République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Amar Télidji Laghouat

Faculté des Sciences
Département de Mathématique et Informatique

Mémoire en vue de l'obtention du diplôme de Magister en informatique

Thème :

Evaluation des performances de l'algorithme AES dans les systèmes embarqués : Cas des Smartphones sous Android

Présenté par :
Rabah LEBSIR

Soutenu le ../../...., devant le jury formé de :

Mr.	M.B. Yagoubi	Président	Université de Laghouat, Algérie	Professeur
Mme	H. Cherroun	Examinateur	Université de Laghouat, Algérie	MCA
Mr.	N. Lagraa	Examinateur	Université de Laghouat, Algérie	MCA
Mr.	M. Benmohammed	Rapporteur	Université de Constantine 2, Algérie	Professeur

N° d'ordre :...../2013 - M/DGI

Remerciement

Dieu, merci pour m'avoir donné la force et la volonté de mener à bien ce travail.

Je tiens à remercier tout particulièrement mon encadreur, le professeur Mohamed Benmohamed pour m'avoir fait confiance et m'avoir suggéré ce thème. Ses compétences, sa permanente disponibilité et ses encouragements m'ont été très bénéfiques et m'ont beaucoup aidé. Je souligne aussi sa démarche pédagogique vis-à-vis des jeunes chercheurs dont le premier but a été de consolider et de promouvoir mes compétences.

Je remercie également la Directrice du Centre de Recherche en Biotechnologie, docteur Halima Benbouza pour sa patience et ses précieux conseils, monsieur Romain Vernoux qui m'a donné un total accès et modification du code source de son application AES Crypto, ainsi que M^{elle} Imen Bensetira pour ses précieux conseils et orientations lors de l'établissement de ma communication internationale.

Mes remerciements les plus sincères, ainsi que mon entière gratitude vont à mes parents ainsi que mes frères et sœurs, Nevine en particulier, qui m'ont accompagné et soutenu tout au long de mon parcours.

Je remercie les membres de mon jury de soutenance d'avoir accepté d'évaluer cette modeste contribution.

A vous tous j'exprime ma reconnaissance et ma gratitude car encore une fois c'est grâce à vous tous et à votre abnégation et à votre altruisme que j'ai pu poursuivre mes études.

Résumé

Les champs d'application des systèmes embarqués sont en croissance rapide, avec des dispositifs tels que les Smartphones, les PDA, les cartes à puce, etc. A l'horizon des technologies futuristes telles que les réseaux de capteurs, qui promettent encore une plus grande interaction entre les humains et les machines.

Comme les systèmes embarqués sont de plus en plus intégrés dans des infrastructures personnelles et commerciales, la sécurité est devenue primordiale.

La sécurité des systèmes embarqués est désormais une partie intégrante de la sécurité informatique. Pour une bonne sécurité, des protocoles, des technologies, des dispositifs, des outils et des techniques sont utilisés pour baisser la probabilité des menaces.

Les fabricants des systèmes embarqués ne donnent pas beaucoup d'attention à la sécurité lors de la conception de ces systèmes, ce qui peut conduire à des échecs de conception.

Le but de ce mémoire est de mettre l'accent sur les principes de la sécurité dans les systèmes embarqués et mesurer l'impact des architectures embarquées sur de différents algorithmes cryptographiques.

Une étude statistique sur l'implémentation de ces algorithmes dans des Smartphones modernes fonctionnant sous le système d'exploitation Android avec des architectures ARM, est présentée dans ce travail, cela permet de mettre l'accent sur les différences pouvant influencer les performances entre ces architectures.

Mots clés : Systèmes embarqués, Sécurité, Cryptographie.

Table des matières

Table des Figures

Table des Tableaux

Lexique

AAA	Authentication, Authorization and Accounting : processus de contrôle et de gestion des accès des utilisateurs à un système informatique.
AMD	Advanced Micro Devices : un fabricant de semi-conducteurs, notamment de microprocesseurs.
Android	Android est un système d'exploitation open source utilisant le noyau Linux, pour Smartphones, tablettes tactiles, PDA et terminaux mobiles, conçu par Android, une startup rachetée par Google, et annoncé officiellement le 5 novembre 2007.
API	Application Programming Interface (interface de programmation applicative).
ARM	Les ARM sont des architectures, développées par ARM Ltd. Ce sont des architectures RISC 32 bits (ARMv3 à ARMv7) et 64 bits (ARMv8) introduite à partir de 1983. Dotés d'une architecture relativement plus simple que d'autres familles de processeurs, et bénéficiant d'une faible consommation, les processeurs ARM sont devenus dominants dans le domaine de l'informatique embarquée, en particulier la téléphonie mobile et les tablettes.
CISC	Complex instruction set computer (Un microprocesseur à jeu d'instruction étendu).

Dalvik	Dalvik est la machine virtuelle logiciel open-source utilisée dans les appareils mobiles Android.
ECC	Elliptice-curve cryptography (cryptographie sur les courbes elliptiques).
ERP	Enterprise Resource Planning (planification des ressources de l'entreprise ou progiciel de gestion intégré PGI). Ce type de logiciel correspond pour une organisation au support de base capable d'assurer une gestion intégrée, définie comme étant l'interconnexion et l'intégration de l'ensemble des fonctions de l'entreprise dans un système informatique centralisé (et généralement configuré selon le mode client-serveur).
FPU	Floating Point Unit : unité de calcul en virgule flottante, c'est une partie d'un processeur, spécialement conçue pour effectuer des opérations sur des nombres à virgule flottante.
FTP	File Transfer Protocol (protocole de transfert de fichiers), c'est un protocole de communication destiné à l'échange informatique de fichiers sur un réseau TCP/IP
GPL	General Public License : C'est une licence qui fixe les conditions légales de distribution des logiciels libres du projet GNU.
HAL	Hardware Abstraction Layer (La couche d'abstraction matérielle) : C'est une spécification et un utilitaire logiciel qui traque les périphériques du système informatique. Le but du HAL est d'éviter aux développeurs d'implémenter manuellement le code spécifique à un périphérique. À la place, ils peuvent utiliser une couche connectable qui fournit des informations à propos du dit périphérique, tel que cela se passe par

	exemple lorsqu'un utilisateur branche ou débranche un périphérique USB.
HTTP	HyperText Transfer Protocol (protocole de transfert hypertexte): C'est un protocole de communication client-serveur développé pour le World Wide Web.
HTTPS	HyperText Transfer Protocol Secure (protocole de transfert hypertexte sécurisé): C'est la combinaison du http avec une couche déchiffrement comme SSL ou TLS.
IPsec	Internet Protocol Security : défini comme un cadre de standards ouverts pour assurer des communications privées et protégées sur des réseaux IP, par l'utilisation des services de sécurité cryptographiques, est un ensemble de protocoles utilisant des algorithmes permettant le transport de données sécurisées sur un réseau IP.
ISA	instruction set architecture : architecture de jeu d'instructions ou tout simplement architecture (de processeur), c'est la spécification fonctionnelle d'un processeur, du point de vue du programmeur en langage machine.
ISO	International Organization for Standardization (Organisation internationale de normalisation).
JIT	just-in-time compilation ou JIT compilation : la compilation à la volée est une technique visant à améliorer la performance de systèmes bytecode-compilés par la traduction de bytecode en code machine natif au moment de l'exécution.

JVM	anglais Java virtual machine (machine virtuelle Java) : C'est une plate-forme informatique qui exécute des programmes compilés sous forme de bytecode Java.
LAN	Local Area Network (réseau local).
LCD	Liquid Crystal Display (écran à cristaux liquides).
NIST	National Institute of Standards and Technology.
PDA	Personal Digital Assistant (assistant numérique personnel).
PKI	Public Key Infrastructure (infrastructure à clés publiques (ICP) ou infrastructure de gestion de clés (IGC)).
PPP	Point to Point Protocol (Protocole point à point).
QoS	Quality of Service (Qualité de service).
Régression	La régression est un ensemble de méthodes statistiques très utilisées pour analyser la relation d'une variable par rapport à une ou plusieurs autres.
RISC	Reduced Instruction-Set Computer (microprocesseur à jeu d'instruction réduit).
RTOS	Real-Time Operating System (Système d'exploitation temps réel).
SCADA	Supervisory Control and Data Acquisition (télésurveillance et acquisition de données).
SSL	Secure Sockets Layer, un protocole de sécurisation des échanges sur Internet, devenu Transport Layer Security (TLS) en 2001.

TCP/IP	La suite TCP/IP est l'ensemble des protocoles utilisés pour le transfert des données sur Internet. TCP (Transmission Control Protocol) et IP (Internet Protocol).
TLS	Transport Layer Security : ensemble de protocoles de sécurisation des échanges sur Internet.
VLIW	Very Long Instruction Word (Mot d'instruction très long).
VPN	Virtual Private Network (réseau privé virtuel).
WAN	Wide area network (réseau étendu).
X86	La famille x86 regroupe les microprocesseurs compatibles avec le jeu d'instructions de l'Intel 8086.

Introduction Générale

Définition d'un Système Embarqué :

Un système embarqué est un système informatique à usage spécifique, par opposition à d'autres types de systèmes informatiques tels que les ordinateurs personnels (PC) ou superordinateurs. Cependant, la définition de «système embarqué» est difficile à cerner, car il évolue constamment avec les progrès technologiques et les baisses dramatiques des coûts de la mise en œuvre de différents composants matériels et logiciels.

Voici quelques-unes des descriptions les plus courantes d'un système embarqué: (1)

- Les systèmes embarqués sont des systèmes limités dans le matériel et le logiciel qu'un ordinateur personnel (PC). Cela est vrai pour un sous-ensemble important de la famille des systèmes embarqués. Pour ce qui est des limitations matérielles, cela peut signifier des limites des performances de traitement, de la consommation d'énergie, de la mémoire, etc. Au niveau logiciel, cela signifie généralement des limitations relatives à certaines applications, pas de système d'exploitation (OS) ou des systèmes d'exploitation très limités.

Cependant, cette définition n'est que partiellement vrai aujourd'hui, elle ne peut définir que des cartes à puce ou quelques autres systèmes embarqués bien destinés. Il y a d'autres systèmes embarqués très complexes.

- Un système embarqué est conçu pour exécuter une fonction spécifique.

La plupart des appareils embarqués sont principalement conçus pour une fonction spécifique. Cependant, on trouve ces jours des dispositifs tels que les assistants numériques (PDA), les Smartphones, etc., qui sont des systèmes embarqués conçus pour être en mesure de faire une variété de fonctions primaires.

- Un système embarqué est un système informatique de meilleure qualité et de fiabilité que les autres types de systèmes informatiques.

Certaines familles de dispositifs embarqués ont un seuil très élevé d'exigences de qualité et de fiabilité. Par exemple, si le contrôleur moteur d'une voiture tombe en panne pendant qu'on conduise sur une autoroute fréquentée ou un mauvais fonctionnement

d'appareils médicaux essentiels pendant la chirurgie, des problèmes très sérieux peuvent se résulter. Cependant, il existe également des dispositifs embarqués, tels que les téléviseurs, les jeux et les téléphones cellulaires, dans lequel un dysfonctionnement est un inconvénient, mais n'est généralement pas un danger de mort.

Une des définitions qui peut être judicieuse est la suivante : (2)

- [Un système embarqué peut être défini comme un système électronique et informatique autonome ne possédant pas des entrées/sorties standards comme un clavier ou un écran d'ordinateur. Le système matériel et l'application 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.

  Un système embarqué : (2)

  - Est un système numérique.
  - Utilise généralement un processeur.
  - Exécute un logiciel dédié pour réaliser une fonctionnalité précise.
  - Remplace souvent des composants électromécaniques.
  - N'a pas réellement de clavier standard.
  - L'affichage est limité ou n'existe pas du tout.
  - N'est pas un PC.
  - N'exécute pas une application scientifique ou commerciale traditionnelle.

Systèmes embarqués en réseau

Les systèmes embarqués en réseau sont, en générale, des systèmes qui sont capables de se connecter à des réseaux et qui comportent un processeur, une mémoire, un espace disque et un système d'exploitation. Ces dispositifs spéciaux se limitent généralement à des fonctionnalités spécifiques.

Quelques exemples courants des systèmes embarqués en réseau sont les Routeurs, les Commutateurs, les Smartphones, les consoles de jeux (ex. : PlayStation), etc.

Un document publié par CISCO indique que « Un nombre croissant de trafic IP et Internet est originaire de périphériques non-PC. En 2011, 22 pour cent du trafic IP avec

origine de périphériques non-PC, mais d'ici à 2016 le trafic IP non-PC passera à 31 pour cent. En 2011, seulement 6 pour cent du trafic Internet grand public avec son origine périphériques non-PC, mais d'ici à 2016 le trafic Internet grand public non-PC passera à 20 pour cent. » (3)

Puisque ces dispositifs sont déployés sur des réseaux, des vulnérabilités et des exploits seront découverts et par conséquent, ces systèmes nécessitent de la protection en utilisant les différentes approches de sécurité.

Sécurité des systèmes embarqués en réseau

Une des croyances erronées est que les Systèmes embarqués sont plus difficiles à attaquer que les systèmes informatiques à usage universel.

Les systèmes embarqués partagent les mêmes faiblesses que les autres systèmes informatiques. Beaucoup de ces dispositifs peuvent contenir des Backdoor, ou des services non sécurisé activés.

Beaucoup des systèmes embarqués modernes, contiennent des fonctionnalités avancées, on peut même installer un serveur FTP ou un serveur Web sur un Smartphone Android, ou même le rendre un point d'accès sans fil ou un modem, augmentant ainsi la probabilité qu'un attaquant puisse pénétrer dans le système ou le mettre dans un état de DoS.

La sécurité repose sur trois principes: la confidentialité, l'intégrité et la disponibilité. Pour assurer ces trois principes dans les systèmes embarqués en réseau, ces derniers doivent êtres sécurisés par les différentes approches de sécurité. On ne peut en aucun cas, les supposer loin des attaques.

Les ressources limitées de ces systèmes embarqués causent un défi important pour l'implémentation des politiques de sécurité efficaces qui, sont généralement exigeantes en ressources. Ainsi, Il faut faire une analyse soigneuse des approches de sécurité afin de s'assurer qu'elles peuvent être implémentées pour sécuriser le système.

Organisation du mémoire

Les systèmes embarqués se vari de l'architecture jusqu'à l'implémentation, ainsi les approches de sécurité peuvent être étudiées et choisies à chaque niveau dans le processus

de création et de mise en œuvre du système embarqué. Un bon choix de conception, du niveau le plus bas de l'architecture jusqu'au niveau le plus haut, donne une bonne sécurité au système.

Ce mémoire est décomposé en deux parties, et chaque partie est décomposées en plusieurs chapitres;

La première partie du mémoire (Etat de l'art), contient trios chapitres ;

Le chapitre 1 concerne l'architecture en couche des systèmes embarqués (allant de la couche physique à la couche application), qui peut faciliter l'étude et l'implémentation de la sécurité, ainsi que la diversité des processeurs embarqués existants pour pouvoir étudier les différences architecturales qui peuvent influencer la sécurité. Le chapitre 2 concerne la Cryptographie et son utilisation pour sécuriser les communications. Dans ce chapitre, on présente le chiffrement symétrique et asymétrique ainsi que les signatures numériques. Le chapitre 3 Concerne la sécurité dans les systèmes embarqués en détail, dans ce chapitre on parle des paramètres de sécurité, des contraintes de sécurité dans les systèmes embarqués et de la conception des systèmes embarqués sécurisés.

La deuxième partie du mémoire (Contribution), concerne l'implémentation, elle est décomposée de la même structure que la partie une ; elle contient 4 chapitres ;

Le chapitre 4 concerne les travaux connexes. Le chapitre 5 concerne les architectures ARM, sur lesquels on va faire des tests. Le chapitre 6 concerne le système d'exploitation Android, on parle de son architecture, de sa sécurité, de la possibilité qu'il soit temps réel, et de ses packages cryptographiques, et on termine par le chapitre 7 qui concerne une analyse expérimentale d'algorithmes cryptographiques dans les systèmes embarqués, et qui est une étude de cas sur les Smartphones-Android.

Partie 1 :

Etat de l'art

1 Systèmes Embarqués et temps réel

1.1 Introduction

L'architecture en couche d'un système embarqué est une abstraction de l'appareil, il s'agit d'une généralisation du système qui ne dispose généralement pas d'informations détaillées, telles que le code source du logiciel ou la conception des circuits. Au niveau architectural, les composants matériels et logiciels du système embarqué sont donc représentés comme des compositions des éléments en interaction. Ces éléments sont une représentation du matériel et/ou du logiciel dont la mise en œuvre détaillée a été extraite.

En bref, une architecture embarquée contient les éléments du système embarqué, les éléments qui interagissent avec le système embarqué, les propriétés de chacun des éléments individuels et les relations interactives entre ces éléments.

Les systèmes embarqués comme tout système informatique implémentent une couche logiciel au-dessus de la couche physique, allant du système d'exploitation jusqu'aux applications. Par contre, la plus part des systèmes embarqués sont temps réel.

Dans ce chapitre, on va présenter le modèle en couche des systèmes embarqués, allant de la couche physique à la couche application.

1.2 Importance de l'architecture d'un système embarqué

Les défis les plus importants l'or de la conception des systèmes embarqués sont : (4)

- La limite des coûts.
- La détermination des contraintes sur le système, comme la fiabilité et la sécurité.
- Le travail dans les limites des éléments disponibles (puissance de traitement, mémoire, batterie, etc.).
- La commercialisation du système.
- Etc.

En bref, une architecture peut être utilisée pour résoudre ces problèmes au début d'un projet. C'est le premier élément à être analysé et utilisé comme un grand plan définissant

le niveau de l'infrastructure de la conception, des options possibles et des contraintes de conception.

L'approche architecturale facilite même l'étude et la conception du système par une variété de personnes même ceux sans bagage technique, qui peuvent agir en tant que planificateurs du projet.

Aussi, les différentes structures de l'architecture peuvent être mis à profit pour la conception d'un autre produits dans le future, présentant des caractéristiques similaires, permettant ainsi une réutilisation des concepts et conduisant à une diminution des coûts de développement.

D'une façon générale, une variété de structures architecturales est utilisée pour introduire les concepts techniques des systèmes embarqués. Au plus haut niveau, le principal outil architectural utilisé pour présenter les éléments principaux situés à l'intérieur d'une conception de systèmes embarqués est le modèle des systèmes embarqués, illustré dans la Figure 1-1: Modèle des systèmes embarqués (1)

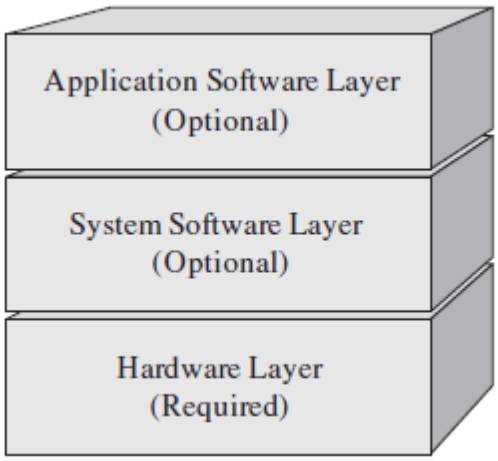


Figure 1-1: Modèle des systèmes embarqués (1)

Ce modèle indique que tous les systèmes embarqués partagent une même similitude au niveau le plus haut ; c'est qu'ils ont tous une couche physique (matériel) ou bien, ils sont composés de toutes les couches (Physique, Système, Applications).

La couche physique contient tous les composants matériels intégrés dans une carte mère ; tandis que les couches Système et Applications contiennent la partie logicielle qui sera exécutée par le système embarqué.

Ce modèle de référence est une représentation en couche d'une architecture d'un système embarqué à partir duquel, on peut dériver une structure architecturale modulaire.

Quel que soit la différence entre les dispositifs embarqués, il est possible de comprendre l'architecture de l'ensemble de ces systèmes en visualisant et en regroupant les composants à l'intérieur de ces dispositifs sous forme de couches. Bien que le concept de décomposition en couches n'est pas destiné seulement à la conception des systèmes embarqués mais plutôt pour tous les systèmes informatique dont les systèmes embarqués font partie. Cependant, ce concept est un bon outil pour visualiser les combinaisons possibles des centaines, voire des milliers, de composants matériels et logiciels qui peuvent être utilisés dans la conception d'un système embarqué.

En général, on choisit cette représentation modulaire de l'architecture des systèmes embarqués pour deux raisons principales: (1)

1. La représentation visuelle des principaux éléments et leurs fonctions associées : cette approche permet de visualiser les différents composants d'un système embarqué et leurs interrelations.
2. Les représentations architecturales en couches sont largement utilisées pour structurer les projets embarqués dû au fonctionnement indépendant des différents modules.

1.3 Les processeurs embarqués

Les processeurs sont les principales unités fonctionnelles d'une carte imprimée, et sont principalement responsables du traitement des instructions et des données. Tout dispositif électronique contient au moins un processeur maître, agissant en tant que dispositif central, et peut avoir d'autres processeurs esclaves qui sont contrôlés par le processeur maître.

Il y a des centaines de processeurs embarqués disponibles, et aucun d'entre eux n'est actuellement dominant dans le domaine de la conception des systèmes embarqués. Malgré le grand nombre de modèles disponibles, les processeurs embarqués peuvent être séparés en différents «groupes» appelés architectures. Ce qui différencie une architecture d'un groupe d'une autre est l'ensemble des instructions de code machine que

les processeurs à l'intérieur du groupe de l'architecture peuvent exécuter. Les processeurs qui peuvent exécuter les mêmes instructions du code machine sont considérés comme étant de la même architecture. Le Tableau 1-1 contient quelques exemples des processeurs embarqués existants ainsi que leurs familles d'architecture.

Tableau 1-1: : Exemples d'Architectures et de Processeurs (1)

Architecture	Processeur	Fabricant
AMD	Au1xxx	Advanced Micro Devices, …
ARM	ARM7, ARM9, …	ARM, …
C16X	C167CS, C165H, C164CI, …	Infineon, …
ColdFire	5282, 5272, 5307, 5407, …	Motorola/Freescale, …
I960	I960	Vmetro, …
M32/R	32170, 32180, 32182, 32192, …	Renesas/Mitsubishi, …
M Core	MMC2113, MMC2114, …	Motorola/Freescale
MIPS32	R3K, R4K, 5K, 16, …	MTI4kx, IDT, MIPS Technologies, …
NEC	Vr55xx, Vr54xx, Vr41xx	NEC Corporation, …
PowerPC	82xx, 74xx,8xx,7xx,6xx,5xx,4xx	IBM, Motorola/Freescale, …
68k	680x0 (68K, 68030, 68040, 68060, …), 683xx	Motorola/Freescale, …
SuperH (SH)	SH3 (7702,7707, 7708,7709), SH4 (7750)	Hitachi, …
SHARC	SHARC	Analog Devices, Transtech DSP, Radstone, …
strongARM	strongARM	Intel, …
SPARC	UltraSPARC II	Sun Microsystems, …
TMS320C6xxx	TMS320C6xxx	Texas Instruments, …
x86	X86 [386,486,Pentium (II, III, IV)…]	Intel, Transmeta, National Semiconductor, Atlas, …
TriCore	TriCore1, TriCore2, …	Infineon, …

1.3.1 Modèles d'architecture ISA

Les fonctionnalités qui sont intégrées dans le jeu d'instructions d'une architecture sont communément appelées le jeu d'instructions de l'architecture ou ISA (Instruction Set Architecture). L'ISA définit des caractéristiques telles que : les opérations qui peuvent être utilisées par les programmeurs pour créer des programmes pour cette architecture, les opérandes (données) qui peuvent être acceptées et traitées par une architecture, le modes d'adressage utilisés pour accéder aux opérandes, et le traitement des interruptions.

1.3.1.1 Modèles ISA à usage général

Les modèles ISA à usage général sont généralement mises en œuvre dans les processeurs ciblés pour être utilisés dans une grande variété de systèmes, plutôt que seulement dans certains types de systèmes embarqués. Les deux types d'architectures ISA à usage général qui sont mis en œuvre dans les processeurs embarqués sont:

1.3.1.2 Le modèle CISC (Complex Instruction Set Computing)

L'ISA CISC, comme son nom l'indique, définit les opérations complexes constitués de plusieurs instructions. Le Motorola/Freescale 68000 est un exemple de l'architecture ISA CISC.

1.3.1.3 Le modèle RISC (Reduced Instruction Set Computing)

Contrairement au modèle CISC, le RISC définit habituellement:

- Une architecture avec des opérations simples et composées de moins d'instructions qu'une architecture CISC.
- Une architecture qui a un nombre réduit de cycles par opération.

