabatier Licence Professionnelle « Conception et Commande de Systèmes Electriques Embarqués

[29] :https://kadionik.vvv.enseirb-matmeca.fr/glmf/hs24/hs24_1.pdf

[30] : http://www.ordinateur.cc/Matériel/Durs-et-stockage/52946.html

[31]http://www.phonandroid.com/toute-l-histoire-et-la-chronologie-d-android-dossier.html

[32] http://socialcompare.com/fr/comparison/android-versions-comparison

[33]http://igm.univ-mlv.fr/~dr/XPOSE2008/android/archi_linux.html

[34]http://www.mobizel.com/2015/04/definition-cest-quoi-un-sdk/

[35] http://connect.ed-diamond.com/Open-Silicium/OS-008/Utilisation-d-Android-pour-des-systemes-embarques-et-temps-reel