ARM, PowerPC, SPARC et MIPS sont des exemples des architectures RISC.

1.3.1.4 CISC vs RISC

Dans le domaine de l'informatique à usage générales, de nombreux modèles de processeur actuels relèvent de la catégorie CISC ou RISC principalement en raison de leurs origines. Les processeurs RISC sont devenus plus complexes, tandis que les processeurs CISC sont devenus plus efficaces en les comparant avec leurs homologues RISC, brouillant ainsi la frontière entre la définition d'une architecture RISC par rapport à une architecture CISC. Techniquement, ces processeurs ont à la fois des attributs RISC et CISC, indépendamment de leurs définitions.

1.3.1.5 Modèles ISA avec parallélisme d'instructions

Les architectures ISA avec parallélisme d'instructions, sont similaires à celles à usage générale, sauf qu'elles exécutent plusieurs instructions en parallèle, comme leur nom l'indique. En fait, les architectures ISA avec parallélisme d'instructions sont considérées comme une évolution des architectures RISC, c'est l'une des principales raisons pour lesquelles les architectures RISCs sont à la base pour le parallélisme.

Les machines SIMD (Single Instruction Multiple Data), MIMD (Multiple Instruction Multiple Data) et VLIW (Very Long Instruction Word Computing) sont des exemples des Modèles ISA avec parallélisme d'instructions.

1.3.2 Performances des processeurs

Il existe plusieurs mesures de performance du processeur, qui sont toutes basées sur le comportement du processeur sur une période de temps donnée. L'une des définitions les plus courantes de la performance du processeur est le débit du processeur, la quantité de travail dont le CPU peut faire en une période de temps donnée.

L'exécution d'un processeur est finalement synchronisée par un système externe ou une horloge maîtresse, située sur la carte. L'horloge maîtresse est simplement un oscillateur produisant une séquence de fréquence fixe des signaux d'impulsions réguliers qui sont généralement divisées ou multipliées dans l'unité centrale pour générer au moins un signal d'horloge interne avec un nombre constant de cycles d'horloge par seconde, pour commander et coordonner l'extraction, le décodage et l'exécution des instructions. La fréquence d'horloge du processeur est exprimée en mégahertz (MHz) ou en Gigahertz (GHz).

En utilisant la fréquence d'horloge, le temps d'exécution du CPU (qui est le temps total en secondes nécessaire pour que le processeur traite un programme) peut être calculé.

À partir de la fréquence d'horloge, on peut calculer la durée pour que le processeur termine un cycle d'horloge (c'est l'inverse de la fréquence d'horloge (1/Fréquence)), appelée la période d'horloge ou le temps du cycle et exprimé en secondes par cycle.

La fréquence de l'horloge du processeur ou de la période d'horloge est généralement dans la documentation des spécifications du processeur.

1.3.2.1 X86 VS ARM

La famille x86 regroupe les microprocesseurs compatibles avec le jeu d'instructions de l'Intel 8086. Les x86 ne sont plus des processeurs CISC traditionnels, mais ils sont devenus plutôt des processeurs hybrides bénéficiant ainsi des avantages de chaque architecture.

Les architectures ARM, sont des architectures RISC 32 bits (ARMv3 à ARMv7) et, 64 bits (ARMv8). Dotés d'une architecture relativement plus simple que d'autres familles de processeurs, et bénéficiant d'une faible consommation, les processeurs ARM sont devenus dominants dans le domaine de la téléphonie mobile et les tablettes.

Le monde des ordinateurs portables pourrait connaître prochainement une évolution avec le remplacement progressif des processeurs x86 par l'architecture ARM. Windows 8 est compatible avec cette architecture (avec certaines limitations), tout comme Google Chrome OS. L'utilisation de l'architecture ARM devrait permettre la réduction de la consommation électrique.

Par contre, les fondeurs des processeurs x86 (Intel et AMD) se préparent à cette concurrence en réduisant la consommation électrique de leurs solutions et en simplifiant leurs architectures, comme avec les Atom et Bobcat. De l'autre côté, les fondeurs des processeurs à base d'architecture ARM, comme NVidia et Qualcomm, continuent d'augmenter les performances de leurs puces, par exemple en augmentant le nombre de cœurs ou en ajoutant de nouvelles instructions. Ces derniers, pour éviter que leurs solutions consomment trop d'énergie, peuvent ajouter un processeur compagnon limité en performance (donc économe en énergie) à côté du multi-cœur énergivore, par exemple le Tegra 3 de NVidia.

1.4 Temps Réel

Les systèmes embarqués temps réel peuvent être trouvés dans beaucoup de domaines d'application différents, par exemple, les véhicules (l'airbag, système de freinage, etc.), les systèmes de communication, les systèmes d'automatisation industrielle, etc.

Parmi les caractéristiques importantes des systèmes Temps réel, c'est qu'ils sont des systèmes informatiques qui interagissent physiquement avec le monde réel, et qui ont des exigences de temps.

En règle générale, les interactions avec le monde réel se font via des capteurs, plutôt que des claviers et des écrans des ordinateurs traditionnels.

La grande majorité des systèmes informatiques embarqués sont des systèmes temps réel. Les applications temps réels sont les plus dominantes de la technologie informatique embarquée.

L'inclusion de la sécurité dans ce type de systèmes peut causer des délais de réponse importants ce qui peut engendrer des échéances, et ainsi, les concepteurs des systèmes embarqués sécurisés doivent toujours penser au temps réel de ces derniers lors de la phase de conception.

D'une manière générale, On distingue le temps réel strict ou dur (de l'anglais hard real-time) et le temps réel souple ou mou (soft real-time) suivant l'importance accordée aux contraintes temporelles. Le temps réel strict ne tolère aucun dépassement de ces contraintes, ce qui est souvent le cas lorsque de tels dépassements peuvent conduire à des situations critiques, voire catastrophiques (pilote automatique d'avion, système de surveillance de centrale nucléaire, etc.). À l'inverse le temps réel souple s'accommode des dépassements des contraintes temporelles dans certaines limites au-delà desquelles le système devient inutilisable (visioconférence, jeux en réseau, etc.).[1]

On peut ainsi considérer qu'un système temps réel strict doit respecter des limites temporelles données même dans les pires des situations d'exécution possibles. En revanche un système temps réel souple doit respecter ses limites pour une moyenne de ses exécutions. On tolère un dépassement exceptionnel, qui sera peut-être rattrapé à l'exécution suivante. (5)

Cette diversité des exigences imposées par les différents domaines d'application (soft real-time/ hard real-time) a nécessité des solutions différentes en utilisant différents protocoles fondés sur des principes de fonctionnement différents. On trouve des réseaux où on met en œuvre les sept couches du modèle OSI alors que dans d'autres (comme dans l'automatisation industrielle), on n'a pas besoin de la fonctionnalité de routage et de contrôle de bout-en-bout. Par conséquent, en général, seules les couches 1 (couche physique), 2 (couche liaison de données), et 7 (couche application) sont utilisés dans ces réseaux.

[1] Voir aussi Wikipédia

Malheureusement, il est impossible de construire des systèmes qui satisfont les contraintes du temps réel strict, en raison de l'imperfection du matériel. Le mieux que l'on puisse faire est un système qui, avec une probabilité très élevée fournit le comportement attendu pendant un intervalle de temps acceptable.

Une des techniques utilisées pour assurer le temps réel est la gestion de la qualité du service (QoS) qui doit être prise en compte soit par l'application, soit par le système d'exploitation, soit par une combinaison des deux techniques.

1.4.1 Logiciels/systèmes d'exploitation embarqués et temps réel

La plus part des systèmes embarqués nécessitent des systèmes d'exploitation ainsi que des applications temps réel.

1.4.1.1 Logiciel embarqué

Un logiciel embarqué (embedded software en anglais) est un programme utilisé dans un équipement (qui n'est pas un PC) industriel. La différence essentielle avec un logiciel classique tient à la complète intégration du logiciel embarqué dans cet équipement : il n'a pas de raison d'être en dehors de l'équipement pour lequel il a été conçu. On parle également de logiciel intégré ou dédié et personne n'achète un tel équipement pour son logiciel lui-même, mais bien évidemment pour la qualité des services que remplit l'équipement.

Cette logique doit être prise en compte par les concepteurs des logiciels embarqués : l'équipement est valorisé uniquement par son aspect fonctionnel. Dans le cas de logiciels de très petite taille destinés à des tâches très spécifiques, on parle d'ailleurs en anglais de deeply embedded software, ce qui peut se traduire par logiciel profondément enfoui. (6)

1.4.1.2 Caractéristiques d'un logiciel embarqué

Les points suivants permettent de caractériser un logiciel embarqué : (7)

1.4.1.2.1 Ciblé

Son domaine d'action est limité aux fonctions pour lesquelles il a été créé.

1.4.1.2.2 Fiable et sécurisé

Le logiciel nécessite une grande fiabilité car il est destiné à un fonctionnement complètement autonome, une erreur ou un retardement d'exécution peut causer des problèmes fatals. Cette contrainte explique la grande différence entre les logiciels embarqués et les logiciels classiques.

1.4.1.2.3 Maintenable dans le temps

Dans la majorité des domaines industriels, et en dehors du logiciel classique, la durée de vie des produits est longue de par des obligations légales ou du moins commerciales qui obligent l'industriel à maintenir le produit pendant une dizaine d'années, cas de l'industrie automobile. Dans le cas d'industries plus sensibles comme l'aéronautique, le militaire ou le spatial, cette durée peut être doublée. Il est donc indispensable que le logiciel embarqué soit maintenable durant toute la durée de vie du produit en cas de découverte de problème de fonctionnement majeur.

1.4.1.2.4 Spécifique

L'interface de dialogue avec l'utilisateur est spécifique : dans la majorité des cas, un tel logiciel n'utilise pas les interfaces classiques clavier/souris propres à la micro-informatique.

S'ils existent, les périphériques d'affichage sont souvent limités à des panneaux de petite taille de type LCD (Liquid Crystal Display), et les périphériques d'entrée à quelques boutons poussoirs ou autres composants inhabituels dans l'informatique traditionnelle.

Tout cela, combiné aux contraintes d'optimisation, nécessite une approche matérielle, proche de l'électronique, ce qui accentue encore le fossé avec le développeur standard habitué à un confort presque indécent. Les concepteurs de logiciels embarqués sont rarement des informaticiens purs, plus souvent des gens qui viennent des domaines hybrides.

1.4.1.2.5 Optimisé

Le plus souvent, ce logiciel est de petite taille si on le compare aux volumes démesurés atteints par les logiciels classiques multi-usages comme en bureautique. Cela est dû à plusieurs raisons :

- La nécessité de fiabilité citée précédemment cohabite mal avec un volume démesuré : plus on écrit de lignes de codes, plus on a de chances que celles-ci contiennent des bogues.
- Le logiciel est souvent embarqué dans des équipements produits à grande échelle sur lesquels le moindre écart de coût peut avoir de fortes répercussions.
- L'optimisation est également importante au niveau du temps de réponse. Le consommateur verra d'un très mauvais œil une soi-disant évolution technologique qui ralentit le fonctionnement de l'équipement.
- Etc.

1.4.1.3 Système d'exploitation embarqué

En Software, il est fréquent d'entendre la terminologie de système embarqué. Cette terminologie désigne le plus souvent un système d'exploitation, version complexe et multi-usage du concept de logiciel.

Un système d'exploitation est un ensemble de programmes permettant :

- de gérer les ressources matérielle en assurant leurs partages entre un ensemble plus ou moins grand d'applications ;
- d'assurer un ensemble de services en présentant aux utilisateurs une interface mieux adaptée à leurs besoins que celle de la machine physique.

Un système d'exploitation est a priori beaucoup trop complexe et surdimensionné pour remplir les tâches décrites précédemment. Mais l'amélioration des performances du matériel a permis dans un grand nombre de cas d'utiliser un système d'exploitation adapté au lieu de simples logiciels dédiés.

Les avantages de l'utilisation d'un système d'exploitation sont les suivants : (7)

- L'écriture du support de standards du marché comme l'USB est extrêmement lourde en cas de non-utilisation d'un système d'exploitation. Dans le cas d'un système, ce travail est réalisé par une autre équipe spécialisée ou un fournisseur externe, ce qui permet de diminuer le temps de développement et donc les coûts.
- Si le système d'exploitation utilisé est suffisamment répandu, il permet aux applications industrielles et embarquées de bénéficier des mêmes avancées

technologiques que les applications classiques. C'est ainsi qu'il est aujourd'hui possible d'utiliser dans des systèmes réduits des protocoles de communication hérités de l'informatique classique et du multimédia, nous pouvons citer par exemple l'utilisation généralisée du protocole TCP/IP et de ses dérivés comme HTTP (Hyper Text Transfer Protocol) ou FTP (File Transfer Protocol) dans des procédures de communication entre des systèmes classiques et des systèmes embarqués.

Les systèmes d'exploitation fournissent en général un environnement de développement facilitant la mise au point des programmes dans un contexte beaucoup plus accueillant et performant que le système cible (celui sur lequel est censé tourner le programme définitif). Les Smartphones Android est un bon exemple. Les développeurs d'application pour ce système ne doivent pas disposer de l'appareil cible immédiatement, ils utilisent un environnement de développement basé sur Eclipse permettant de simuler le fonctionnement de l'appareil sur des stations de travail.

Par contre, le principal inconvénient de l'utilisation d'un véritable système d'exploitation proche d'un système classique est la taille mémoire nécessaire. Il est bien évident que, si l'espace disponible est réduit à quelques centaines d'octets, un vrai système d'exploitation ne s'imposera pas.

1.4.1.4 La contrainte du temps réel dans les systèmes embarqués

Les systèmes d'exploitation à usage universelle ne prennent pas des contraintes externes pour la gestion du temps, et ainsi, ils ne peuvent pas répondre à des contraintes temporelles alors qu'il faut respecter des échéances pour les tâches critiques dans le cas des systèmes embarqués, aussi, l'inclusion de la sécurité dans les systèmes embarqués peut causer des délais d'exécution plus importants d'où la nécessité du temps réel.

D'un point de vue technique, les champs d'application des systèmes embarqués peuvent être grossièrement divisés en deux grandes familles :

- le contrôle de processus sans contrainte ou à faible contrainte temps réel ;
- le contrôle de processus avec contrainte temps réel.

Cette séparation est fondamentale car elle déterminera complètement le choix des composants logiciels à utiliser. Dans le premier cas, on pourra envisager l'utilisation d'un système d'exploitation dérivé d'un système classique comme Linux. L'adaptation se situera principalement au niveau du développement de pilotes de périphériques et de l'optimisation du système en taille ou en performances. Dans le second cas, les contraintes matérielles nécessiteront l'utilisation de composants spécialisés, soit un logiciel spécifique, soit un système d'exploitation dit temps réel (Real Time Operating System ou RTOS).

Les équipements grand publique fonctionnaient de manière isolée et n'avaient aucun lien avec les réseaux informatiques. Le développement des services sur Internet a incité les industriels à intégrer des produits initialement peu communicants dans des environnements en réseau. Cette intégration nécessite l'utilisation de protocoles de communication hérités de l'informatique, et donc le plus souvent d'intégrer des couches logicielles supportant ces protocoles (Ex. : Distributeur de billets).

Un système d'exploitation temps réel comme un système d'exploitation à usage général fournit des services pour l'accès et le partage des ressources. Un système d'exploitation temps réel, cependant, fournit des services supplémentaires adaptés à la contrainte du temps réel. L'utilisation d'un système d'exploitation à usage universel dans les systèmes temps réel présente plusieurs inconvénients:

- L'utilisation excessive des ressources (mémoire, CPU, etc.).
- L'accès difficile au matériel et dispositifs à cause de l'absence des interruptions dans le niveau Application.
- Synchronisation difficile entre les applications.

1.4.1.4.1 Temps partagé et temps réel

La gestion du temps est l'un des problèmes majeurs des systèmes d'exploitation. La raison en est simple : les systèmes d'exploitation modernes sont tous multitâches, or ils utilisent du matériel basé sur des processeurs qui ne le sont pas, ce qui oblige le système à partager le temps du processeur entre les différentes tâches. Cette notion de partage implique une gestion du passage d'une tâche à l'autre qui est effectuée par un ensemble d'algorithmes appelé ordonnanceur (ou scheduler).

Un système d'exploitation classique comme Unix, Linux ou Windows utilise la notion de temps partagé, par opposition au temps réel. Dans ce type de système, le but de l'ordonnanceur est de donner à l'utilisateur une impression de confort d'utilisation tout en assurant que toutes les tâches demandées sont finalement exécutées. Ce type d'approche entraîne une grande complexité dans la structure même de l'ordonnanceur qui doit tenir compte de notions comme la régulation de la charge du système ou la date depuis laquelle une tâche donnée est en cours d'exécution. De ce fait, on peut noter plusieurs limitations par rapport à la gestion du temps. (8)

Tout d'abord, la notion de priorité entre les tâches est peu prise en compte, car l'ordonnanceur a pour but premier le partage équitable du temps entre les différentes tâches du système (on parle de quantum de temps).

Ensuite, les différentes tâches doivent accéder à des ressources dites partagées, ce qui entraîne des incertitudes temporelles. Si une des tâches effectue une écriture sur le disque dur, ce dernier n'est plus disponible pour les autres tâches à un instant donné et le délai de disponibilité du périphérique n'est pas prévisible. (6)

En outre, la gestion des entrées/sorties peut générer des temps morts, car une tâche peut être bloquée en attente d'accès à un élément d'entrée/sortie.

La gestion des interruptions reçues par une tâche n'est pas optimisée. Le temps de latence (soit le temps écoulé entre la réception de l'interruption et son traitement) n'est pas garanti par le système. Par comparaison, le temps de latence dans le cas d'un système temps réel est souvent inférieur à 100 microsecondes.

Enfin, l'utilisation du mécanisme de mémoire virtuelle peut entraîner des fluctuations dans les temps d'exécution des tâches. Toutes ces limitations font que le temps de réponse d'un système classique n'est pas garanti. (6)

Le cas des systèmes temps réel est différent. Il existe un grand nombre de définitions d'un système dit temps réel, et la plupart d'entre elles sont contradictoires. Une définition simple d'un tel système pourra être la suivante :

« Un système est dit temps réel lorsqu'il est soumis à des contraintes de temps et qu'il y répond dans un intervalle acceptable » (9), ou bien :

« Un système temps réel est une association logiciel/matériel où le logiciel permet, entre autres, une gestion adéquate des ressources matérielles en vue de remplir certaines tâches ou fonctions dans des limites temporelles bien précises » (6)

On peut diviser les systèmes en deux catégories : (10)

- Les systèmes dits à contraintes souples (soft real time). Ces systèmes acceptent des variations dans le traitement des données de l'ordre de la demi-seconde (ou 500 ms) ou la seconde. On peut citer l'exemple des systèmes multimédias : si quelques images ne sont pas affichées, cela ne met pas en péril le fonctionnement correct de l'ensemble du système. Ces systèmes se rapprochent fortement des systèmes d'exploitation classiques à temps partagé.
- Les systèmes dits à contraintes dures (hard real time) pour lesquels une gestion stricte du temps est nécessaire pour conserver l'intégrité du service rendu. On citera en guise d'exemples les contrôles de processus industriels sensibles comme la régulation des centrales nucléaires ou les systèmes embarqués utilisés dans l'aéronautique.

Les systèmes à contraintes dures doivent répondre à trois critères fondamentaux : (6)

1. Le déterminisme logique : les mêmes entrées appliquées au système doivent produire les mêmes effets.
2. Le déterminisme temporel : une tâche donnée doit obligatoirement être exécutée dans les délais impartis ; on parle d'échéance.
3. La fiabilité : le système doit être disponible. Cette contrainte est très forte dans le cas d'un environnement embarqué car les interventions d'un opérateur sont très difficiles ou même impossibles. Cette contrainte est indépendante de la notion de temps réel mais la fiabilité du système sera d'autant plus mise à l'épreuve dans le cas de contraintes dures.

En résumé, on peut dire qu'un système temps réel doit être prévisible (prédictible), les contraintes temporelles pouvant s'échelonner entre quelques microsecondes et quelques secondes.

1.4.1.5 Les systèmes d'exploitation temps réels commercialisés

Dans le marché des systèmes d'exploitation temps réel. Il y'en a beaucoup qui fournissent des mécanismes adéquats pour permettre le développement des systèmes temps réel, tels que RT-Linux, LYNX, QNX, etc. le problème majeur de ces systèmes d'exploitation c'est qu'ils offrent un ensemble riche de primitives qui doivent être utilisés avec beaucoup de soin. Ils fournissent habituellement des protocoles de verrouillage des ressources qui peuvent créer des problèmes catastrophiques s'ils sont mal utilisés ou dans le cas des projets très complexes. Dans ces situations il est beaucoup souhaitable d'utiliser des primitives qui contribuent à des synchronisations imprévisibles des systèmes à concevoir. (1)

Il y a aussi un ensemble de Systèmes d'exploitation temps réel qui ont été conçus pour résoudre ces problèmes, l'idée de base est de fournir un ensemble restreint de primitives qui guident les ingénieurs vers une bonne conception de leur système. Asterix et SSX5 sont des exemples de ce type d'OS.

1.4.1.6 Linux

Au sens strict, Linux est le nom du noyau de système d'exploitation libre, multitâche, multi plate-forme et multi-utilisateur de type UNIX, souvent désigné comme le noyau Linux. Il fut initialement développé sur un processeur de type Intel x86 mais il a depuis été adapté sur un grand nombre d'architectures matérielles comme les ARM, PowerPC, SPARC, ETC.

Linux est conforme à la norme Posix[2], ce qui signifie que les programmes développés sous Linux peuvent être recompilés facilement sur d'autres systèmes d'exploitation compatibles Posix.

Linux est également réputé pour sa grande interopérabilité, c'est-à-dire qu'il peut facilement s'intégrer dans un réseau informatique utilisant d'autres systèmes d'exploitation.

Le système d'exploitation Linux est libre, le code source des différents composants du système est disponible gratuitement sur le réseau Internet. Ce même code source peut

[2] Portable Operating System Interface (Voir aussi (63))

être redistribué gratuitement en respectant les règles de la GPL (General Public Licence) et de sa variante, la LGPL (Lesser General Public Licence), elles sont définies par la FSF (Free Software Foundation) pour le projet GNU. Le principe général de la GPL est celui de la conservation de la liberté, chacun devant au moins redistribuer ce qu'il a reçu. (11)

1.4.2 Linux et l'embarqué

Le graphique de la Figure 1-2 indique la répartition de l'utilisation de Linux sur des délais de projets variant de zéro (projets actuels) à deux ans.

Une enquête menée par (12) en 2003 a révélé que 33 pour cent des entreprises ont choisi linux embarqués dans leurs produits en 2001/2002 tandis que 49 pour cent veulent l'utiliser d'ici deux ans (2004/2005). En 2005, 43 pour cent disent avoir utilisé Linux, tandis que 55 pour cent programme son utilisation dans le futur. (12)

L'utilisation de l'Unix dans les systèmes embarqués est en évolution dramatique, et cela est causé par plusieurs critères. (Voir Figure 1-3)

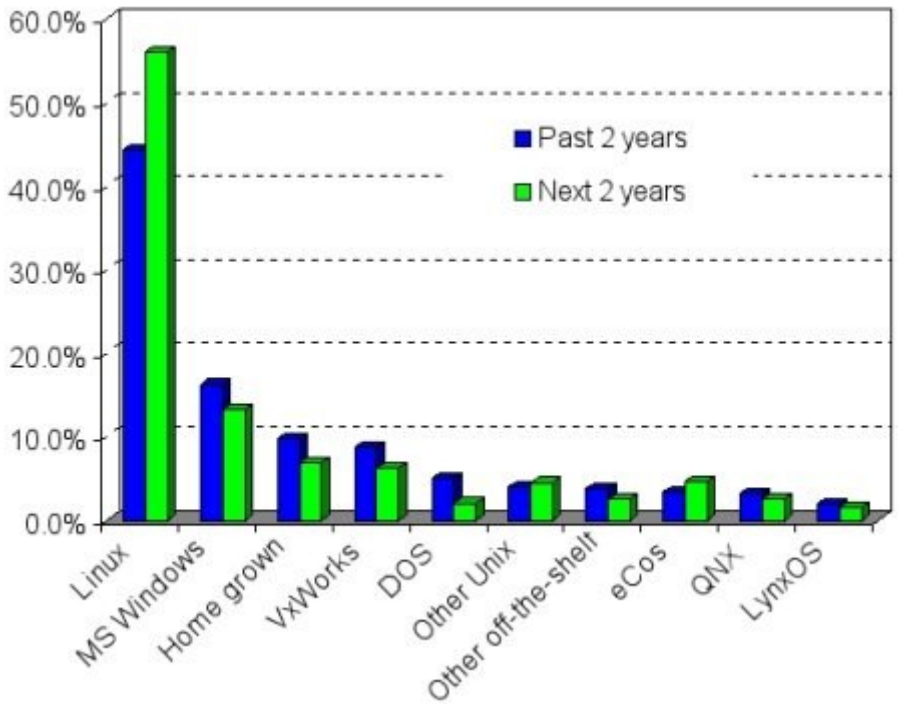


Figure 1-2: Quel système avez-vous choisi pour vos produits embarqués (12)

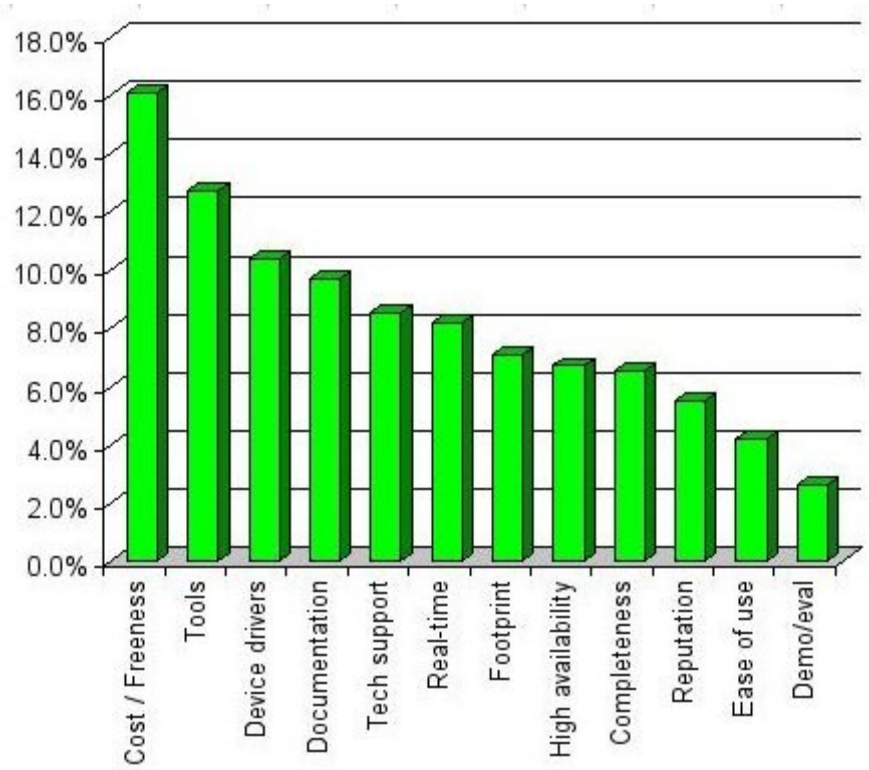


Figure 1-3: quels sont les facteurs qui ont influencé votre choix (13)

1.4.2.1 Fiabilité

Linux est réputé pour sa fiabilité et l'on peut affirmer que cette réputation n'est pas usurpée.

1.4.2.2 Faible coût

La contrainte économique est bien évidemment très importante dans le cas du développement d'un système embarqué. Linux est non seulement exempt de royalties mais les outils de développement sont également disponibles sous GPL. Le seul effort financier nécessaire à l'adoption de Linux se situe sur la formation souvent indispensable et le support technique.

1.4.2.3 Performances

Les performances de Linux ne sont plus à prouver. De nombreux tests comparatifs (benchmarks) entre Linux et d'autres systèmes concurrents comme Windows NT ont démontré la supériorité de Linux.

Cependant, vu la diversité des systèmes embarqués, et pour se rendre compte des capacités de Linux, le mieux est encore de le tester soi-même.

1.4.2.4 Portabilité et adaptabilité

La portabilité est également un des points forts de Linux, il est aujourd'hui porté sur un très grand nombre de processeurs et d'architectures matérielles, y compris des processeurs de faible puissance.

1.4.2.5 Ouverture

De par sa conception Open Source, Linux est ouvert. Ce concept d'ouverture se situe à deux niveaux.

Le premier niveau concerne l'interopérabilité de Linux avec d'autres systèmes d'exploitation.

Le second niveau d'ouverture concerne l'adoption dans Linux des nouvelles technologies, pour peu que celles-ci constituent des standards ouverts. Le fait que Linux soit totalement Open Source facilite l'adaptation et le test de nouveaux protocoles car tous les secrets du système sont à portée de main, du moins si l'on sait où les chercher.

1.4.2.6 Les cas où Linux n'est pas inadapté ?

Il peut exister des cas dans lesquels Linux sera inadapté. Si le système à embarquer nécessite uniquement des fonctions de base n'incluant pas de support réseau ni de multitâche et si cet équipement n'est pas destiné à évoluer, il n'est pas forcément intéressant d'utiliser un système aussi riche que Linux.

1.4.3 Langages de programmation pour les systèmes embarqués

Les langages typiquement utilisés aujourd'hui pour le développement de systèmes embarqués sont : (6)

- l'assembleur : pour des raisons économiques de nombreux systèmes embarqués sont écrits en assembleur. Dans beaucoup d'applications où la complexité logicielle est très faible, l'assembleur permet de réaliser des programmes de taille minimale, donc d'utiliser des composants bon marché ;
- le C : La plupart des systèmes automobiles par exemple sont écrits en C. Ce langage est devenu incontournable dans les années 1990 quand les systèmes électroniques ont gagné en complexité ;

- le C++ : jugé parfois trop complexe pour être totalement maîtrisé par une large communauté de programmeurs et posant des problèmes difficiles de vérification par des outils automatiques, n'a pas réellement percé pour les logiciels embarqués complexes ;
- Ada : Il est utilisé dans beaucoup de systèmes embarqués complexes comme les applications militaires.

1.4.3.1 Langage C

Le langage C (14) a été développé dans les années 1970 comme langage d'implémentation pour un système d'exploitation naissant : Unix. Dérivé d'un langage non typé, le BCPL, il s'est doté d'une structure typée pour devenir rapidement le langage procédural de référence et l'outil dominant des programmeurs pour des décennies.

Au-delà du langage typé de haut niveau, il bénéficie d'opérateurs de bas niveau et d'une proximité avec le modèle d'exécution des processeurs. Cela permet aux programmeurs d'implémenter des algorithmes de façon très efficace et qui restent néanmoins portables.

Les systèmes Unix et leurs outils, qui constituent une part dominante du marché des serveurs, sont généralement écrits en C. Ce sont la proximité du matériel et les opérateurs bas niveau qui ont fait du C le langage de référence dans l'embarqué. La programmation en langage C exige de se concentrer sur l'implémentation d'algorithmes et le flot d'exécution de son programme.

1.4.3.2 Langages orientés objet

Dans les langages orientés objet, le programmeur se focalise en revanche sur les structures de données et les méthodes qui transforment ces structures de données. Le langage apporte une vision plus précise des interactions et des dépendances entre ces structures de données (par exemple, l'héritage). L'orientation objet doit être vue comme un nouveau moyen de décrire ses algorithmes, organisé autour des structures de données, proposant au programmeur de combiner des transformations sur ces structures. En fait, c'est plutôt une nouvelle façon de penser ses applications qu'une technique de programmation.

Dans un langage orienté objet, l'entité principale est la classe : c'est le type des données manipulées, elle représente un lien explicite entre les fonctions/procédures et les structures de données. Les procédures membres sont appelées des méthodes, elles ont le même comportement que les fonctions C mais tirent avantage de l'organisation en classes. Enfin, si la classe représente le type, l'objet en est un spécimen (aussi appelé instance). Les propriétés caractéristiques des langages orientés objet sont multiples, on citera notamment les quatre suivantes, qui apportent beaucoup de souplesse au programmeur : (15)

- mécanisme de désignation : plusieurs méthodes peuvent avoir le même nom et leur comportement varie en fonction de l'objet sur lequel elles s'appliquent ;
- encapsulation : chaque classe peut cacher certains de ses composants, les méthodes privées sont invisibles (donc inutilisables) en dehors de la classe ;
- héritage : mécanisme qui étend ou change le code existant d'une classe « mère » en créant une classe « fille » qui lui est liée. C'est une fonctionnalité clé pour assurer la réutilisabilité des composants ;
- polymorphisme : la classe fille peut être utilisée à la place de la classe mère (le contraire est faux). La même méthode peut exister dans les deux classes et c'est au moment de l'exécution qu'est effectué le choix de la bonne version à employer.

1.4.3.3 Java

Java est un vrai langage orienté objet, créé par la société américaine Sun Microsystems et publié en 1995.

Les exigences suivantes ont servi de base à la définition de ce langage : (15)

- simple : le nombre de constructions du langage est réduit au minimum. La syntaxe Java est proche de celle du C/C++ pour faciliter la migration ;
- distribué : vérifié par la pénétration historique de Java dans le World Wide Web, le langage apporte par exemple le RMI (Remote Method Invocation) pour l'invocation de méthodes distantes ;
- interprété : le compilateur génère du byte-code, pas du code natif. Un interpréteur est nécessaire pour exécuter ce byte-code. Cette architecture facilite le transfert

d'un programme sur des plates- formes d'exécution différentes. Une plate-forme d'exécution, que l'on appelle JVM (Java Virtual Machine), est constituée d'un interpréteur et des bibliothèques de base ;

- robuste : c'est un langage fortement typé qui permet de nombreuses vérifications dès la compilation (problèmes de transtypage par exemple, source courante de bugs). La déclaration explicite des méthodes est obligatoire. L'absence de pointeurs évite les effacements ou corruption de mémoire. Le mécanisme de ramasse-miettes évite les fuites mémoires et les bugs pernicieux dus aux cycles d'allocation et de libération de mémoire. À tout ceci s'ajoutent de nombreuses vérifications effectuées automatiquement à l'exécution, comme la validité d'un index de tableau ;
- protégé : l'absence de pointeur empêche le programmeur de contourner les protections mises en place dans le langage. De plus, le compilateur ne décide pas de l'agencement des objets, effectué par la JVM, donc le programmeur ne peut pas deviner comment les données Java sont stockées en regardant simplement le code source, ce qui rend difficile toute stratégie de contournement par des moyens extérieurs au langage ;
- neutre du point de vue de l'architecture : une application Java doit pouvoir être exécutée sur n'importe quel système muni d'une JVM ;
- portable : en plus de la caractéristique précédente, la portabilité est facilitée par le retrait du langage de tout aspect lié au type de matériel. Par exemple, chaque type de base est défini explicitement : un entier de type Int est codé sur 32 bits, quel que soit le microprocesseur. Cela évite les contorsions auxquelles étaient habitués les programmeurs de systèmes embarqués en C jonglant avec des contrôleurs 8 bits, 16 bits ou 32 bits (la taille d'un Int y est généralement celle du bus) ;
- performant : certes, les applications Java sont généralement moins performantes que leur équivalent C car elles sont interprétées. Cependant, les nouvelles architectures d'exécution comprenant un compilateur JIT (Just In Time : une partie du code est compilée sur la cible avant d'être exécutée, pour être réutilisée

efficacement) ou AOT (Ahead Of Time : une partie du code est compilée au chargement) le rendent comparable au C ;

- multithread (multiples fils d'exécution) : Java fournit un support pour l'exécution de plusieurs threads (classe Thread), ainsi qu'un mot-clé dédié à la synchronisation de ces threads (synchronized) ;
- dynamique : les classes sont chargées par la JVM au moment où elle en a besoin, sans nécessiter l'interruption du flot d'exécution.

1.4.3.4 Java et le temps réel[3]

Très tôt dans l'histoire du langage, l'opportunité d'utiliser Java pour le développement de systèmes temps réel, été discutée au sein de la communauté des développeurs. En effet, un certain nombre de caractéristiques du langage, présentées dans la suite, ne permettent pas son usage immédiat pour la programmation de systèmes temps réel. Dès juin 1998, un groupe de travail RTJWG (Real-Time Java Working Group), comprenant tous les acteurs majeurs du domaine, a été mis en place par le NIST[4] en vue de définir les exigences pour une extension temps réel de Java.

Par la suite, l'opposition entre d'un côté les partisans d'une spécification qui reste contrôlée par l'inventeur du langage, Sun Microsystems, et de l'autre les partisans d'une spécification ouverte de type ISO, a conduit à un véritable schisme.

À l'heure actuelle, deux groupes internationaux, le Real-Time Java Expert Group et le J Consortium, proposent deux spécifications totalement distinctes, respectivement la RTSJ (Real-Time Specification for Java) et le Real-Time Core. Les exigences des deux spécifications sont assez proches, avec cependant pour la RTSJ la volonté de ne pas particulariser cette spécification, notamment en n'apportant aucun changement syntaxique, et du côté du Real-Time Core la volonté de faciliter la création de profils dédiés à certains types d'applications, autorisant ou non la liaison avec des applications Java « classiques ». À côté de cela, la grande majorité des environnements de développement Java dédiés à l'embarqué est basée sur des extensions propriétaires ou

[3] Source : (15)

[4] National Institute of standards and technology

des profils dédiés, tel MIDP (Mobile Information Device Profile) pour les téléphones mobiles, qui ne suivent aucune de ces deux spécifications.

Des travaux sont actuellement en cours pour harmoniser ces approches et donner aux programmeurs de systèmes temps réel le standard dont ils ont besoin.

Le développement d'applications pour l'embarqué évolue. Longtemps dominé par le langage Ada pour les applications critiques et par les solutions propriétaires, axées sur les langages C/C++, le monde de l'embarqué et du temps réel cherche un nouvel environnement de référence, de préférence commun avec le monde des systèmes d'informations, qui pourrait bien être Java. Ce langage, porté par ses atouts indéniables comme la robustesse, la portabilité et surtout sa capacité à coopérer avec les composants existants écrits en C, est en train de faire sa révolution en douceur.

Beaucoup de systèmes embarqués avec une forte contrainte de temps réel ne sont pas encore prêts pour intégrer Java, d'autres domaines d'application comme la téléphonie mobile, les terminaux bancaires ou les systèmes de télésurveillance d'installations industrielles ou domestiques mettent déjà à profit les qualités de Java. La téléphonie mobile en a déjà fait un standard, qui permettra aux portables non seulement d'utiliser des pages Web interactives mais aussi de personnaliser l'interface homme-machine.

Le jour où le langage aura atteint sa maturité pour la gestion d'applications temps réel, il faut également s'attendre à le voir percer par exemple dans les systèmes aéronautiques, déjà intéressés par ses capacités de prototypage facile sur station de travail et l'abondance de programmeurs déjà formés.

1.4.4 Communications temps réels

Aujourd'hui, on trouve des solutions pour surveiller et contrôler un processus depuis un Smartphone, l'un des exemples est l'application ScadaMobile (16) qui permet aux ingénieurs de surveiller et de contrôler les données d'exploitation sur un réseau Ethernet/IP ou Modbus TCP/IP grâce à la carte Wi-Fi intégrée dans le Smartphone.

La communication en temps réel vise à fournir une communication des données entre les périphériques distribués qui soit rapide et déterministe. Beaucoup de domaines nécessitent des communications temps réel, allant des petits réseaux de terrain aux

grandes applications distribuées (Ethernet / Internet). Il y a également un intérêt croissant pour les solutions sans fil.

1.4.4.1 Protocoles temps-réel basés sur Ethernet

Historiquement, les bus de terrain étaient principalement conçus par les fournisseurs d'API (Automates Programmable Industriel). Les spécificités des contraintes de communications industrielles font que les bus de terrain requièrent un haut degré d'expertise. Ils doivent être robustes à l'environnement industriel (couche physique), déterministes afin de garantir le rafraichissement des données dans le temps de cycle (couche liaison de données) et interopérables pour pouvoir échanger des informations entre tous les types d'équipements industriels (couche application). (15)

Ethernet se positionne comme la solution standard pour les communications industrielles en dépit de ses inconvénients intrinsèques (particulièrement sa méthode d'accès au médium non déterministe). Comme pour les réseaux de terrains, chaque fournisseur réseaux fait la promotion de sa solution.

De nombreuses propositions ont été faites pour améliorer les capacités temps-réel des réseaux Ethernet.

La première approche est basée sur la désactivation du protocole d'accès au support CSMA/CD (responsable du non-déterminisme de la norme IEEE 802.3) et sur la mise en œuvre d'autres mécanismes permettant de faire circuler les données sur le réseau dans des tranches de temps prédéterminés (time slots).

La deuxième approche préconise l'introduction des switches, permettant de segmenter les réseaux en domaines exempts de collision, et chargés de distribuer les paquets de données de façon très contrôlée et précise dans le temps.

Enfin, dans la troisième approche, le mécanisme Ethernet et l'infrastructure sont carrément modifiés (ex. EtherCat) (15) (17)

En conséquence, de nombreux produits ont été proposés sur le marché afin de rendre Ethernet déterministe (EtherCAT, Profinet IRT, Modbus/TCP, Ethernet/IP, etc.). Le défi est alors pour les industriels de sélectionner la technologie la plus appropriée tout en conservant un regard sur leur pérennité.

1.4.4.2 Les Communications sans-fil[5]

Il n'existe aucun protocole de communications sans-fil assurant la propriété temps réel. Les protocoles utilisés dans ce type de domaine sont IEEE 802.11 (WLAN) et le Bluetooth. Cependant, ces protocoles ne fournissent pas les garanties temporelles nécessaires pour les communications en temps réel dur.

Il existe d'autres protocoles de communication utilisés pour les réseaux de capteurs qui essayent de minimiser la consommation d'énergie au lieu de garantir la propriété du temps réel. On peut citer comme exemple : LR-WPAN (802.15.4), SMAC (Sonsor MAC), T-MAC (Timeout MAC), BMAC (Berkeley MAC).

1.5 Conclusion

Dans ce chapitre on a présenté l'utilité de l'architecture en couche d'un système embarqué. Ensuite on a présenté un aperçu des tendances de mise en réseau des systèmes embarqués, leur conception, ainsi que certaines applications.

Vu la diversité des systèmes embarqués, engendrant une diversité de contraintes et d'exigences, la conception de ces derniers nécessite des réponses à plusieurs questions, telles que : la nécessité du temps réel, le type du système d'exploitation, les applications finales, etc.

Les concepteurs des systèmes embarqués sont en face à plusieurs choix de systèmes d'exploitation. Les avantage de Linux le rend la première cible pour la plus part des concepteurs.

Aussi, au niveau Application, les concepteurs sont en face à un large éventail de langages de programmation, allant de ceux mono-plate-forme (tel que le langage C) aux applications multi-plate-forme (tel que le langage Java). Le choix du langage dépend principalement du domaine d'application. Un bon choix du langage de programmation engendre une bonne économie que ce soit en temps qu'en argent.

Les supports de communication sont aussi une exigence très importante à prendre en considération lors de la conception des systèmes embarqués. On doit toujours rechercher

[5] Pour plus de détail Voir (64)

les meilleures solutions pour les problèmes de sécurité et du temps réel, tout en conservant les exigences du domaine d'application.

Chacun des types des systèmes embarqués doit être étudié séparément, et doit avoir une solution sur mesure pour une performance élevé et une sécurité accrue.

2 Cryptographie

2.1 Introduction

Un réseau peut être sécurisé en utilisant les dispositifs physiques, le contrôle d'accès AAA[vi], l'installation des pare-feu et l'implémentation d'IPS[vii]. Ces solutions combinées protègent les dispositifs d'infrastructure aussi bien que les dispositifs terminaux dans un réseau local. Mais comment le trafic réseau est-il protégé en traversant l'Internet public ?

La cryptologie est la science de chiffrement et de déchiffrement des codes secrets. Le développement et l'utilisation des codes s'appellent la cryptographie, et le déchiffrement des codes s'appelle la cryptanalyse. La cryptographie a été employée pendant des siècles pour protéger les documents secrets. Aujourd'hui, des méthodes cryptographiques modernes sont employées afin d'assurer des communications sécurisées.

Une communication sécurisée exige une garantie que le message vient réellement de la source énoncée (authentification). Elle exige également une garantie que personne n'a changé le message (intégrité). Enfin, la communication sécurisée assure que si le message est intercepté, il ne peut pas être déchiffré (confidentialité).

Les principes de la cryptologie peuvent être employés pour expliquer comment les protocoles et les algorithmes modernes sont employés pour protéger les communications. Beaucoup de réseaux assurent l'authentification avec des protocoles tels que HMAC, l'intégrité en mettant en application MD5 ou SHA-1, La confidentialité de données par les algorithmes de chiffrement symétriques, y compris le DES, le 3DES, et l'AES, ou les algorithmes asymétriques, y compris RSA et l'infrastructure à clé publique (PKI). Des algorithmes de chiffrement symétriques sont basés sur le fait que chacune des parties communicantes connaît la clé partagée. Des algorithmes de chiffrement asymétriques sont fondés sur l'hypothèse que les deux parties de

[vi] En sécurité informatique, AAA correspond à un protocole qui réalise trois fonctions : l'authentification, l'autorisation, et la traçabilité (en anglais : Authentication, Authorization, Accounting/Auditing). AAA est un modèle de sécurité implémenté dans certains routeurs Cisco mais que l'on peut également utiliser sur toute machine qui peut servir de NAS (Network Access Server).

[vii] Intrusion Protection Système

communication n'ont pas précédemment partagé un secret et doivent établir une méthode sûre pour communiquer.

2.2 Protection des communications

Le premier but pour des administrateurs réseau est de protéger l'infrastructure réseau, y compris les routeurs, les commutateurs, les serveurs, et les hôtes. Ceci est effectué en utilisant des dispositifs physiques, le contrôle d'accès AAA, pare-feu, et surveiller les menaces en utilisant l'IPS.

Le prochain but est de protéger les données car elles sont véhiculées à travers divers liens. Ceci peut inclure le trafic interne, mais le plus grand souci est de protéger les données qui sont échangées en dehors de l'organisation.

Les communications sécurisées impliquent quelques tâches primaires :

- Authentification - les garanties que le message n'est pas une contrefaçon et vient réellement de l'émetteur légitime.
- Intégrité - semblable à une fonction de somme de contrôle dans une trame de données, garantit que personne n'a intercepté le message ou l'a changé.
- Confidentialité - garanties que si le message est intercepté, il ne peut pas être déchiffré.

2.2.1 L'Authentification

L'Authentification garantit qu'un message vient de la source laquelle il prétend venir.
L'authentification peut être accomplie avec des méthodes cryptographiques. C'est particulièrement important pour des applications ou des protocoles, comme l'email.
La non-répudiation de données est un service semblable qui permet à l'expéditeur d'un message d'être uniquement identifié. Avec des services de non-répudiation en place, un expéditeur ne peut pas nier avoir été la source de ce message. La caractéristique la plus importante de la non-répudiation est qu'un dispositif ne peut pas nier la validité d'un message envoyé. La non-répudiation se fonde sur le fait que seulement l'expéditeur a les caractéristiques ou la signature unique pour la façon dont ce message est traité. Même le dispositif de réception ne peut savoir la façon dont l'expéditeur a traité ce message pour prouver l'authenticité.

2.2.2 L'intégrité

L'intégrité des données s'assure que les messages n'ont pas été changés en transit. Avec l'intégrité des données, le récepteur peut vérifier que le message reçu est identique au message envoyé et qu'aucune manipulation ne s'est produite.

2.2.3 La Confidentialité

La confidentialité de données assure l'intimité de sorte que seulement le récepteur puisse lire le message. Le cryptage est le processus de brouiller des données de sorte qu'il ne puisse pas être lues par les parties non autorisées. En appliquant le cryptage, les données lisibles s'appellent le texte en clair, alors que celles chiffrées s'appellent le cryptogramme. Le message lisible est converti en cryptogramme, qui est illisible. Le décryptage renverse le processus. Une clé est exigée pour chiffrer et déchiffrer un message. La clé est le lien entre le texte clair et le cryptogramme.

L'utilisation d'une fonction de hachage est une autre manière d'assurer la confidentialité de données.

La différence entre le hachage et le chiffrement est dans la façon dont les données sont stockées. Avec le texte chiffré, les données peuvent être déchiffrées avec une clé. Avec la fonction de hachage, après que les données soient saisies et converties en utilisant la fonction de hachage, le texte clair est perdu. Les données hachées sont simplement là pour la comparaison.

Le but du chiffrement est de garantir la confidentialité de sorte que seulement les entités autorisées puissent lire le message.

2.3 Cryptographie

L'authentification, l'intégrité, et la confidentialité sont des composants de cryptographie. La cryptographie est la pratique et l'étude de dissimulation d'information. Les services cryptographiques sont la base pour beaucoup de réalisations de sécurité et sont employés pour assurer la protection des données quand ces données s'exposent aux parties non autorisées. La compréhension des fonctions de base de la cryptographie et les notions de la confidentialité et l'intégrité est importante pour créer une politique de sécurité réussie. Il est également important de comprendre les questions qui sont impliquées dans la gestion de la clé de cryptage.

De diverses méthodes de chiffrement, et des dispositifs physiques ont été employés pour chiffrer et déchiffrer le texte, chacune de ces méthodes de chiffrage emploie un algorithme spécifique, appelé l'algorithme de cryptage, pour chiffrer et déchiffrer des messages. Un algorithme de cryptage est une série d'étapes bien définies qui peuvent être suivies comme procédure en chiffrant et en déchiffrant des messages.

2.4 Cryptanalyse

Tant qu'il y a la cryptographie, il y aura la cryptanalyse. La cryptanalyse est la pratique et l'étude de détermination de la signification d'information chiffrée (déchiffrer le code), sans accès à la clé secrète partagée.

Un grand choix de méthodes est employé dans la cryptanalyse.

2.4.1 Attaque par force brute

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.

2.4.2 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.

2.4.3 Attaque à texte clair connu

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

2.4.4 Attaque à texte clair choisi

L'attaquant choisit lui-même le message à coder. 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.

2.4.5 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., 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.

2.5 La cryptologie

La cryptologie est la science de chiffrement et de déchiffrement des codes secrets. La cryptologie combine deux disciplines distinctes de la cryptographie, qui est le développement et l'utilisation des codes, et la cryptanalyse, qui est la rupture de ces codes. Chacune des disciplines est la symbiose de l'autre, parce que chacune rend l'autre meilleur.

Tandis que la cryptanalyse est souvent liée à des fins malhonnêtes, c'est devenu actuellement une nécessité. Sans elle, c'est impossible de prouver la sûreté d'un algorithme cryptographique.

Dans le monde des communications et de la mise en réseau, l'authentification, l'intégrité, et la confidentialité de données sont mises en application en utilisant de divers protocoles et algorithmes. Le choix du protocole et de l'algorithme varie selon le niveau de la sécurité exigé pour atteindre les buts dans la politique de sécurité de réseau.

Par exemple, pour l'intégrité de message, l'algorithme MD5 est plus rapide mais moins sûr que le SHA2. La confidentialité peut être mise en application en utilisant le DES, le 3DES, ou l'AES. Encore, le choix varie selon les exigences de sécurité définies dans le document de politique de sécurité à implémenter.

De vieux algorithmes de chiffrement, tels que celui de César ou la machine d'Enigma, ont été basés sur le secret de l'algorithme pour assurer la confidentialité. Avec la technologie moderne, où le reverse engineering est souvent simple, les algorithmes de domaine publique sont souvent employés. Avec la plupart des algorithmes modernes, le décryptage réussi exige la connaissance des clés cryptographiques appropriées. Ceci signifie que la sécurité repose sur la capacité à maintenir les clés de chiffrement secrètes, avec un algorithme qui soit publique.

2.6 Hachage cryptographique

Une fonction de hachage prend des données binaires, appelées le message, et produit une représentation condensée, appelée le résumé de message. Le brouillage est basé sur une fonction mathématique à sens unique qui est relativement facile à calculer, mais par contre plus dur à renverser. La fonction de hachage cryptographique est conçue pour vérifier et assurer l'intégrité des données. Elle peut également être employée pour vérifier l'authentification. La procédure prend un bloc variable de données et renvoie une chaîne binaire de longueur constante appelée la valeur de hachage ou le résumé de message.

La fonction de hachage cryptographique est appliquée dans différentes situations :

- Pour fournir la preuve de l'authenticité quand elle est employée avec une clé secrète symétrique d'authentification, telle que la sécurité d'IP (IPsec) ou l'authentification du protocole de routage.
- Pour fournir l'authentification en produisant des réponses à sens unique lors des échanges dans des protocoles d'authentification tels que le protocole PPP[viii] et CHAP[ix].
- Pour fournir une preuve de contrôle de l'intégrité de message, comme ceux utilisés dans les contrats digitalement signés, et les certificats de l'infrastructure à clé publique (PKI), comme ceux admis en accédant à un site sécurisé utilisant un navigateur.

Mathématiquement, une fonction de hachage (H) est un processus qui prend une entrée (x) et renvoie une chaine de caractères à taille fixe, qui s'appelle la valeur de hachage (h). La formule pour le calcul est h = H (x).

[viii] Point to Point Protocol
[ix] Challenge Handshake Authentication Protocol

2.7 Intégrité avec MD5 et SHA-1

L'algorithme MD5 est un algorithme de hachage qui a été développé par Ron Rivest et est employé dans un grand nombre d'applications Internet aujourd'hui. MD5 est une fonction à sens unique qui rend facile le calcul d'une hache des données en entrée. MD5 est également résistant aux collisions, ce qui signifie qu'il est très peu susceptible de produire deux messages avec la même hache. MD5 est essentiellement une séquence complexe des opérations binaires simples, telles que le OU exclusif (XOR), qui sont effectuées sur des données d'entrée et produisent une hache de 128 bits.

MD5 est basé sur MD4, un algorithme plus ancien. MD4 a été cassé, et MD5 est maintenant considéré moins sûr que SHA-1 par beaucoup d'autorités de cryptographie. Ces autorités considèrent MD5 moins sûr parce que quelques faiblesses non critiques ont été trouvées dans un des blocs MD5 constitutifs.

Le National Institute of Standards and Technology des États-Unis (NIST) a développé un algorithme de hachage sûr (SHA), l'algorithme qui est spécifié dans la norme sûre de hachage (SHS). SHA-1, édité en 1994, a corrigé une faille non publiée dans SHA. Sa conception est très semblable aux fonctions de hachage MD4 et MD5 que Ron Rivest a développé.

L'algorithme SHA-1 prend un message de longueur un peu moins que 2^64 bits et produit un résumé de message de 160 bits. L'algorithme est légèrement plus lent que MD5, mais le résumé de message plus grand le rend plus sûr contre des attaques de collision et d'inversion par force brute.

Le NIST a édité quatre fonctions de hachage supplémentaires dans la famille de SHA, chacune avec des résumés de messages de plus en plus longs:

- SHA-224 (224 bit)
- SHA-256 (256 bit)
- SHA-384 (384 bit)
- SHA-512 (512 bit)

Ces quatre versions sont collectivement connues comme SHA-2, bien que le terme SHA-2 ne soit pas normalisé. SHA-1, SHA-224, SHA-256, SHA-384, et SHA-512 sont les algorithmes de hachage sûrs exigés par loi pour l'usage dans certaines applications

de gouvernement des États-Unis, y compris l'utilisation dans d'autres algorithmes cryptographiques et protocoles, pour la protection d'informations sensibles non secrètes. MD5 et SHA-1 sont basés sur MD4. Ceci rend MD5 et SHA-1 semblables de plusieurs manières. SHA-1 et SHA-2 sont plus résistants aux attaques par force brute parce que leur résumé est plus long que le résumé MD5par au moins 32 bits.

2.8 Authenticité avec HMAC

Dans la cryptographie, un code d'authentification de message par hachage à clé (HMAC ou KHMAC) est un type de code d'authentification de message (MAC). Un HMAC est calculé en utilisant un algorithme spécifique qui combine une fonction de hachage cryptographique avec une clé secrète. Les fonctions de hachage sont la base du mécanisme de protection de HMACs.

Seulement l'expéditeur et le récepteur connaissent la clé secrète, et la production de la fonction de hachage dépend des données d'entrée et de la clé secrète. Seulement les parties qui ont accès à cette clé secrète peuvent calculer le résumé d'une fonction de HMAC. Cette caractéristique défait les attaques man-in-the-middle et fournit l'authentification d'origine de données.

Si deux parties partagent une clé secrète et emploient des fonctions de HMAC pour l'authentification, un résumé correctement construit de HMAC d'un message qu'une partie a reçu indique que l'autre partie était le créateur du message, parce qu'il est la seule entité possédant la clé secrète.

La force cryptographique du HMAC dépend de la force cryptographique de la fonction de hachage sous-jacente, sur la taille et la qualité de la clé, et de la taille du message de hachage en bits. Quand un résumé de HMAC est créé, des données d'une longueur arbitraire sont entrées dans la fonction de hachage, ainsi qu'une clé secrète. Le résultat est une hache de longueur constante qui dépend des données et de la clé secrète. Le soin doit être pris pour distribuer des clés secrètes seulement aux parties qui sont impliquées parce que, si la clé secrète est compromise, l'autre partie peut forger et changer des paquets, violant l'intégrité de données.

Les réseaux privés virtuels d'IPsec (VPNs) se fondent sur des fonctions de HMAC pour authentifier l'origine de chaque paquet et pour fournir la vérification d'intégrité des données.

Les ponts et les clients d'IPsec emploient des algorithmes de hachage, tels que MD5 et SHA-1 dans le mode de HMAC, pour fournir l'intégrité de paquet et l'authenticité.
Les signatures digitales sont une alternative à HMAC.

2.9 Chiffrement

Le chiffrement peut fournir la confidentialité à plusieurs couches du modèle OSI en incorporant divers outils et protocoles :

- Les dispositifs de chiffrement fournissent la confidentialité de la couche liaison de données.
- Les protocoles de couche réseau, tels que l'ensemble de protocoles d'IPsec, fournissent la confidentialité de couche réseau.
- Les protocoles comme (SSL) ou Transport Layer Security (TLS) fournissent la confidentialité de la couche session.
- L'email sûr, la session sûre de base de données, et la transmission de messages sûre fournissent la confidentialité de couche application.

Il y a deux approches pour assurer la sécurité des données quand on emploie diverses méthodes de chiffrement. La première est de protéger l'algorithme. Si la sécurité d'un système de chiffrement est basée sur le secret de l'algorithme lui-même, le code d'algorithme doit être précieusement gardé. Si l'algorithme est indiqué, chaque partie impliquée doit changer l'algorithme. La deuxième approche est de protéger les clés. Avec la cryptographie moderne, tous les algorithmes sont publics. Les clés cryptographiques assurent le secret des données. Les clés cryptographiques sont des séquences de bits qui sont entrés dans un algorithme de chiffrement avec les données à chiffrer.

Deux classes de base des algorithmes de chiffrement protègent les clés : symétrique et asymétrique. Chacun diffère dans son utilisation des clés. Les algorithmes de chiffrement symétriques emploient la même clé, appelée la clé secrète, pour chiffrer et déchiffrer des données. La clé doit être pré-partagée. Une clé pré-partagée est connue par l'expéditeur et le récepteur avant que toute communication chiffrée ne débute. Des longueurs de clés plus courtes signifient une exécution plus rapide. Les algorithmes symétriques demandent généralement beaucoup moins de calculs que les algorithmes asymétriques.

Les algorithmes de chiffrement asymétriques emploient différentes clés pour chiffrer et déchiffrer les données. Des messages sûrs peuvent être échangés sans devoir avoir une clé pré-partagée. Puisque les deux parties n'ont pas un secret partagé, des longueurs de clés très longues doivent être employées pour contrecarrer les attaquants. Ces algorithmes requièrent plus de ressources et leur exécution est plus lente. Dans la pratique, les algorithmes asymétriques sont typiquement des centaines à des milliers de fois plus lents que les algorithmes symétriques.

Le chiffrement symétrique ou à clé secrète, est la forme de cryptographie la plus utilisée généralement, parce que la longueur de clé plus courte augmente la vitesse de l'exécution. En plus, les algorithmes à clé symétrique sont basés sur des opérations mathématiques simples qui peuvent être facilement accélérées par le matériel. Le chiffrage symétrique est employé souvent pour le chiffrage dans des réseaux informatiques quand la confidentialité des données est exigée, comme pour protéger un VPN.

Avec le chiffrage symétrique, la gestion des clés peut être un défi. Les clés de chiffrage et de déchiffrage sont identiques. L'expéditeur et le récepteur doivent échanger la clé symétrique et secrète en utilisant un canal sûr avant que n'importe quel chiffrage puisse se produire. La sécurité d'un algorithme symétrique repose sur le secret de la clé symétrique. En obtenant la clé, n'importe qui peut chiffrer et déchiffrer des messages.

Le DES, le 3DES, l'AES, l'algorithme de chiffrement de logiciel (SEAL), et les séries des algorithmes de Rivest (RC), qui incluent RC2, RC4, RC5, et RC6, sont tous des algorithmes de chiffrement bien connus qui emploient des clés symétriques. Il existe beaucoup d'autres algorithmes de chiffrement, tels que le Blowfish, le Twofish, le Threefish, et le Serpent. Cependant, ces protocoles ne sont pas supportés par des plates-formes matérielles ou n'ont pas encore gagné une large acceptation.

Les techniques les plus utilisées généralement dans la cryptographie à chiffrement symétrique sont des chiffrements par bloc et des chiffrements par flux.

2.9.1 Les chiffrements par bloc

Les chiffrements par bloc transforment un bloc de longueur constante de texte clair en bloc commun du cryptogramme de 64 ou 128 bits. La longueur de bloc se rapporte à combien de données sont chiffrées en même temps. Actuellement la longueur de bloc,

également connue sous le nom de longueur fixe, parce que beaucoup de chiffrements par bloc est 64 bits ou 128 bits. La longueur principale se rapporte à la taille de la clé de cryptage qui est employée. Ce cryptogramme est déchiffré en appliquant la transformation inverse au bloc de cryptogramme, utilisant la même clé secrète.

Les chiffrements par bloc ont habituellement comme données de sortie des conséquences qui sont plus grandes que les données d'entrée, parce que le cryptogramme doit être un multiple de la longueur de bloc. Par exemple, le DES chiffre des blocs dans de gros morceaux de 64-bit utilisant une clé de 56 bits. Pour accomplir ceci, l'algorithme par bloc prend un gros morceau de données à la fois, par exemple, 8 octets chaque morceau, jusqu'à ce que la longueur de bloc entière soit pleine. S'il y a moins de données d'entrée que celles d'un bloc plein, l'algorithme ajoute des données artificielles (blancs) jusqu'à ce que les 64 bits soient employés.

Les chiffrements par bloc commun incluent le DES avec une longueur de bloc 64 bits, l'AES avec une longueur de bloc de 128, 192 ou 256 bits et le RSA avec une longueur de bloc variable.

2.9.2 Chiffrements par flux

À la différence des chiffrements par bloc, les chiffrements par flux chiffrent le texte clair un octet ou un bit à la fois. Des chiffrements de flux peuvent être considérés comme chiffrement par bloc avec une longueur de bloc d'un bit. Avec un chiffrement de flux, la transformation de ces plus petites unités de texte clair varie, selon l'étape où sont produits pendant le procédé de chiffrage. Les chiffrements par flux peuvent être beaucoup plus rapides que les chiffrements par bloc, et généralement n'augmentent pas la taille de message, parce qu'ils peuvent chiffrer un nombre arbitraire de bits. Les chiffrements par flux communs incluent A5, qui est employé pour chiffrer des communications de téléphone portable de GSM, et le chiffrement de RC4. Le DES peut également être employé dans le mode de chiffrement de flux. Le choix d'un algorithme de chiffrement est l'une des décisions les plus importantes qu'un professionnel de sécurité prend en établissant un système cryptographique.

Deux critères principaux devraient être considérés en sélectionnant un algorithme de chiffrement:

L'algorithme est reconnu par la communauté cryptographique. La plupart des nouveaux algorithmes sont cassés très rapidement, ainsi les algorithmes qui avaient résisté à des attaques pendant un certain nombre d'années sont préférés. Les inventeurs et les instigateurs vendent trop souvent les avantages de nouveaux algorithmes.

L'algorithme se protège adéquatement contre des attaques par force brute. Un bon algorithme cryptographique est conçu de telle manière qu'il résiste à des attaques cryptographiques communes. La meilleure manière de casser les données qui sont protégées par l'algorithme est d'essayer de déchiffrer les données utilisant toutes les clés possibles. L'algorithme doit laisser les longueurs des clés qui répondent aux exigences de confidentialité d'une organisation. Par exemple, le DES n'assure pas assez de protection pour les besoins les plus modernes en raison de sa clé courte.

D'autres critères à considérer :

L'algorithme soutient des longueurs de clé variables et l'évolutivité. Les longueurs variables des clés et l'évolutivité sont également des attributs souhaitables d'un bon algorithme de chiffrement. Plus la clé de cryptage est longue, plus il est dur pour un attaquant de la casser. Par exemple, une clé de 16 bits a 65.536 clés possibles, mais une clé de 56 bits a 7,2 x 10^16 clés possibles. L'évolutivité fournit une longueur flexible de clé et permet à l'administrateur de sélectionner la force et la vitesse du chiffrement exigé.

2.9.3 Quelques Algorithmes Symétriques : Data Encryption Standard

Data Encryption Standard (DES) est un algorithme de chiffrement symétrique qui fonctionne généralement en mode bloc. Il chiffre les données en blocs de 64 bits. L'algorithme DES est essentiellement une séquence de permutations et de substitutions de bits de données associés à une clé de cryptage. Le même algorithme et clé sont utilisés pour le chiffrement et le déchiffrement.

DES a une longueur de clé fixe. La clé est de 64 bits de longueur, mais seulement 56 bits sont utilisés pour le chiffrement. Les 8 bits restants sont utilisés pour la parité. Le bit le moins significatif de chaque octet de la clé est utilisé pour indiquer la parité impaire.

Bien que DES utilise généralement le mode de chiffrement par bloc, il peut également crypter en utilisant le mode de chiffrement par flux.

En raison de sa longueur de clé courte, DES est considéré comme un bon protocole pour protéger les données pendant un temps très court. 3DES est un meilleur choix pour protéger les données. Il dispose d'un algorithme qui est très grand et a une résistance plus grande aux attaques.

Avec les progrès technologique des calculateurs informatiques, le DES avec ses clés courtes est devenue incapable de résister aux attaques. Une façon d'augmenter la longueur des clés, sans modifier l'algorithme lui-même, est d'utiliser le même algorithme avec des clés différentes plusieurs fois.

La technique d'application DES trois fois pour un bloc de texte est appelé 3DES. Aujourd'hui, les attaques en force sur 3DES sont considérés comme irréalisables parce que l'algorithme de base a été bien testé depuis plus de 35 ans et il est considéré comme très fiable.

3DES utilise une méthode appelée 3DES-Encrypt-Decrypt-Encrypt (3DES-EDE). Tout d'abord, le message est chiffré en utilisant la première clé de 56-bit, appelé K1. Ensuite, les données sont déchiffrées en utilisant la seconde clé 56-bit, appelé K2, et enfin, les données sont chiffrées à nouveau, en utilisant la troisième clé de 56-bit, appelé K3.

La procédure 3DES-EDE est beaucoup plus efficace pour augmenter la sécurité que de crypter simplement les données trois fois avec trois clés différentes.[x]

Pour déchiffrer le message, la méthode 3DES-EDE est utilisée aussi. Tout d'abord, le texte chiffré est déchiffré en utilisant la clé K3. Ensuite, les données sont chiffrées en utilisant la clé K2 et enfin, les données sont déchiffrées en utilisant la clé K1.

Bien que 3DES est sûr, il est aussi très gourmand en ressources. Pour cette raison, l'algorithme de chiffrement AES a été développé. Il est aussi sûr que 3DES, mais beaucoup plus rapide et moins gourmand.

2.9.4 Quelques Algorithmes Symétriques : RC

Les algorithmes RC ont été conçus par Ronald Rivest, qui a aussi inventé MD5. Les algorithmes RC sont largement déployés dans les applications embarqués souffrants d'une limite en ressources.

Il y a plusieurs algorithmes RC largement utilisés:

[x] Voir CISCO Security, Chapitre 6

RC2 : chiffrement par blocs avec une taille de clé variable, il a été conçu pour remplacer le DES.

RC4 : c'est un algorithme de chiffrement par flux. Il utilise une taille de clé variable. Il est utilisé dans les communications sécurisées, comme le SSL.

RC5 – Un algorithme de chiffrement par bloc avec une taille variable (que ce soit pour les blocs que pour les clés). RC5 peut être utilisé comme une solution pour remplacer le DES.

RC6 - Développé en 1997, RC6 a été finaliste AES. C'est un algorithme de chiffrement par bloc (de 128 ou 256bits) avec une clé de 128 à 256 bits et qui repose sur le RC5. Son but principal de conception était de répondre aux exigences de l'AES.

2.9.5 Quelques algorithmes asymétriques : RSA

Le RSA a été inventé par Rivest, Shamir et Adleman en 1978. C'est l'exemple le plus courant de cryptographie asymétrique, toujours considéré comme sûr avec la technologie actuelle, pour des clés suffisamment grosses (1024, 2048 voire 4096 bits). D'ailleurs le RSA128 (algorithme avec des clés de 128 bits), proposé en 1978 par Rivest, Shamir et Adleman, n'a été 'cassé' qu'en 1996, en faisant travailler en parallèle de nombreux ordinateurs sur internet.

Mais le concept de chiffrement asymétrique avec une clé publique était légèrement antérieur (1976). L'idée générale était de trouver deux fonctions f et g sur les entiers, telles que g(f)=Id, et telle qu'on ne puisse pas trouver f, la fonction de décryptage, à partir de g, la fonction de cryptage. On peut alors rendre publique la fonction g (ou clé), qui permettra aux autres de crypter le message à envoyer, tout en étant les seuls à connaitre f, donc à pouvoir décrypter. (18)

2.9.6 Chiffrements Symétriques VS Asymétriques

Un avantage important des chiffrements asymétriques par rapport aux chiffrements symétriques est qu'aucun canal secret n'est nécessaire pour pratiquer l'échange de la clé publique. Le récepteur doit simplement être sûr de l'authenticité de la clé publique. Les chiffrements symétriques demandent un canal secret pour envoyer la clé secrète (générée d'un côté du canal de communication) vers l'autre côté.

Les chiffrements asymétriques créent aussi moins de problèmes de gestion de clés que les chiffrements symétriques. 2n clés seulement sont nécessaires pour que n entités communiquent en toute sécurité entre elles. Dans un système basé sur des chiffrements symétriques, il faudrait n (n – 1)/2 clés secrètes. Dans une entreprise de 5 000 employés, par exemple, le déploiement généralisé d'une solution de sécurité basée sur le cryptage symétrique exigerait plus de 12 millions de clés. Avec une solution asymétrique, il n'en faudrait que 10 000. (19)

L'inconvénient des chiffrements asymétriques par rapport aux chiffrements symétriques est qu'ils tendent à être environ « 1000 fois plus lents ». Autrement dit, il faudra plus de 1000 fois plus de temps de CPU pour traiter un cryptage ou décryptage asymétrique par rapport au symétrique.

En raison de ces caractéristiques, les chiffrements asymétriques sont généralement utilisés pour l'authentification de données (par l'intermédiaire de signatures numériques), pour la distribution d'une clé de cryptage symétrique en masse (aussi appelée enveloppe numérique), pour des services de non-répudiation, et pour un accord de clé. Les chiffrements symétriques, cependant, sont plutôt réservés au cryptage de masse.

2.9.7 Crypto-système à courbes Elliptiques(ECC)

Avec le large déploiement des systèmes embarqués mobiles limités en ressources et leur utilisation dans des applications cryptiques comme e-banking, e-commerce et l'accès à distance à des informations de sociétés (tel que l'utilisation des ERP via un Smartphone) il y a eu une croissance importante de la quantité des données sensibles échangées sur Internet. Cela a créé un besoin d'utilisation des mécanismes de sécurité efficace et des protocoles qui peuvent opérer sur des réseaux sans-fil ou filaire.

La majorité des protocoles de sécurité Internet (tels que SSL/TLS, IPSec) emploient des crypto systèmes asymétriques pour échanger les clés de chiffrement symétrique et utilisent des algorithmes symétriques rapides de chiffrement par bloc (tels que AES, 3DES) pour assurer la confidentialité, l'intégrité et l'authentification.

RSA est le crypto système asymétrique le plus utilisé (20). Pour casser une clé symétrique, il faut casser le crypto système à clé publique utilisé pour échanger cette

clé, et pour renforcer le crypto système à clé publique, les tailles de clé symétrique et asymétrique doivent augmenter. NIST affiche la croissance des tailles de clé symétriques et leur équivalence dans les crypto systèmes à clé publique : (Tableau 2-1)

Tableau 2-1: Comparaison des tailles des clés nécessaire pour une sécurité similaire (21)

Algo à Clés Symétriques	Algo à Clés Asymétriques (RSA)	ECC
80	1024	160
112	2048	224
128	3072	256
192	7680	384
265	15360	512

Dans quelques mémoires Tel que (22), les auteurs ont mentionné que, le cryptosystème ECC a l'avantage d'offrir des tailles de clés plus petites comparées par des cryptosystèmes traditionnels, et par conséquent, selon ces études, les cryptosystèmes ECC sont plus adaptables aux systèmes embarqués qui sont limités en ressources.

Cela n'est pas toujours vrai, parce que selon des études profondes sur les cryptosystèmes ECC (tel que (23) et (24) , le RSA est vraiment plus long que les ECC dans la génération des clés (pour le RSA 1024 et plus) (voir Tableau 2-2 et Tableau 2-3), ce qui n'est pas le cas des systèmes embarqués à ressources limitées (comme les Smartphones et les capteurs), mais dans le cas de ces appareils qui ne génèrent pas des clés, il sera préférable d'utiliser le RSA, puisque selon ces études, le RSA est comparable à ECC (en terme de temps) dans la création de la signature numérique et plus rapide que les ECC pour la vérification de la signature (ce qui est vraiment nécessaire dans le cas du mobile-banking).

Tableau 2-2: : Performance lors de la génération des clés (23)

Génération de la clé	**Longueur de la clé**		**Temps (ms)**	
	ECC	RSA	ECC	RSA
	163	1024	0.085	0.16
	233	2240	0.18	7.47
	283	3072	0.27	9.80
	409	7680	0.64	133.90
	571	15360	1.44	676.06

Tableau 2-3: Performance lors de la génération de la signature (23)

Longueur de clé		**Temps (ms)**	
ECC	RSA	ECC	RSA
163	1024	0.15	0.01
233	2240	0.34	0.15
283	3072	0.59	0.21
409	7680	1.18	1.53
571	15360	3.07	9.20

Tableau 2-4: Performance lors de la vérification de la signature (23)

Longueur de clé		Temps (ms)	
ECC	RSA	ECC	RSA
163	1024	0.23	0.01
233	2240	0.51	0.01
283	3072	0.86	0.01
409	7680	1.80	0.01
571	15360	4.53	0.03

2.9.8 Signature numérique

La signature numérique[xi] est un mécanisme permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur, par analogie avec la signature manuscrite d'un document papier. La signature électronique n'est devenue possible qu'avec la cryptographie asymétrique. Elle se différencie de la signature écrite par le fait qu'elle n'est pas visuelle, mais correspond à une suite de nombres. (25)

En plus d'assurer l'authenticité et l'intégrité des messages, les signatures numériques sont utilisées pour fournir l'assurance de l'authenticité et de l'intégrité des codes de logiciels mobiles et classique. Les fichiers exécutables, ou éventuellement l'ensemble de l'installation de paquets d'un programme, sont enveloppés d'une enveloppe numériquement signé, ce qui permet à l'utilisateur de vérifier la signature avant d'installer le logiciel.

[xi] Parfois appelée signature électronique

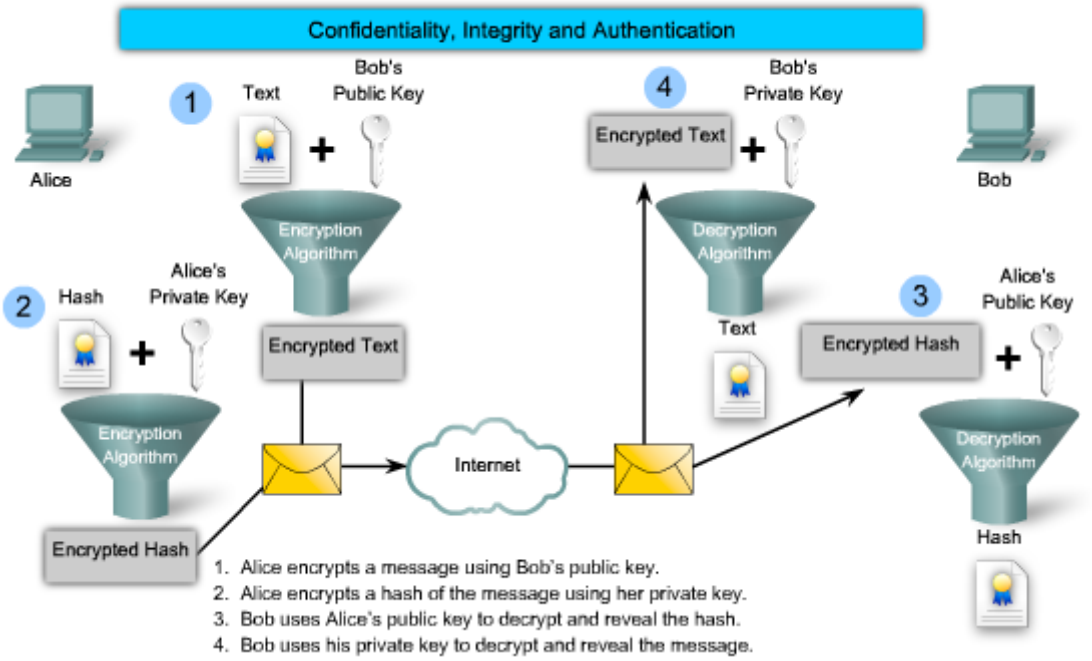


Figure 2-1: Signature numérique (26)

La signature numérique fournit plusieurs assurances concernant le code:

- Le code n'a pas été modifié depuis qu'il a quitté l'éditeur du logiciel.
- Le code est authentique et est effectivement compilé par l'éditeur.

L'utilisateur du logiciel doit également obtenir la clé publique, qui est utilisée pour vérifier la signature. Par exemple, la clé peut être inclue dans l'installation du système d'exploitation.

2.10 Advanced Encryption Standard (AES)

2.10.1 Historique

Depuis l'invention par Biham et Shamir de la cryptanalyse différentielle, le DES commençait à ne plus offrir de garantie de sécurité suffisante. Ainsi, le NIST (National Institut of Standards and Technology) a lancé en 1997 un appel d'offres pour remplacer le DES et élire un algorithme de chiffrement à clé secrète «capable de protéger la confidentialité des informations pour le XXIe siècle», tant pour le gouvernement américain que pour le secteur privé.

L'appel d'offres stipulait que le remplaçant de DES (où AES pour Advanced Encryption Standard) devait définir un algorithme public de chiffrement à clé secrète par blocs, libre de droits et utilisable dans le monde entier. La taille des blocs ainsi que celle des clés

étaient spécifiées : le NIST demandait de pouvoir chiffrer des blocs de 128 bits avec des clés de taille 128, 192 ou 256 bits.

En 1998, le NIST avait reçu quinze propositions. Le 2 octobre 2000, le NIST a annoncé le vainqueur. Le nouveau standard de chiffrement ou AES est Rijndael, ainsi nommé par ses concepteurs Daemen et Rijmen, deux chercheurs Belges.

2.10.2 Spécifications d'AES

Rijndael est un chiffre itéré de taille de blocs et de clés de 128, 192 ou 256 bits.

Les différentes transformations utilisées opèrent sur un résultat intermédiaire appelé état qu'on représente par un tableau d'octets à deux dimensions, 4 lignes et Nb colonnes. (27)

Transformations de tour

Avant le premier tour, on effectue une opération « Ou exclusif » avec les bits de la clé de tour (AddRoundKey).

A chaque tour, en fonction d'une clé de tour obtenue au moyen d'un algorithme de séquencement qui sera décrit plus loin, on effectue les opérations suivantes (voir Figure 2-2) sur l'état :

SubBytes : C'est une fonction non-linéaire opérant indépendamment sur chaque bloc à partir d'une table dite de substitution.

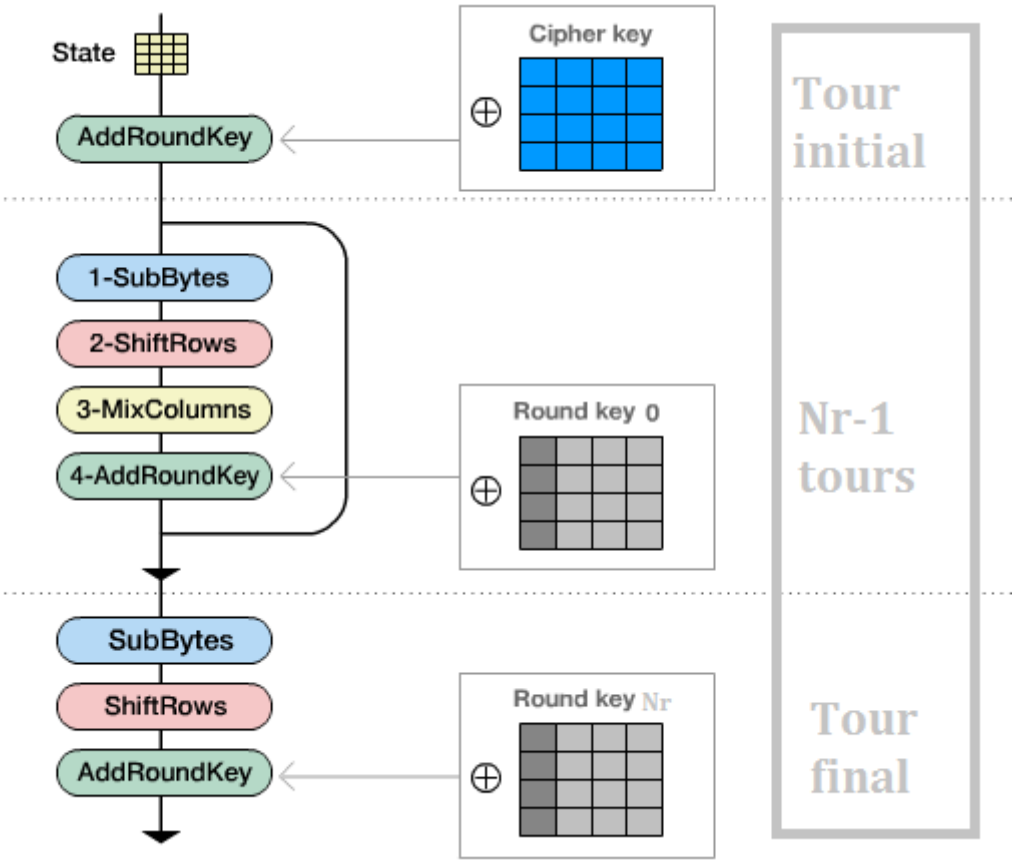


Figure 2-2: Algorithme AES (28)

ShiftRows : C'est une fonction opérant des décalages (typiquement elle prend l'entrée en 4 morceaux de 4 octets et opère des décalages vers la gauche de 0, 1, 2 et 3 octets pour les morceaux 1, 2, 3 et 4 respectivement).

MixColumns : C'est une fonction qui transforme chaque octet d'entrée en une combinaison linéaire d'octets d'entrée et qui peut être exprimée mathématiquement par un produit marticiel sur le corps de Galois (28).

AddRoundKey : On effectue un XOR entre les bits de l'état et ceux de la clé de tour.

2.10.3 Algorithme de séquencement de clé

Les différentes clés de tour sont obtenues par dérivation de la clé principale au moyen de l'algorithme de séquencement de la clé qui se déroule en deux étapes :

- une phase d'expansion de la clé ;
- une phase de sélection de la clé de tour.

Le principe de base de l'algorithme est le suivant :

Le nombre de bits pour la clé de tour est égal à la longueur du bloc multiplié par « le nombre de tours plus un » (par exemple, pour une taille de bloc de 128 bits et 10 tours, il faudra 1408 bits) ;

La clé principale est étendue en une clé étendue ;

Les clés de tour sont issues de la clé étendue de la façon suivante : la première clé de tour consiste en les Nb premiers mots de la clé étendue, la seconde en les Nb suivants et ainsi de suite.

2.10.4 Déchiffrement

Rijndael a été conçu pour fonctionner de la même manière au chiffrement qu'au déchiffrement en remplaçant chaque opération élémentaire par son inverse.

2.10.5 Attaques

Attaques sur des versions simplifiées: Des attaques existent sur des versions simplifiées d'AES. Niels Ferguson et son équipe ont proposé en 2000 une attaque sur une version à 7 tours de l'AES 128 bits. Une attaque similaire casse un AES de 192 ou 256 bits contenant 8 tours. Un AES de 256 bits peut être cassé s'il est réduit à 9 tours avec une contrainte supplémentaire. En effet, cette dernière attaque repose sur le principe des « related-keys » (clés apparentées). Dans une telle attaque, la clé demeure secrète mais l'attaquant peut spécifier des transformations sur la clé et chiffrer des textes à sa guise. Il peut donc légèrement modifier la clé et regarder comment la sortie de l'AES se comporte.

Attaques sur la version complète : Certains groupes ont affirmé avoir cassé l'AES complet mais après vérification par la communauté scientifique, il s'avérait que toutes ces méthodes étaient erronées. Cependant, plusieurs chercheurs ont mis en évidence des possibilités d'attaques algébriques, notamment l'attaque XL et une version améliorée, la XSL. Ces attaques ont été le sujet de nombreuses controverses et leur efficacité n'a pas encore été pleinement démontrée, le XSL fait appel à une analyse heuristique dont la réussite n'est pas systématique. De plus, elles sont impraticables car le XSL demande au moins 287 opérations voire 2100 dans certains cas. Le principe est d'établir les équations

(quadratiques / booléennes) qui lient les entrées aux sorties et de résoudre ce système qui ne comporte pas moins de 8.000 inconnues et 1.600 équations pour 128 bits.

2.10.6 Conclusion sur l'algorithme AES

La conception de Rijndael a été inspirée par les attaques linéaires et différentielles ainsi que par des attaques utilisant des manipulations algébriques (par interpolation) et motivée par les critères techniques que les auteurs ont détaillé dans leur livre.

La cryptanalyse de ce système reste pour l'instant impossible à déterminer. En l'absence d'une preuve formelle sur l'efficacité d'attaques similaires au XSL, l'AES est donc considéré comme sûr. On peut toutefois parier que dans les années à venir, les avancées en cryptanalyse et la relative simplicité de la structure d'AES devraient ouvrir des brèches dans l'algorithme. Si une découverte pareille se produira, des méthodes similaires à AES comme Camellia pourraient rapidement devenir obsolètes.

2.11 Conclusion

La sécurité des communications requiert : L'intégrité, l'authentification et la confidentialité. La cryptologie est composée de deux disciplines : La cryptographie qui est liée à la recherche et l'utilisation des méthodes cryptographiques, tandis que la cryptanalyse recherche des méthodes pour les casser.

Dans le cas des systèmes informatiques[xii], on utilise le chiffrement symétrique pour chiffrer les données et l'asymétrique pour chiffrer les clés de chiffrement symétrique.

Pour une bonne sécurité d'un système informatique, il faut toujours essayer d'implémenter des schémas cryptographiques testés ainsi que des protocoles de sécurité standards.

[xii] Que ce soit embarqué ou à usage général

3 Sécurité dans les systèmes embarqués

3.1 Introduction

Dans les dernières années, il y a eu une tendance croissante pour l'utilisation des systèmes embarqués. Plusieurs de ces systèmes sont reliés à d'autres réseaux et sont ainsi inclus dans des LAN, des WAN, et dans l'Internet. Par conséquent, ces systèmes sont exposés aux attaques potentielles de sécurité, qui peuvent compromettre leur intégrité et causer des dommages considérables.

Les ressources limitées des systèmes embarqués causent un défi important pour l'implémentation des politiques de sécurité efficaces qui, sont généralement exigeantes en ressources.

Pour une bonne étude de la sécurité dans les systèmes embarqués, il faut d'abord cerner ses domaines d'application. Pour cette raison, ce chapitre concerne les systèmes embarqués dont ils correspondent à la définition suivante :

« Un système informatique est typiquement considéré comme un système embarqué quand il s'agit d'un dispositif programmable avec des ressources limitées (énergie, mémoire, puissance de calcul, etc.) qui sert à une ou plusieurs applications et est embarqué dans un plus grand système. Leurs ressources limitées les rendent inefficaces pour être employées en tant que systèmes informatique à usage universel. Cependant, ils doivent habituellement répondre à des exigences dures, telles que les conditions de traitement en temps réel. »

Les systèmes embarqués peuvent être classifiés dans deux catégories générales :

1. systèmes embarqués autonomes, où tous les composants (le matériel et le logiciel) du système sont physiquement étroits et incorporés dans un seul dispositif, par exemple, un PDA, et il n'y a aucun attachement à un réseau.
2. systèmes embarqués répartis (gérés en réseau), où plusieurs composants autonomes (Où chacun étant lui-même un système embarqué) communiquent les

uns avec les autres au-dessus d'un réseau afin de fournir des services ou soutenir une application. (29)

L'augmentation des capacités des systèmes embarqués combinée à leur coût décroissant a permis leur adoption dans une large gamme d'applications et de services, des applications financières aux militaires. Ainsi, en plus des conditions typiques pour la réponse, la fiabilité, la disponibilité, la robustesse, et l'extensibilité, beaucoup des systèmes embarqués ont des exigences de sécurité importantes.

3.2 Paramètres de sécurité

La sécurité est un terme générique utilisé pour désigner plusieurs exigences différentes dans les systèmes informatiques.

De nos jours, elle est devenue un problème majeur dans l'informatique. La transmission d'informations sensibles et le désir d'assurer la confidentialité de celles-ci est devenue un point primordial dans la mise en place des infrastructures informatiques.

Selon le système et son utilisation, plusieurs propriétés de sécurité peuvent être satisfaites dans chaque système et dans chaque environnement opérationnel. Globalement, les systèmes de sécurité doivent satisfaire à tous ou à un sous-ensemble des exigences suivantes (30) (31): La Confidentialité, l'Intégrité, la Non-répudiation, la Disponibilité, l'Authentification et le Contrôle d'accès.

Vu les exigences de la sécurité, la conception d'un système sécurisé exige l'identification et la définition des paramètres suivants : les capacités des attaquants, le niveau auquel la sécurité devrait être implémentée, et la technologie mise en œuvre et l'environnement opérationnel. (31)

3.2.1 Capacités des attaquants

Les utilisateurs malveillants peuvent être classés en plusieurs catégories, en fonction de leurs connaissances, de leurs équipements, etc. Une proposition d'une classification en trois catégories, selon leurs connaissances, leurs équipements hardware-software, et leurs fonds est donnée comme suit: (32)

Classe I - Les étrangers intelligents (clever outsiders). Des attaquants très intelligent, pas très bien financés et sans équipement sophistiqué. Ils n'ont pas de connaissances

spécifiques sur le système à attaquer; fondamentalement, ils tentent d'exploiter les vulnérabilités matérielles et problèmes logiciels.

Classe II - Les bien-informés (knowledgeable insiders) Les attaquants ayant des connaissances techniques remarquables, en utilisant des équipements hautement sophistiqués et, souvent, avec des informations de l'intérieur du système à attaquer. Le groupe des attaquants peut même contenir d'anciens employés qui ont participé au cycle de développement du système.

Classe III - Les financés par des organisations (funded organizations) : Les attaquants qui travaillent la plupart du temps en équipes, et ont d'excellentes qualifications techniques et des fonds théoriques. Ils sont bien financés, ont accès à des outils très avancés et ont également les possibilités pour analyser le système (techniquement et théoriquement), développant ainsi des attaques sophistiquées. Ces organisations pourraient être des fondations d'éducation bien organisées, des institutions gouvernementales, etc.

3.2.2 Niveaux d'implémentation de la sécurité

La sécurité peut être implémentée à différents niveaux du système, allant de la protection physique à la sécurité des applications et du réseau. De différents mécanismes doivent être employés pour implémenter la sécurité à différents niveaux.

Généralement les niveaux de la sécurité considérés sont quatre : (31)

1. Physique.
2. matériel (hardware).
3. logiciel (software).
4. sécurité du réseau et des protocoles.

Les mécanismes de sécurité physiques ciblent à protéger les systèmes contre tout accès physique non autorisé au système lui-même. La protection physique des systèmes assure la confidentialité et l'intégrité des données et des applications.

Les mécanismes de sécurité physique sont considérés réussis quand ils s'assurent qu'une attaque possible aura peu de chance d'aboutir et la possibilité élevée de suivre l'attaquant malveillant, dans un temps raisonnable.

L'adoption à grande échelle des systèmes informatiques embarqués dans une variété de périphériques tels que les cartes à puce, les Smartphones et les réseaux de capteurs, ainsi que la capacité à les mettre en réseau, par exemple, par l'Internet, a mené à la révision et à la reconsidération de la sécurité physique.

La Sécurité du Matériel peut être considérée comme un sous-ensemble de la sécurité physique, se référant aux questions de la sécurité concernant les parties matérielles d'un système informatique.

Au niveau matériel, les attaques exploitent des vulnérabilités des circuits et de la technologie et profitent des défauts du matériel. Ces attaques n'exigent pas nécessairement des équipements très sophistiqués et chers.

Plusieurs manières d'attaquer des cartes à puce et des microcontrôleurs sont possibles, grâce à l'utilisation des tensions inhabituelles et des températures qui affectent le comportement des pièces du matériel spécifique (33). Des mécanismes spéciaux de protection matérielle sont nécessaires pour éviter ces types d'attaques; Ces mécanismes comprennent des revêtements de la puce en silicone, la complexité dans la disposition des composants, etc.

Un des objectifs majeurs dans la conception de systèmes sécurisés est le développement des logiciels sécurisés, qui sont exempts de défauts et des failles de sécurité qui peuvent apparaître sous certaines conditions. De nombreuses failles de sécurité logicielles ont été identifiées dans les systèmes réels, d'ailleurs, il y a eu plusieurs cas où les intrus malveillants piratent des systèmes par l'exploitation des défauts logiciels (34).

L'utilisation de l'Internet, qui est un réseau qui n'est pas sûre pour le transfert de l'information, comme étant un réseau de base pour les entités de communication, et le large déploiement des réseaux sans fil mettent l'accent sur les protocoles de sécurité existant qui doivent être étudiés pour chaque type de système embarqué et trouver les améliorations qui doivent être faites dans les architectures des protocoles existants afin de fournir de nouveaux protocoles sécurisés. De tels protocoles assureront l'authentification entre les entités communicantes, l'intégrité des données communiquées, la protection des parties communicantes, et la non-répudiation.

En outre une attention particulière doit être accordée à la conception de protocoles de sécurité pour les systèmes embarqués, en raison de leurs contraintes physiques, qui sont : l'énergie limitée, la capacité de traitement limitée, les ressources en mémoire, ainsi que leur coût et les besoins de communication.

3.2.3 Technologie d'implémentation et environnement opérationnel

Les systèmes embarqués peuvent être classifiés selon leur technologie en systèmes statiques et systèmes programmables et selon leur architecture en systèmes fixes et systèmes extensibles. (35) Quand la technologie statique est employée, les fonctions mises en œuvre par le matériel sont fixes et inflexibles, mais elles offrent une plus haute performance et peuvent réduire le coût. Cependant, les systèmes statiques peuvent être plus vulnérables aux attaques, parce que, une fois qu'une faille est identifiée, par exemple, dans la conception du système, il est impossible de patcher les systèmes déjà déployés, particulièrement dans le cas de grandes installations, telles que les cartes SIM pour la téléphonie mobile, ou les cartes TV. Les systèmes statiques devraient être mis en application seulement une fois et correctement, ce qui n'est jamais le cas.

En revanche, les systèmes programmables ne sont pas limités comme ceux statiques, mais ils peuvent être plus flexible entre les mains d'un pirate, ainsi, la flexibilité de système peut permettre à un attaquant de manœuvrer le système d'une manière non prévue ou non définie par le concepteur. La programmation est typiquement réalisée par l'utilisation du logiciel spécialisé au-dessus d'un matériel ou un processeur à usage universel.

Les architectures fixes se composent de matériel spécifique qui ne peut pas être changé. Typiquement, il est presque impossible d'ajouter des fonctionnalités, mais elles sont moins coûteuses dans l'implémentation et sont, généralement moins vulnérables parce qu'elles offrent des choix limités aux attaquants. Une architecture extensible est comme un processeur d'usage universel, capable de se connecter par interface à plusieurs périphériques par des raccordements normalisés. Les périphériques peuvent être changés ou améliorés facilement pour augmenter la sécurité ou pour fournir de nouvelles fonctionnalités. Cependant, un attaquant peut relier au système des périphériques malveillants ou connecter le système à un autre, fournissant des cas non essayés ou

inattendus. Comme les tests dans les systèmes statiques sont plus difficiles, on ne peut pas être tout à fait sûr que le système fonctionne correctement sous chaque entrée possible.

3.3 Contraintes de sécurité dans la conception des systèmes embarqués

La conception des systèmes sécurisés exige des considérations spéciales, parce que les fonctions de sécurité sont exigeantes en ressources, notamment en termes de puissance de traitement et de la consommation d'énergie. Les ressources limitées des systèmes embarqués requièrent des approches de conception nouvelle, afin de faire un équilibre entre l'efficacité (la vitesse et le coût) et la satisfaction des exigences fonctionnelles et opérationnelles.

3.3.1 Énergie

Les systèmes embarqués sont souvent alimentés par des batteries, c.-à-d. ils sont limités en énergie. Ainsi, la capacité de la batterie constitue une importance majeure pour le traitement de la sécurité dans les systèmes embarqués. Malheureusement, les améliorations des capacités des batteries ne suivent pas les améliorations de la performance, de la complexité, et de la fonctionnalité croissantes des systèmes qu'elles alimentent. (36)

Ainsi, le sous-système d'alimentation des systèmes embarqués est un point faible de la sécurité de ces systèmes. Par exemple, un attaquant malveillant peut former une attaque DoS en drainant la batterie du système plus rapidement que d'habitude.[xiii]

L'inclusion des fonctions de sécurité dans un système embarqué impose des exigences supplémentaires sur la consommation d'énergie due à: La puissance de traitement supplémentaire nécessaire pour effectuer diverses fonctions de sécurité, telles que l'authentification, le chiffrement, le déchiffrement, etc., La transmission de données liées à la sécurité entre les différentes entités (si le système est distribué, comme le cas

[xiii] Par exemple : un utilisateur malveillant peut demander à plusieurs reprises de l'appareil à exécuter une application gourmande en énergie.

des réseaux de capteurs sans fil). Et l'énergie nécessaire pour stocker des paramètres liés à la sécurité.

Ainsi, l'inclusion des algorithmes cryptographiques dans la conception embarquée peut mener à la grande consommation de l'énergie du système.

En outre, beaucoup d'études ont montré que, dans un réseau sans fil de capteurs, les fonctions de sécurité consomment beaucoup d'énergie à cause de l'échange supplémentaire entre les nœuds des informations de chiffrement (les clés d'échange, les informations d'authentification).

3.3.2 Puissance de traitement

Le traitement de la sécurité (Security processing) est un terme utilisé pour indiquer l'effort du système de calcul qui est dédié à la mise en œuvre des exigences de sécurité. Puisque les systèmes embarqués sont limités dans la capacité de traitement, une étude rigoureuse doit être faite face à l'exécution des algorithmes cryptographiques complexes, qui sont utilisés dans la conception de la sécurité d'un système embarqué.

L'adoption des systèmes embarqués modernes dans les systèmes à haute gamme tels que les serveurs, les pare-feu, et les routeurs, et l'augmentation des taux de transmission des données et la complexité des protocoles de sécurité, tels que le SSL, mettent l'accent sur la sécurité dans les architectures embarquées existantes et démontrent que parmi ces architectures embarquées y'en a celles qui doivent être améliorées, afin de suivre les exigences.

3.3.3 Flexibilité et disponibilité

La conception et l'implantation de la sécurité dans un système embarqué ne signifie pas que le système ne changera pas ses caractéristiques de sécurité dans le temps. Bien que les exigences de sécurité évoluent et que les protocoles de sécurité soient constamment renforcés, les systèmes embarqués doivent être flexibles et adaptables aux changements des exigences de sécurité, sans perdre leurs objectifs de performance, ni de disponibilité, ni leurs objectifs de sécurité primaire.

Un autre point qui doit être abordé est l'implémentation de différentes exigences de sécurité à différentes couches de l'architecture de protocole. L'un des exemples est un

Smartphone qui doit pouvoir exécuter plusieurs protocoles de sécurité, tels que : IPSec, SSL, et WPA, selon l'application spécifique.

3.3.4 Coût d'implémentation

L'inclusion de la sécurité dans la conception des systèmes embarqués peut augmenter le coût de ces systèmes considérablement. Le problème provient des limitations fortes des ressources des systèmes embarqués, alors que le système doit fournir de grandes performances ainsi qu'un niveau de sécurité élevé tout en conservant un faible coût d'implémentation.

Le choix du niveau de sécurité influence le coût de la conception et de la mise en œuvre de manière significative; ainsi, le fabricant est confronté à un compromis entre les exigences de sécurité à implanter et le coût de fabrication.

Il est nécessaire de faire une analyse soigneuse et détaillée du système à concevoir, en termes de capacités des attaquants, des conditions environnementales dans lesquels le système va fonctionner, etc., afin d'estimer le coût de façon réaliste.

3.4 Conception des systèmes embarqués sécurisés

Sécuriser des systèmes embarqués, nécessite de fournir des propriétés de sécurité de base, tels que l'intégrité des données. Dans cette section, nous décrivons les principaux problèmes de conception au niveau système et au niveau application.

3.4.1 Problèmes de conception au niveau système

La conception des systèmes embarqués sécurisés doit se pencher sur plusieurs questions et paramètres allant de la technologie du matériel utilisé jusqu'aux méthodologies de développement logiciel.

Bien que plusieurs techniques utilisées dans le développement des systèmes à usage général puissent effectivement être aussi bien employées dans le développement des systèmes embarqués, il y a des problèmes de conception spécifiques qui doivent être traitées séparément, parce qu'ils sont uniques ou plus faibles dans les systèmes embarqués. Les principaux problèmes de conception sont : la sécurité des communications, et la conception des logiciels embarqués.

Une des approches de sécurité consiste à construire des coprocesseurs embarqués dédiés à la cryptographie. Le coprocesseur SafeXcel (37)qui peut s'intégrer aux architectures ARM est un exemple de cette approche. Intel (38) (39) a annoncé une nouvelle génération de processeurs embarqués 64 bits qui ont certaines caractéristiques qui peuvent accélérer le traitement des algorithmes gourmands (hungry algorithms), tels que les algorithmes cryptographiques ; ces dispositifs incluent de plus grands ensembles de registres, l'exécution parallèle des calculs, et des améliorations dans la multiplication de grands nombres entiers, etc.

Les logiciels embarqués, tels que le système d'exploitation ou les applications spécifiques, constituent un facteur crucial dans la conception du système embarqué sécurisé. Trois facteurs fondamentaux rendent le développement des logiciels embarqués un domaine de sécurité très difficile: la complexité du système, l'extensibilité du système, et la connectivité. (40)

La mémoire limitée des systèmes embarqués, en particulier le manque d'espace disque et la mémoire virtuelle, rendent le système vulnérable en cas d'applications gourmandes en mémoire cela peut facilement provoquer une indisponibilité du système (out of memory).

3.4.2 Problèmes de conception au niveau application

Les applications pour les systèmes embarqués présentent des défis importants pour les concepteurs, afin de parvenir à des systèmes efficaces et sûrs. Une question clé dans la conception des systèmes embarqués sécurisés est l'identification des utilisateurs et le contrôle d'accès. La croissance explosive des appareils mobiles et leur utilisation critique, dans des transactions sensibles, telles que les transactions bancaires, e-commerce, etc., demande des systèmes sécurisés à haute performance et à faible coût. Un système embarqué doit stocker des informations qui lui permettent d'identifier et de valider les utilisateurs qui ont accès au système.

La question est : comment un système embarqué peut emmagasiner cette information ? Les systèmes embarqués utilisent plusieurs types de mémoire pour stocker différents types de données: (1) EPROM pour stocker des données de programmation utilisées pour servir des applications génériques, (2) RAM pour stocker les données temporaires,

et (3) EEPROM et mémoires FLASH pour stocker des codes mobiles téléchargeables (41)

La nécessité de protéger cette information ainsi que la demande croissante en communication (par exemple, l'accès mobile à Internet), rendent les systèmes embarqués vulnérables aux attaques via les réseaux, conduisant à une demande croissante d'espace de stockage sécurisé.

L'utilisation des algorithmes de chiffrement les plus puissants pour assurer l'intégrité et la confidentialité des données doit être étudiée dans la plupart des systèmes embarqués, principalement en raison de leurs ressources de calcul limitées.

3.5 Cryptographie et systèmes embarqués

Les systèmes embarqués sécurisés devraient soutenir les fonctions de sécurité de base qui sont : la confidentialité, l'intégrité, et la non-répudiation. La Cryptographie fournit un mécanisme qui assure que les trois conditions précédentes soient remplies. Cependant, la mise en œuvre de la cryptographie dans les systèmes embarqués peut être une tâche difficile à cause de l'exigence de hautes performances, alors qu'elle va être réalisée dans un environnement à ressources limitées. L'augmentation de la performance cause habituellement une augmentation des coûts, ce qui n'est pas toujours souhaitable.

D'autre part, les protocoles cryptographiques modernes ne reposent pas sur un algorithme cryptographique spécifique, mais permettent plutôt l'utilisation d'un large éventail d'algorithmes pour une sécurité accrue et une adaptabilité à des avances de la cryptanalyse. Par exemple, les protocoles réseau SSL et IPSec soutiennent de nombreux algorithmes de chiffrement pour effectuer la même fonction, et le protocole permet la négociation des algorithmes à utiliser, afin de s'assurer que les deux parties utilisent le niveau de sécurité souhaitable.

D'autre part, la communauté cryptographique a concentré ses efforts sur la preuve théorique de la sécurité de divers algorithmes de cryptographie et a donné peu d'attention à des implémentations réelles sur des plateformes matérielles spécifiques. Ce manque de communication entre les fournisseurs et les cryptographes s'intensifie dans le cas des

systèmes embarqués, ce qui peut entraîner plusieurs choix de conception difficile à implémenter.

3.6 Conclusion

La sécurité constitue une exigence importante dans les systèmes modernes de l'informatique embarquée. Leur utilisation très répandue dans des services comportant des informations sensibles, en conjonction avec leurs ressources limitées ont conduit à un nombre important d'attaques qui exploitent les caractéristiques innovantes de ces systèmes et entraînent une perte d'informations critiques. Le développement des systèmes embarqués sécurisés est un domaine émergent dans l'ingénierie informatique nécessitant des compétences dans la cryptographie, dans la technologie de communication, etc.

Dans ce chapitre, nous avons mentionné les exigences de sécurité des systèmes informatiques embarqués et nous avons décrit les technologies qui sont plus critiques par rapport aux systèmes informatiques à usage générale.

4 Travaux Connexes

4.1 Introduction

L'utilisation des systèmes embarqués se multiplie d'une façon rapide, et sont devenus indispensables dans les dernières années (ex. utilisation des Smartphones).

Malheureusement, la sécurité des systèmes embarqués peut ne pas atteindre nos attentes et suivre le rythme de développement des fonctionnalités de ces systèmes.

Dans cette partie nous présentons quelques études concernant la sécurisation des systèmes embarqués.

4.2 R Venugopalan & P Ganesan. (42)

Les auteurs présentent deux contributions. Tout d'abord, une enquête sur les exigences de traitement et de calcul pour les algorithmes cryptographiques les plus utilisés sur des architectures embarqués. L'objectif de leur travail est de couvrir une large classe d'algorithmes cryptographiques et déterminer l'impact des architectures embarquées sur leur performance afin d'aider les concepteurs à prévoir les performances de leurs systèmes vis-à-vis des tâches cryptographiques pouvant être exécutées. Le deuxième point abordé par leur étude (deuxième contribution), concerne les méthodes pour calculer le temps de calcul des architectures embarquées pour les algorithmes cryptographiques. Cela permet de projeter des limitations de calcul et de déterminer le seuil de schémas de chiffrement possibles sous un ensemble de contraintes pour une architecture embarquée.

4.2.1 Algorithmes

Dans leur étude, les auteurs ont choisi d'évaluer des algorithmes cryptographiques symétrique ainsi que des fonctions de hachage (Tableau 4-1, 4-2) qui font partie de la plus part des protocoles de sécurité (ex. Protocole IP Sec).

Tableau 4-1: Paramètre et schémas cryptographiques (42)

Algorithm	Type	|key/hash|	|Block|
RC4 [2]	stream	128 bits	8 bits
IDEA [2]	block	128 bits	64 bits
RC5 [1]	block	64 bits	64 bits
MD5 [2][3]	1-way hash	128 bits	512 bits
SHA1 [4]	1-way hash	128 bits	512 bits

4.2.2 Plates-formes Hardware

Les auteurs ont évalué la performance des fonctions cryptographiques sur cinq différents processeurs embarqués, allant des processeurs de faible fréquence, aux processeurs à haute performance.

Tableau 4-2: Plates-formes Hardware (42)

Platform	Wordsize	clockfreq.	cache
Atmega 103	8 bits	4 MHz	none
Atmega 128	8 bits	16 MHz	none
M16C/10	16 bits	16 MHz	none
SA-1110	32 bits	206 MHz	16/8KB
PXA250	32 bits	400 MHz	32/32KB
UltraSparc2	64/32 bits	440 MHz	16/16KB

4.2.3 Analyses

Les auteurs ont présenté des résultats de temps d'exécution des différentes mesures concernant les différentes architectures et ont développé un model approximatif concernant le temps d'exécution applicable aux autres architectures des microcontrôleurs.

4.2.3.1 Méthode expérimentale

Les auteurs ont exécuté les mêmes algorithmes sur les différentes architectures, en modifiant la taille des données en entrée (pour chaque algorithme). Ils ont répété chaque test 1000 fois avec la même entrée et ont choisi la valeur moyenne comme référence.

4.2.4 Evaluation des performances

La figure et le tableau suivants (Figure 4-1 et Tableau 4-3) montrent les résultats obtenus par leurs tests

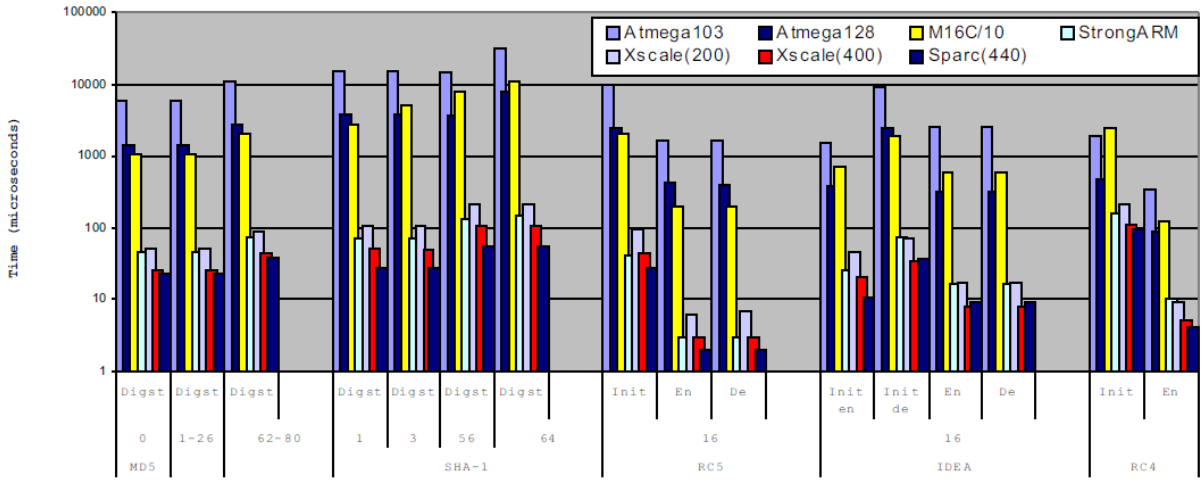


Figure 4-1: Temps d'exécution pour chaque algorithme

Tableau 4-3: Temps d'exécution pour chaque algorithme

Algorithm	Size	Action	Atmega103	Atmega128	M16C/10	StrongARM	Xscale(400)	Xscale(200)	Sparc(440)
MD5	0	Digest	5863	1466	1083	46	26	53	23
	1-26	Digest	5890	1473	1075	46	26	53	23
	62-80	Digest	10888	2722	2011	74	45	90	39
SHA-1	1	Digest	15249	3812	2651	69	51	102	27
	3	Digest	15781	3945	5303	69	50	103	27
	56	Digest	14543	3636	7955	133	102	205	55
	64	Digest	31107	7777	10907	145	103	207	56
RC5	16	Init	9641	2410	2074	41	45	91	28
		Enc	1651	413	197	3	3	6	2
		Dec	1636	409	202	3	3	7	2
IDEA	16	Init enc	1523	381	727	26	21	47	11
		Init dec	9417	2354	1927	76	35	69	36
		Enc	2555	325	596	16	8	17	9
		Dec	2614	325	597	16	8	17	9
RC4		Init	1886	472	2455	155	108	216	96
		Enc	344	86	123	10	5	9	4

4.2.4.1 Explication des résultats

Les auteurs ont remarqué une différence de temps de calcul (Même après la normalisation des horloges), précisément, les architecture avec une taille des mots

mémoires plus grande sont les plus rapide, et cela est à cause de l'utilisation des opérations (des algorithmes cryptographiques) qui nécessitent de larges mots mémoires.

Aussi, les auteurs ont constaté les limitations de calcule des algorithmes cryptographiques dans les systèmes embarqués, et cela est à cause de leurs architectures.

4.3 P. Kocher & al (43)

Les auteurs ont présenté la nécessité de sécuriser les systèmes embarqués par de différentes approches, et ont présenté les défis associés à la sécurisation de ces systèmes.

Ils ont essayé de fournir une vue unifiée de la sécurité des systèmes embarqués en analysant d'abord les exigences de sécurité du point de vue de l'utilisateur final. Ils ont identifié alors les défis concernant les architectes des systèmes embarqués.

Ils ont aussi étudié les techniques pour faire face à ces défis, et ont présenté les modifications architecturales nécessaires à la sécurisation de ces systèmes.

4.3.1 Mécanismes de sécurité

Les auteurs ont présenté quelques mécanismes pour assurer le niveau de sécurité souhaité. Ils ont présenté les différentes solutions suivantes : Algorithmes symétriques, Algorithmes de Hachage, Algorithmes Asymétriques, Protocoles de communication sécurisés, etc.

Les auteurs ont aussi mis l'accent sur le temps de calcul nécessaire ainsi que la consommation d'énergie dans le cas de l'utilisation des algorithmes cryptographiques qui sont très gourmands.

Ils ont proposé une architecture (voir Figure 4-2) pour un meilleur traitement de la sécurité.

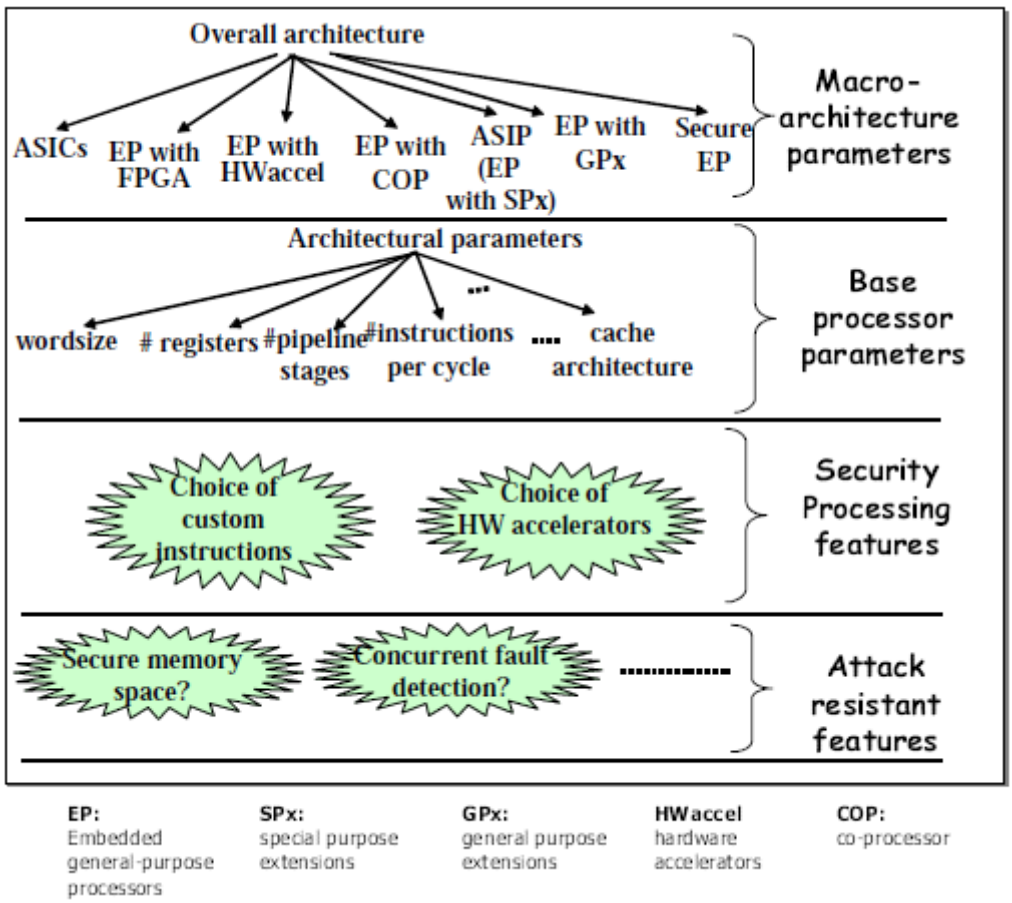


Figure 4-2: Proposition d'architecture pour la sécurisation des systèmes embarqués

4.4 A. Raghunathan & al (44)

Dans leur article, les auteurs ont présenté la nécessité de sécuriser les systèmes embarqués (plus précisément, les PDA et les téléphones cellulaires) et ont listé les exigences pour sécuriser la plus part de ces systèmes comme suit :

- L'identification de l'utilisateur se réfère au processus de validation des utilisateurs avant de les autoriser à utiliser le système.
- L'accès au réseau sécurisé fournit une connexion au réseau ou un accès au service uniquement si le dispositif est autorisé.
- Sécuriser les fonctions de communication par l'authentification des pairs communicants et assurer la confidentialité et l'intégrité des données transmises, ce qui empêche la répudiation d'une transaction de communication et protège l'identité des entités communicantes.
- Sécuriser le stockage permet de garantir la confidentialité et l'intégrité des informations sensibles stockées dans le système.

- La disponibilité veille à ce que le système peut remplir sa fonction et servir les utilisateurs légitimes à tout moment, sans être perturbé par des attaques de déni de service.

Les auteurs ont aussi présenté les défis pour sécuriser les systèmes embarqués et ont présenté l'insuffisance des processeurs, des batteries, etc.

Les auteurs ont présenté quelques solutions pour sécuriser ces systèmes ; telles que : l'utilisation des accélérateurs hardwares, l'utilisation des coprocesseurs destinés au traitement cryptographique, modifications dans l'utilisation des protocoles de sécurité, etc.

Enfin les auteurs ont mis l'accent sur la nécessité de faire un compromis entre le coût de l'implémentation de la sécurité et les attaques potentielles sur ce type de système et déterminer le niveau des attaques auxquelles ces systèmes peuvent résister et que les contremesures se diffèrent d'un système embarqué à l'autre, et ne peuvent en aucun cas être les mêmes pour tous les types de ces systèmes.

4.5 H. Luo & al (45)

En se basant sur la structure en couches de Smartphones, les auteurs ont proposé un cadre de sécurité hiérarchique pour les Smartphones (la sécurité physique, la sécurité du système d'exploitation, la sécurité des applications, la sécurité des données des utilisateurs et la sécurité des communications).ils présentent aussi des solutions de sécurité préliminaires, et donnent une direction de recherche pour le future.

4.5.1 Architecture des Smartphones

Les auteurs proposent l'architecture dans la Figure 4-3, comme l'architecture commune pour tous les Smartphones.

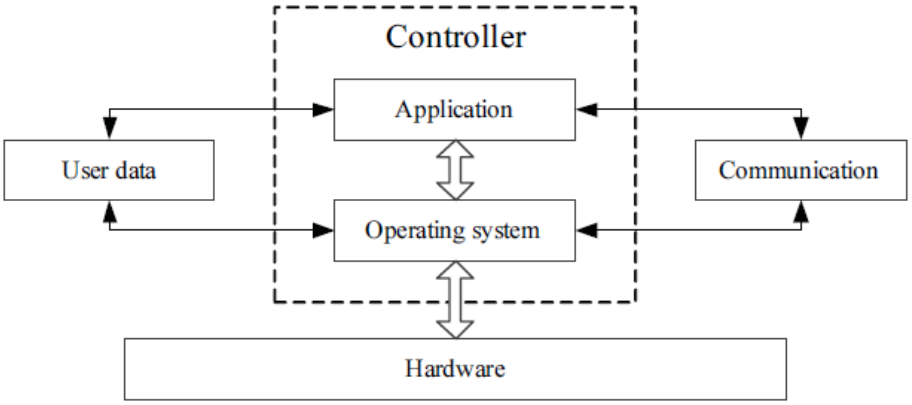


Figure 4-3: Architecture Commune entre les Smartphones

4.5.2 Structure hiérarchique de sécurité proposée

Pour une sécurité accrue des Smartphones, les auteurs ont proposé la structure hiérarchique suivante (Figure 4-4).

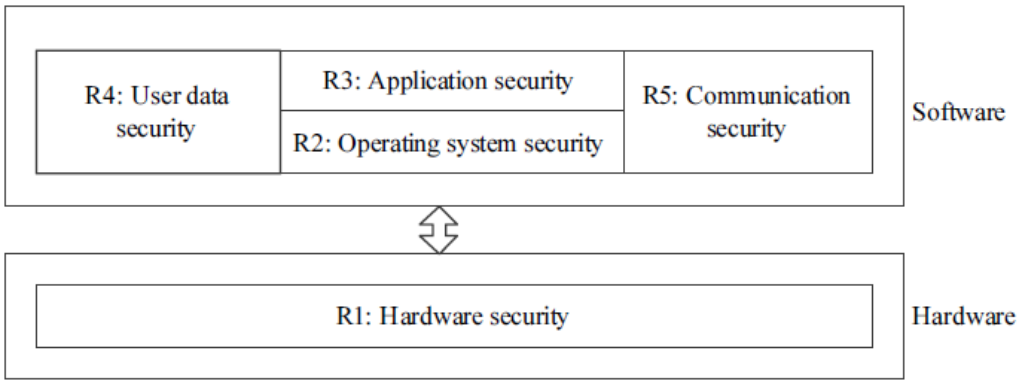


Figure 4-4: Structure hiérarchique de sécurité des Smartphones

4.5.3 Solutions de sécurité

Les auteurs ont mentionné deux approches pour améliorer la sécurité des Smartphones.

4.5.3.1 Améliorations au niveau du système lui-même

Ils sont difficiles à appliquer vue la nécessité de mettre à jour le système et nécessitent souvent plus d'études pour éviter l'incompatibilité avec le système qu'on veuille sécuriser. Voici les points les plus importants :

- Sécuriser le Boot
- Système de confirmation (ex. SMS, MMS, Etc.)
- APIs sécurisées (ex. HTTPS)
- Chiffrement des données

- Contrôle d'accès
- Etc.

4.5.3.2 Améliorations des outils de sécurité

Ça concerne les outils logiciels qu'on puisse utiliser pour avoir plus de sécurité :

- Anti-Spam
- Anti-virus
- Systèmes de détection d'intrusions (IDS)

4.6 Conclusion

Dans Cette partie, nous avons présenté quelques études concernant les approches de sécurisation des systèmes embarqués.

Dans (42) , les auteurs ont présenté la limitation des performances de calcule pour les systèmes embarqués en implémentant les différents algorithmes cryptographiques ; dans (43), les auteurs ont présenté quelques mécanismes pour assurer le niveau de sécurité souhaité. Ils ont aussi mis l'accent sur le temps de calcul nécessaire ainsi que la consommation d'énergie dans le cas de l'utilisation des algorithmes cryptographiques qui sont très gourmands.

Dans (44), les auteurs ont listé les exigences pour sécuriser la plus part des systèmes embarqués, ils ont aussi présenté les défis pour sécuriser les systèmes embarqués et ont présenté l'insuffisance des processeurs, des batteries, etc.

Dans (45), les auteurs ont proposé un cadre de sécurité hiérarchique pour les Smartphones, et ont présenté quelques approches pour les sécuriser.

Chaque type de système embarqué contient des spécifications à part qui le différencie des autres, et par conséquent, chaque système doit être étudié séparément des autres afin d'établir une solution de sécurité sur mesure.

Partie 2 :

Contribution

5 Architecture ARM

5.1 Introduction

ARM est une architecture RISC 32-bit, développée par la société ARM Ltd. Les processeurs ARM possèdent des caractéristiques uniques qui rendent l'architecture ARM la plus populaire dans les systèmes embarqués en particulier les Smartphones et les tablettes. Tout d'abord, les cœurs ARM sont très simples par rapport à la plupart des autres processeurs généralistes, ce qui signifie qu'ils peuvent être fabriqués en utilisant un nombre relativement restreint de transistors, ce qui laisse beaucoup d'espace sur la puce pour des applications spécifiques. Deuxièmement, l'ISA des processeurs ARM et la conception du pipeline visent à minimiser la consommation d'énergie, une exigence essentielle dans les systèmes embarqués mobiles. Troisièmement, l'architecture ARM est très modulaire: [la seule composante obligatoire d'un processeur ARM est le pipeline entier; tous les autres composants, y compris les caches, FPU (virgule flottante) et d'autres coprocesseurs sont facultatifs, ce qui donne une grande souplesse dans la conception des processeurs pour les applications spécifiques]. (46)[xiv]

Une étude (voir Figure 5-1) faite par (12) a montré que les systèmes embarqués utilisent de plus en plus une architecture ARM au lieux d'une architecture x86, avec 31 pour cent (en augmentation) des systèmes utilisant l'architecture ARM avec 24 pour cent (en diminution) des systèmes utilisant l'x86.

[xiv] La plus part de ce chapitre est résumé du Site officiel ARM sauf les sections mentionnées.

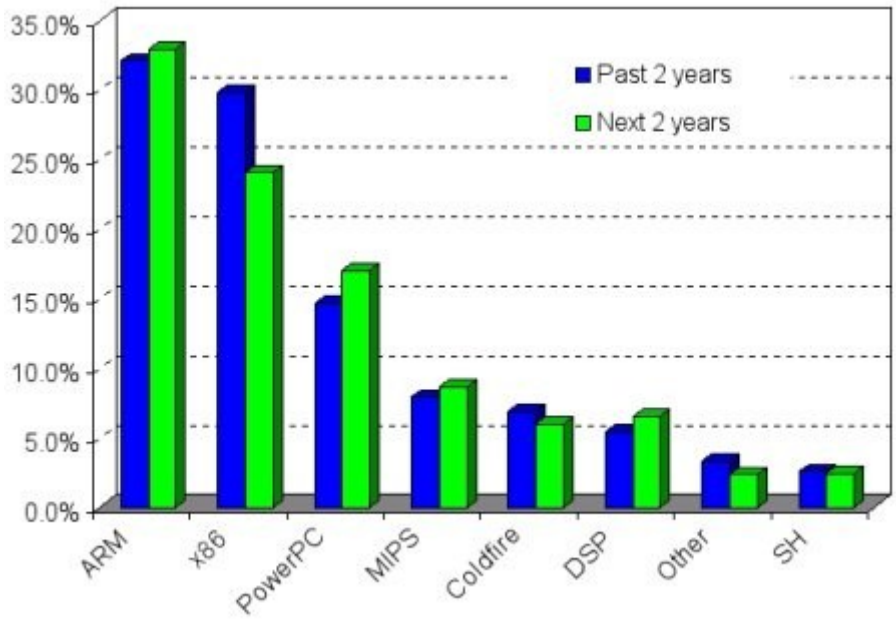


Figure 5-1: Architectures pour systèmes embarqués (12)

5.2 Améliorations dans l'ARM

Le cœur le plus célèbre est l'ARM7TDMI (47) qui comporte 3 niveaux de pipeline.

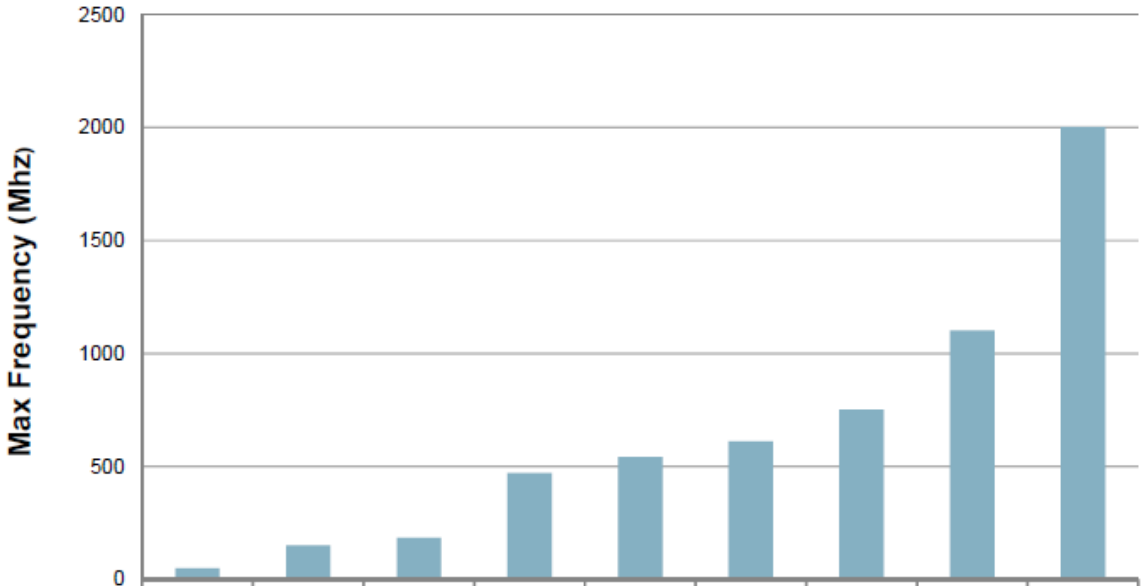
De plus, le ARM7TDMI dispose d'un second jeu d'instructions appelé THUMB permettant le codage d'instructions sur 16 bits et, ainsi, de réaliser un gain de mémoire important, notamment pour les applications embarquées (ex. : pour les Smartphones).

ARM Ltd. a ensuite développé le cœur ARM9 qui comporte 5 niveaux de pipeline. Cela permet ainsi l'augmentation du nombre d'opérations logiques sur chaque cycle d'horloge et donc une amélioration des performances en vitesse. (48)

De nombreux systèmes d'exploitation sont compatibles avec cette architecture :

- Symbian S60 avec les Nokia N97 ou Samsung Player HD ;
- iOS avec l'iPhone et l'iPad ;
- Linux, avec la plupart des distributions ou avec Android ;
- BlackBerry OS avec les BlackBerry
- Windows CE, Windows Phone 7 et Windows RT2, une version de windows 8.
- le système Playstation Vita ;
- ...]

La Figure 5-2 montre quelques processeurs ARM les plus utilisés.



	Cortex-M0	Cortex-M3	ARM7	ARM926	ARM1026	ARM1136	ARM1176	Cortex-A8	Cortex-A9 Dual-core
Max Freq (MHz)	50	150	184	470	540	610	750	1100	2000
Min Power (mW/MHz)	0.012	0.06	0.35	0.235	0.36	0.335	0.568	0.43	0.5

*Represents attainable speeds in 130, 90, 65, or 45nm processes

Figure 5-2: Puissances des processeurs ARM (49)

5.3 Des processeurs ARM en 64-bit pour 2014

L'ARM a annoncé que sa prochaine génération de processeurs, constituerait des puces 64-bit qui vont être les plus efficientes du marché. (49) (50),

Deux modèles 64 bits, l'ARM Cortex-A57 et A53. Tous deux seront basés sur le jeu d'instructions ARMv8 et seront gravés sur des processeurs à 20 nm.

Ces architectures sont considérées comme étant plus performantes et plus économiques que leurs précédentes.

L'ARM Cortex-A57 est conçu pour offrir une performance beaucoup plus grande que les processeurs actuels, tout en conservant la même consommation. Il sera décliné en plusieurs versions disposant de 1 à 4 cœurs.

Le Cortex-A57 contient un accélérateur cryptographique qui permet un gain de 3 fois jusqu'aux 10 fois de performance de chiffrement et de déchiffrement au niveau applicatif.

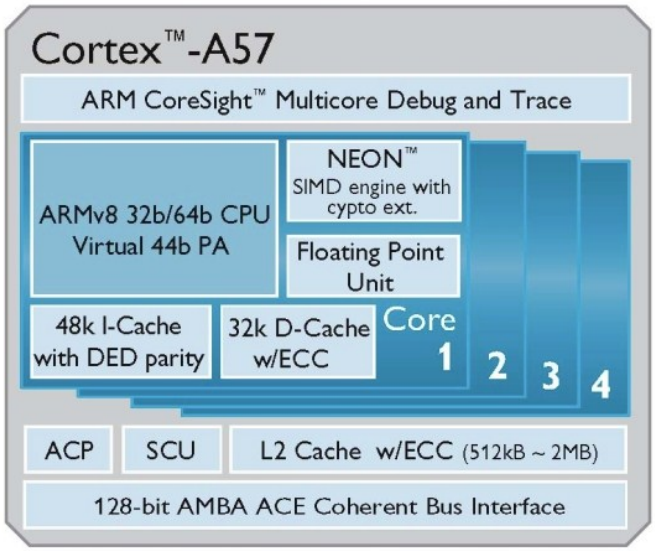


Figure 5-3: ARM Cortex A-57

Le Cortex-A57 contient un accélérateur cryptographique qui permet un gain de 3 fois jusqu'aux 10 fois de performance de chiffrement et de déchiffrement au niveau applicatif.

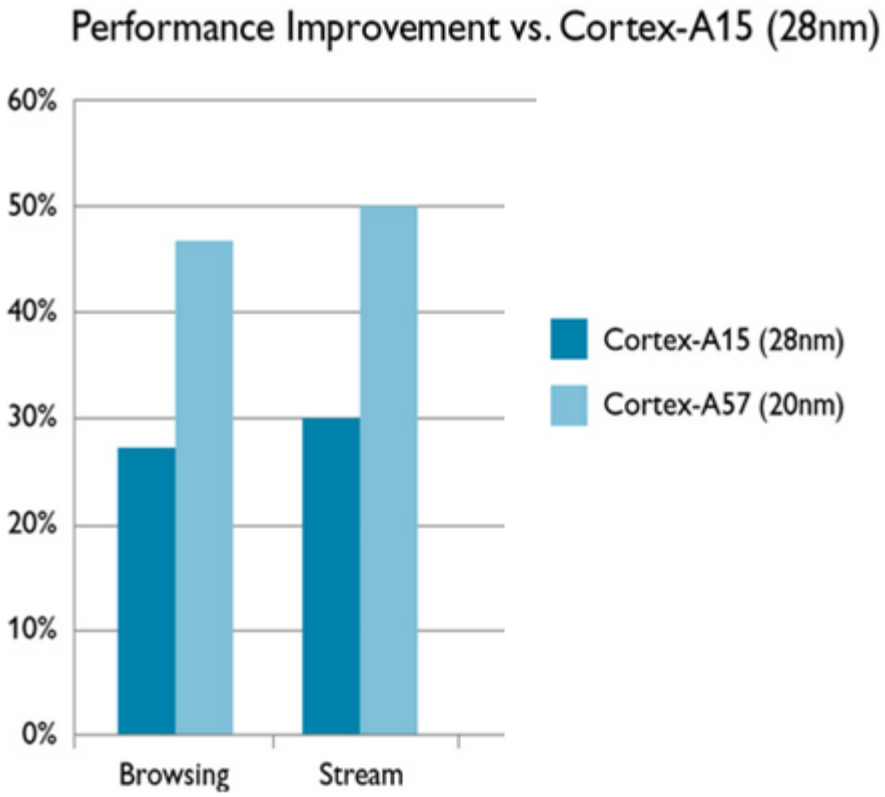


Figure 5-4: Evolution - du Cortex-A15 au Cortex-A57) (51)

5.4 ARM et la Sécurité

L'implémentation de la cryptographie au niveau application peut mener à l'augmentation des temps d'exécution des applications et ainsi elle peut causer des échéances pour les applications temps réel, et aussi, augmente la consommation d'énergie.

5.4.1 Le processeur ARM SC300

L'ARM SC300 est basé sur le processeur ARM Cortex-M3 (48), héritant ainsi des caractéristiques architecturales, une performance élevée, et un cout bas. Il inclut une configuration NVIC[xv] pour un grand déterminisme, réduisant ainsi le nombre d'interruptions, ce qui donne une très grande capacité d'exécuter de multiples algorithmes cryptographiques.

5.4.2 ARM SecureCore

Cette technologie regroupe un ensemble de dispositifs pour aider les fabricants à la création des dispositifs de sécurité capables de faire face à des attaques avancées qui sont gérés par les organisations et les laboratoires.

Les processeurs SecureCore s'intègrent avec les cartes SIM et les autres applications à carte à puce. (48)

5.4.3 Le Coprocesseur SafeXcel

SafeXcel est un module cryptographique qui fournit des performances maximales pour les applications qui nécessitent une sécurité élevée avec des contraintes temporelles (49). À l'aide des accélérateurs matériels dédiés, le module cryptographique fournit un premier gain de performance par rapport à l'implémentation de la sécurité au niveau applicatif. Le deuxième gain vient de l'utilisation du stockage local accessible rapidement à l'intérieur du module, ce qui élimine le temps pour la gestion de la mémoire.

[xv] Nested Vector Interrupt Controller

Le module cryptographique SafeXcel (Figure 5-5) fournit des implémentations matérielles des algorithmes cryptographiques suivants pour des performances optimales :

- AES, SHA-1, SHA-256, ARC4 (Pour les algorithmes cryptographiques symétriques)
- PKA (Public Key Acceleration)
- TRNG (True Random Number Generator) un module qui fournit un générateur de nombre aléatoire

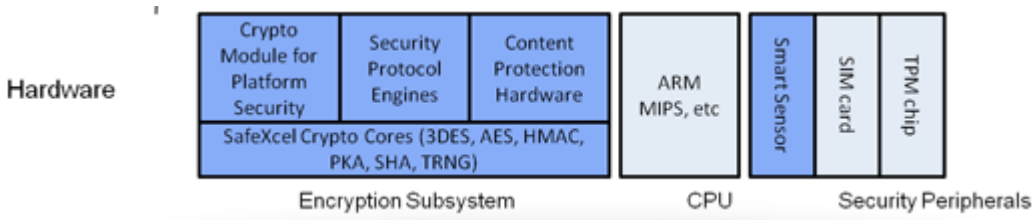


Figure 5-5: SafeXcel Crypto (52)

5.5 Conclusion

Le monde de l'embarqué est en grande évolution, et l'architecture ARM l'envahisse de plus en plus, surtout pour le cas des Smartphones qui nécessitent l'implémentation de la sécurité et la haute performance.

Aujourd'hui Intel se trouve en face d'un grand concurrent dans la fabrication de microprocesseurs. Cette concurrence va sûrement générer des processeurs de très haute performance avec des tailles et des coûts très réduits, ce qui engendre un grand bénéfique que ce soit au niveau de la rapidité d'exécution qu'au niveau des coûts conduisant ainsi à une sécurité plus élevée des systèmes embarqués.

6 Système d'exploitation Android

6.1 Introduction

Android est un système d'exploitation open source utilisant le noyau Linux, conçu spécifiquement pour les Smartphones, tablettes tactiles, PDA et terminaux mobiles. Il est proposé de façon gratuite et librement modifiable aux fabricants de téléphones mobiles, ce qui facilite son adoption.

Il comporte une interface spécifique, développée en Java, les programmes sont exécutés via un interpréteur JIT, toutefois il est possible de passer outre cette interface, en programmant ses applications en C, mais le travail de portabilité en sera plus important.

Si la majorité des périphériques Android sont basés sur l'architecture ARM, l'hétérogénéité des versions et des coprocesseurs peut varier grandement d'un constructeur à l'autre. Pour communiquer avec les périphériques, Android contient une couche d'abstraction matérielle qui est une spécification et un utilitaire logiciel qui traque les périphériques du système informatique, le but de cette couche est d'éviter aux développeurs d'implémenter manuellement le code spécifique à un périphérique. (53) À la place, ils peuvent utiliser une couche connectable qui fournit des informations à propos du périphérique, tel que cela se passe par exemple lorsqu'un utilisateur branche ou débranche un périphérique USB.

Cette couche implémente un certain nombre de fonctions spécifiques au matériel : interfaces d'entrées-sorties, contrôleur d'interruptions, caches matériels, mécanismes de communication multiprocesseur, etc., elle isole ainsi le noyau du système des spécificités des plates-formes matérielles. (53)

6.2 L'architecture d'Android

Connaitre l'architecture d'un système d'exploitation est primordiale afin de faire des choix pour l'implémentation de la sécurité ainsi que le temps réel.

Android est conçue pour des appareils mobiles au sens large. Ce système est utilisable aussi pour des tablettes, des ordinateurs portables, des bornes interactives, etc.

La plate-forme Android est composée de différentes couches : (54)

- un noyau Linux qui lui confère notamment des caractéristiques multitâches ;
- des bibliothèques graphiques, multimédias ;
- une machine virtuelle Java adaptée : la Dalvik Virtual Machine ;
- un Framework applicatif proposant des fonctionnalités de gestion de fenêtres, de téléphonie, de gestion de contenu, etc. ;
- des applications dont un navigateur web, une gestion des contacts, un calendrier, etc.
- Les composants majeurs de la plate-forme Android sont résumés sur le schéma de la Figure 6-1.

6.2.1 DES BIBLIOTHÈQUES C/C++

Au-dessus du noyau proprement dit se loge un ensemble de librairies C/C++ constituant les couches bases du système. Parmi celles-ci on peut noter la libc (librairie système C standard) dont l'implémentation a été adaptée pour l'occasion à un mode embarqué

Le système Android est donc constitué d'un noyau Linux et d'une suite de bibliothèques C/C++ fort utile. L'accès à ces librairies se fait exclusivement par l'API Java d'Android. Il n'est donc pas possible de faire des appels directs aux couches basses et il faudra se contenter des fonctionnalités exposées. (55)

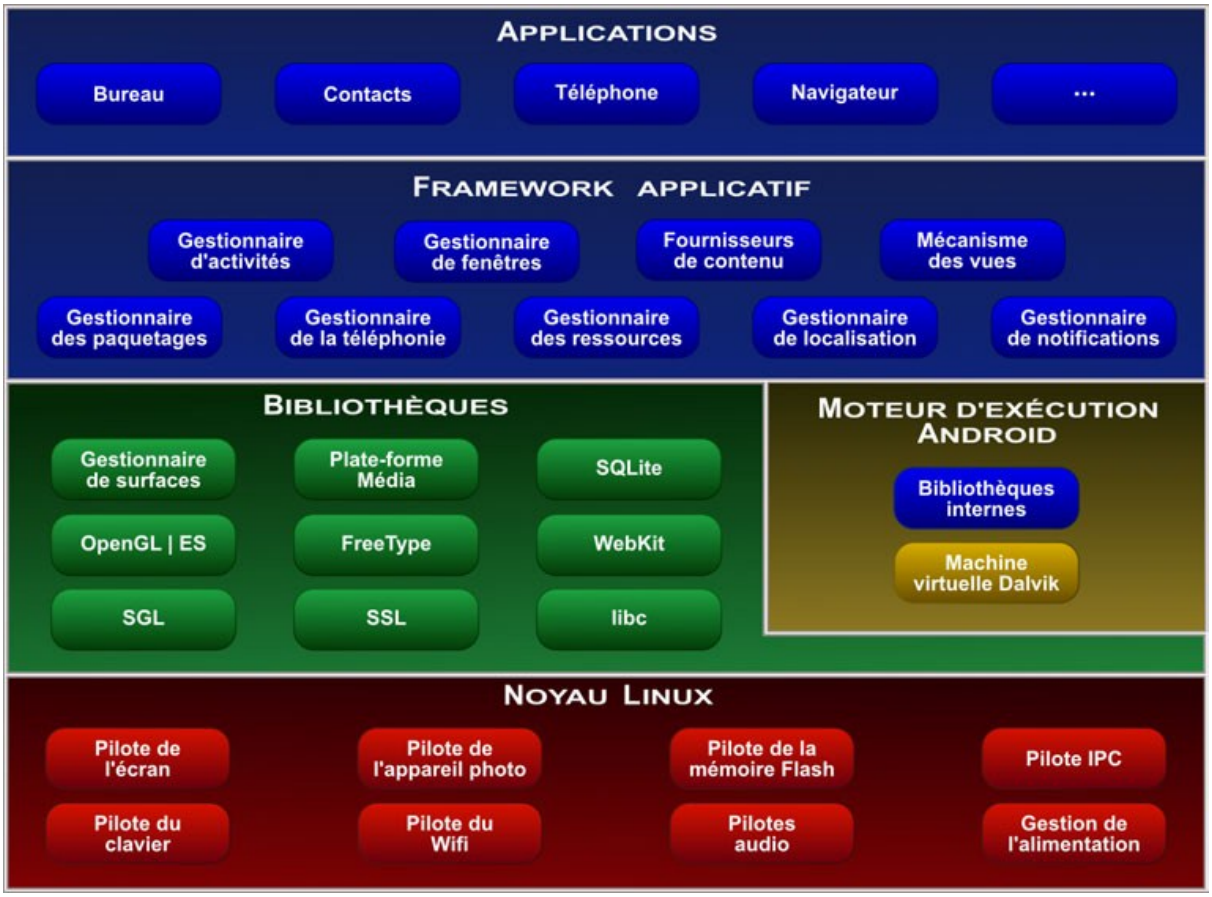


Figure 6-1: Les composants d'Android (54)

6.2.2 UN MIDDLEWARE JAVA

Si nous remontons les couches de la plateforme, après le noyau Linux et les librairies C/C++, nous arrivons au middleware, l'environnement d'exécution Android. Cet environnement est constitué en premier lieu d'une machine virtuelle.

Cette machine virtuelle a été dénommée Dalvik. Dalvik interprète donc du code, dont les sources sont écrites en Java. Voir Figure 6-2

6.2.2.1 DALVIK[xvi]

Google a fait le choix de s'écarter des standards Java et de proposer sa propre machine virtuelle. Google ne garde de Java que le langage et une partie des API standard et met de côté la plateforme d'exécution (JVM).

La raison d'être de Dalvik est de fournir un environnement optimisé pour le mobile où les ressources CPU et mémoires sont précieuses. Par exemple, pour de meilleures

[xvi] Cette partie est extraite du (55) avec quelques modifications

performances sur un système embarqué, Dalvik est une machine virtuelle dite «registered-based», basée sur les registres, et non pas «stack-based» comme la JVM.

Par ailleurs, sur Android, par défaut, chaque application tourne dans son propre processus Linux, c'est-à-dire sa propre VM. L'un des points forts d'Android étant ses capacités multitâches, il doit être possible de lancer efficacement plusieurs VM Dalvik sur le terminal.

À chaque fois qu'une application est démarrée, une VM se crée, puis le processus meurt à la fermeture du programme. On est bien loin du modèle de la JVM pensée pour être rarement arrêtée ou relancée et censée exécuter simultanément plusieurs applications distinctes isolées entre elles par une hiérarchie de ClassLoader.

Les besoins d'Android imposaient vraiment de concevoir une machine virtuelle spécifique.

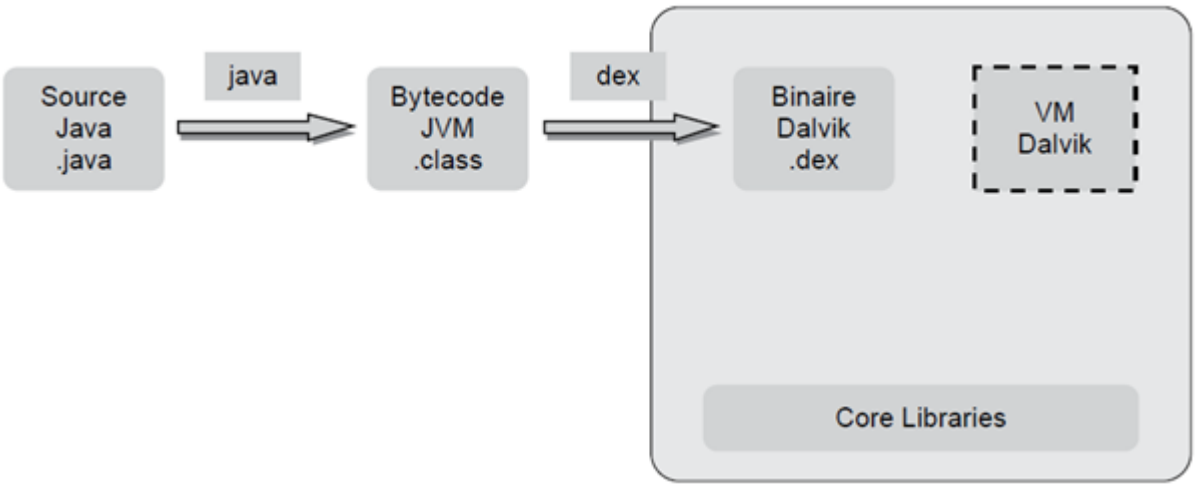


Figure 6-2: La chaine de compilation Android (55)

Le langage Java et sa syntaxe sont conservés tels quels, son API standard n'est pas modifié non plus, du moins le sous-ensemble qui a été retenu, car une partie de cette API n'a pas été portée vers Android.

Enfin, le format binaire des classes est carrément incompatible, la machine virtuelle Android ne respecte pas la spécification de Sun de la JVM.

Le « Write once, run anywhere » n'est plus de mise, c'est pour cela que Dalvik ne peut se prévaloir du qualificatif de JVM.

6.3 Android et les applications temps réel

Le système d'exploitation Android est largement utilisé dans plusieurs types de plates-formes mobiles et embarqués, y compris les téléphones portables et les tablettes. Comme il est librement disponible et hautement configurable, Android peut être vu comme une plate-forme logicielle pour la construction des systèmes embarqués répartis qui peuvent nécessiter en plus de la sécurité, des contraintes de temps réel.

L'inclusion de la sécurité dans les systèmes embarqués peut causer des délais d'exécution plus importants. Les systèmes d'exploitation à usage universel ne prennent pas des contraintes externes pour la gestion du temps, et ainsi, ils ne peuvent pas répondre à des contraintes temporelles alors qu'il faut respecter des échéances pour les tâches critiques dans le cas des systèmes embarqués d'où la nécessité du temps réel et des schéduleurs déterministes. Par conséquent, l'implémentation de la sécurité dans l'embarqué, nécessite des considérations de développement au niveau application ainsi que des modifications dans le système.

Beaucoup d'études s'intéressent à la possibilité de l'utilisation d'Android dans des applications critiques comme l'e-banking, ou dans d'autres plates-formes embarquées tel que le domaine de l'automobile ou le militaire, exigeant ainsi des garanties en temps réel.

Avec toutes les caractéristiques regroupées sur une seule plate-forme, l'imprévisibilité du temps de réponse des applications en interaction et partageants le temps d'exécution, le système risque d'échouer, entraînant ainsi un mauvais fonctionnement. Par exemple, si quelqu'un utilise son système e-banking, et un appel téléphonique à haute priorité est reçu, le scheduleur peut écarter l'application e-banking pendant une longue période, ou si le temps de réponse à un appel téléphonique était trop long, l'appel serait manqué.

Dans (56), les auteurs ont montré que, S'il y a plus d'une douzaine d'interruptions par seconde, Android ne peut pas garantir un comportement fiable par rapport à des contraintes temps réel (par exemple, les temps de réponse augmentent de manière significative).

Puisque Android prend en charge la préemption et le multitâches, les concepteurs des applications sécurisés, qui nécessitent du temps réel qui veulent utiliser Android comme

système d'exploitation doivent effectuer des mesures de son comportement avec soin, sur la fiabilité de leur système, en mettant l'accent principalement sur la machine.

6.3.1 L'inclusion du temps réel dans Android

Android est basé sur le noyau Linux, qui n'est pas temps réel ; quatre directions possibles pour intégrer le temps réel dans l'architecture d'Android (57) La première approche (Figure 6-3) consiste à remplacer le noyau Linux par un autre qui garantit les caractéristiques temps réel, et au même temps, elle propose l'inclusion d'une machine virtuelle temps réel. La seconde approche (Figure 6-4) respecte l'architecture standard d'Android et propose une extension de la machine virtuelle Dalvik, ainsi que le remplacement du noyau standard (Linux Kernel) par un système d'exploitation temps réel basé sur Linux (RTLinux). La troisième approche (Figure 6-5) remplace le noyau Linux par une version Linux temps réel et les applications temps réel utilisent directement le noyau. Enfin, la quatrième approche (Figure 6-6) propose l'ajout d'un superviseur en temps réel qui prend en charge l'exécution parallèle de la plate-forme Android dans une partie alors que l'autre partie sera dédiée aux applications temps-réel.

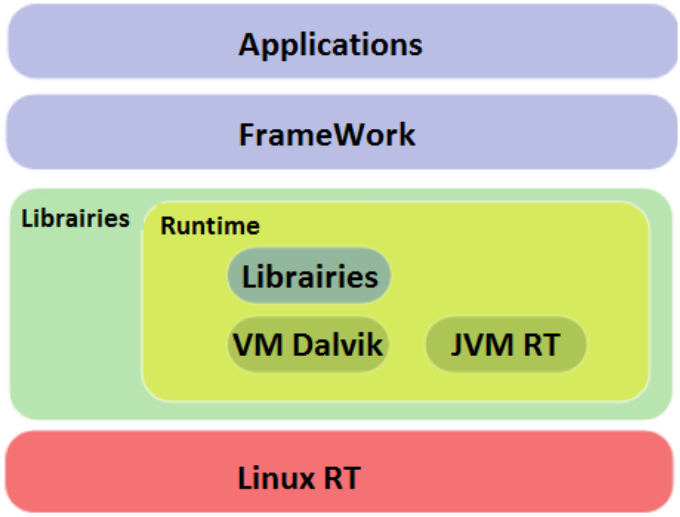


Figure 6-3: Android-Entièrement temps réel (57)

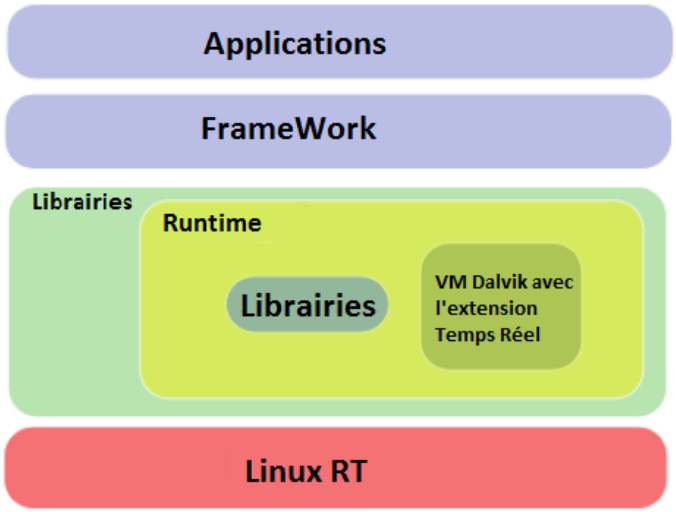


Figure 6-4: Android- Extension temps réel (57)

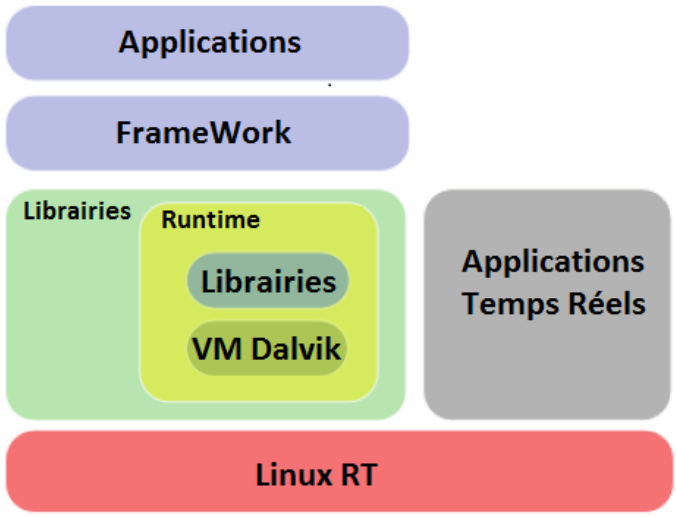


Figure 6-5: Android- Partie en temps réel (57)

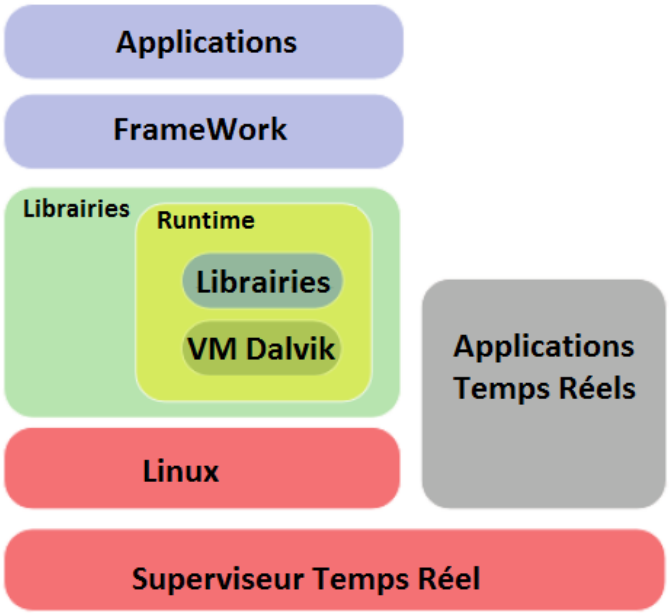


Figure 6-6: Android - Superviseur temps réel (57)

L'inconvénient majeur de la première approche est que tous les pilotes supportés nativement doivent être implémenté dans le nouveau système d'exploitation avec la prise en charge du déterminisme (un schéduleur déterministe). Cette tâche peut être très couteuse surtout dans la phase de création, mais elle peut être aussi bénéfique en ce qui concerne les autres couches qui n'auront aucune modification, et ainsi chaque partie du système (VM, Noyau, etc.) peut évoluer séparément pour garantir le temps réel.

La différence majeure entre la première approche est la deuxième c'est que, cette dernière propose une modification de l'extension Dalvic avec des capacités Temps réel, et ainsi il y'aura des modifications de la ramasse miette et d'autres algorithmes.

L'inconvénient majeur de la deuxième approche c'est qu'à chaque version d'Android, on doit refaire tout le travail.

La troisième approche est aussi basée sur le noyau LinuxRT, l'avantage de cette approche est la possibilité d'exécuter nativement des applications temps réel sur le système.

L'inconvénient majeur, c'est que les applications temps réel nécessitant une machine virtuelle ne peuvent pas s'exécuter.

Finalement, la quatrième approche nécessite le déploiement d'un superviseur, qui peut faire fonctionner l'Android sur une partie et les applications temps réel sur une autre partie, d'une façon parallèle. L'inconvénient majeur, c'est que cette solution nécessite des processeurs plus puissants que celles des téléphones mobiles.

6.4 Packages Cryptographiques dans Android

Il existe de nombreux exemples dans le monde de cryptage où des erreurs ont compromis les objectifs de sécurité.

Pour une bonne implémentation de la cryptographie, Android contient la Bibliothèque Javax.crypto.

6.4.1 javax.crypto

Ce paquage fournit les classes et les interfaces pour des applications cryptographiques mettant en œuvre des algorithmes cryptographiques symétriques et asymétriques, que ce soit par flux ou par bloc.

6.4.2 Interfaces et Classes

Javax.crypto.* : fournit les classes et les interfaces pour effectuer des opérations cryptographiques sur les algorithmes cryptographiques.

Java.security.* : fournit les classes et les interfaces pour le Framework de sécurité Java. Elles permettent entre autres de réaliser des opérations de génération de nombres aléatoire et de signatures à l'aide d'algorithmes asymétriques

6.5 Bonnes pratiques pour une bonne sécurité

La bonne utilisation de la cryptographie dans les systèmes embarqués spécialement et les systèmes informatique d'une façon générale est complexe, une petite erreur d'implémentation peut transformer un système sécurisé, en un système très vulnérable aux attaques. C'est pour cette raison, il faut suivre les pratiques suivantes :[xvii]

S'assurer qu'un fichier stocké sur l'appareil, ne peut en aucun cas, être lu par une personne qui ne possède pas l'appareil (physiquement)

[xvii] Voir aussi (62)

Aussi, on ne doit jamais stocker la clé secrète en clair sur le dispositif. Il veut mieux la régénérer à chaque fois qu'on aura besoin

Utiliser des algorithmes de sécurité standard pour résoudre les problèmes de sécurité standard

Si on a besoin de créer une session cryptée entre un client et un serveur, la première pensée doit être de la faire comme un service Web à l'aide d'une session HTTPS, plutôt que de concevoir un nouveau protocole. HTTPS est un protocole bien testé et standard pour créer et maintenir une connexion sécurisée entre un navigateur et un serveur Web.

L'utilisation des algorithmes de sécurité standard s'étend à l'utilisation des implémentations standards de ces algorithmes.

Il faut toujours utiliser les bibliothèques fournies avec le système. Si on a besoin d'un algorithme qui n'est pas fourni avec le système, on doit toujours essayer de l'avoir d'une organisation connue dans le domaine de la sécurité.

Intégrer un spécialiste de sécurité lors de la phase de conception

Il faut toujours être très prudent lors de la conception de systèmes qui utilisent le chiffrement. Il se peut qu'on fasse des petites erreurs qui dégradent les capacités de sécurité du système.

Etre sûr que l'implémentation de la sécurité ne conduit pas en un blocage de l'appareil (DoS).

Beaucoup de systèmes embarqués ne sont pas capables d'implémenter des schémas cryptographiques complexes, ainsi, il faut faire une analyse soigneuse sur les capacités du système.

6.6 Conclusion

L'Android peut être vu comme une cible pour beaucoup de systèmes embarqués, dont beaucoup nécessitent d'être sécurisés et temps réel. Ce système d'exploitation pour l'embarqué, est principalement conçu pour l'industrie de la téléphonie mobile et cela a considérablement influencé son architecture.

Cependant, avec un petit effort, il est possible d'avoir un comportement temps réel, en étudiant l'environnement dans lequel le système va fonctionner.

Aussi, pour une bonne sécurité du système, La meilleure pratique de sécurité des données est de conserver toutes les données d'entreprise sur des serveurs derrière le pare-feu et d'utiliser l'appareil mobile pour y accéder. Mais parfois, le stockage des données d'entreprise localement sur un périphérique mobile ne peut pas être évité.

Les bibliothèques de chiffrement Android permettent de protéger les données sur un périphérique mobile avec le plus haut niveau de sécurité ainsi que l'utilisation des protocoles sécurisés (tel que HTTPS).

7 Etude Expérimentale

7.1 Introduction

Les ressources limitées des systèmes embarqués causent un défi important pour l'implémentation des algorithmes cryptographiques puissants, qui sont généralement exigeants en ressources. L'objectif de cette étude est de mesurer l'impact des architectures embarquées sur de différents algorithmes cryptographiques. Une étude statistique sur l'implémentation de ces algorithmes dans les Smartphone Android est présentée dans ce chapitre, cela permet de mettre l'accent sur les différences pouvant influencer les performances entre ces architectures. (58)[xviii]

7.2 Algorithme

Dans notre travail, on a choisi d'implémenter deux algorithmes cryptographiques[xix], l'AES sur 256 bits et le RC5 sur 128 bits. L'AES est le standard de chiffrement pour les organisations du gouvernement des États-Unis, Il est également approuvé par la NSA (National Security Agency) pour les informations top secrètes (59). Tandis que le RC5 est suggéré comme un bon algorithme pour les systèmes embarqués en particulier les réseaux de capteurs.

Tableau 7-1: Propriétés des algorithmes implémentés (58)

Algorithme	Type	Clé (bit)	Block (bit)
AES	Block	256	128
RC5	Block	128	64

7.3 Plates-formes Hardware

Nous avons évalué la performance des fonctions cryptographiques sur les différents processeurs embarqués suivants dédiés à des Smartphones :

[xviii] La source principale de cette partie est notre article (58)

[xix] L'implémentation de l'AES est développée avec l'AGL Eclipse, basée sur le code source de l'application AES Crypto (voir Google Play) avec des modifications du code (Renvoie du temps de calcul pour chaque opération)

L'implémentation du RC5 est développée avec l'AGL Windev Mobile.

- Cortex-A8 1Ghz/64KO de Cache (RISC) embarqué dans une tablette GalaxyTab
- Scorpion 1 GHz/256KO de Cache (RISC) embarqué dans un Smartphone Sony Ericson
- Cortex-A9 DualCore 1,2Ghz/64KO de Cache (RISC) embarqué dans un Smartphone Galaxy SII

Le point commun de ces Smartphones c'est qu'ils fonctionnent tous sous le système d'exploitation Android.

Pour des raisons de comparaison on a appliqué les mêmes tests sur un ordinateur portable HP avec un processeur AMD Athlon Dual Core 1,4 GHz d'une architecture Hybride (CISC/RISC).

7.4 Méthodes expérimentales

Nos expériences consistent à tester les algorithmes cryptographiques sur chaque architecture en augmentant la taille des données en entrée. Pour chaque lot de données, on répète plusieurs fois l'expérience et la valeur retenue est la minimale des valeurs mesurées.

Pourquoi prendre la valeur minimale ?

Dans d'autres études (60) (42), les auteurs ont pris la valeur moyenne des résultats de calcul, dans notre étude, on a plutôt choisi de prendre la valeur minimale pour deux raisons :

1. D'après les résultats des tests on a remarqué que les mémoires caches de chaque architecture influent de manière significative sur les résultats obtenus : en fixant la taille des données de test, le temps de calcul dans les dernières mesures est beaucoup plus rapide que dans les premières, et on a remarqué une stabilisation du temps de calcul après les dix premières mesures (c'est le temps réel de calcul pour chaque processeur) (voir Figure 7-1). Les auteurs dans (42) ont testé les algorithmes sur des architectures embarqués qui n'ont pas de mémoires caches (ex. : Atmega, M16C/10, etc.), et c'est pour cette raison, ils ont choisi la valeur moyenne.

Aussi, le caractère multitâches du système d'exploitation Android et le non déterminisme de son schéduleur (Android[xx] n'est pas temps réel)[xxi] causent de temps en temps des retardements dans les temps d'exécution des algorithmes cryptographiques (cela peut causer des échéances dans le cas des systèmes critiques).[xxii]

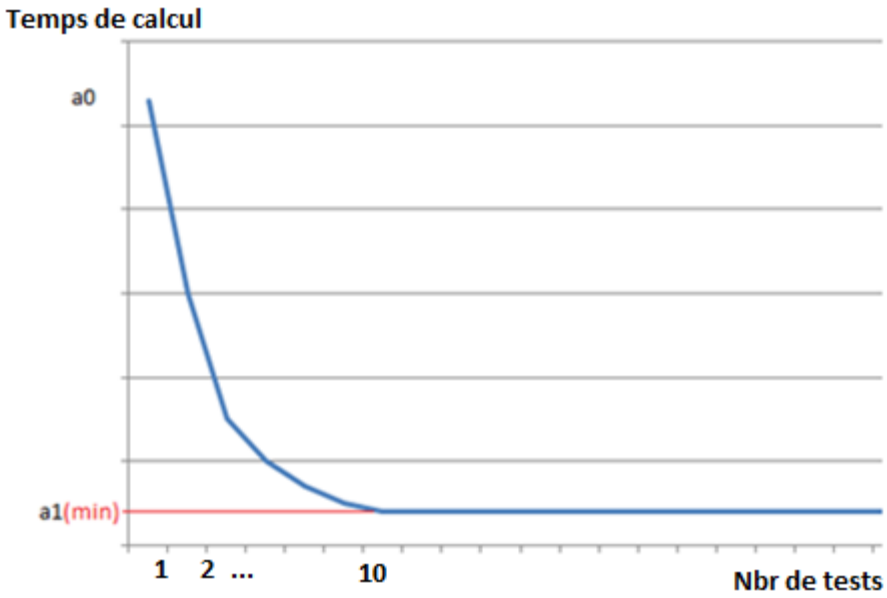


Figure 7-1: Effet des mémoires (58)

2. L'utilisation de la moyenne des temps de calcul comme référence cause des erreurs remarquables dans les résultats. Pour bien éclaircir cela, on a calculé la régression pour chacun des tests comme le montre les graphes suivants (Figure 7-2 jusqu'à Figure 7-5) en prenant dans les premiers tests le temps moyen comme référence et le temps Min dans les deuxièmes : On remarque que, dans le cas de l'utilisation de la valeur moyenne comme référence, on aura une erreur de 0.1% alors qu'en utilisant la valeur Min, l'erreur disparait. La différence de temps de calcule est due à plusieurs paramètres tels que l'utilisation des mémoires caches dans les processeurs embarqués modernes ainsi que le caractère multitâches des systèmes d'exploitation utilisés.

[xx] Android est le système d'exploitation des Smartphones étudiés.
[xxi] Voir chapitre 6.
[xxii] Le temps de calcul considéré = Temps de traitement CPU + Temps d'accès à la mémoire cache.

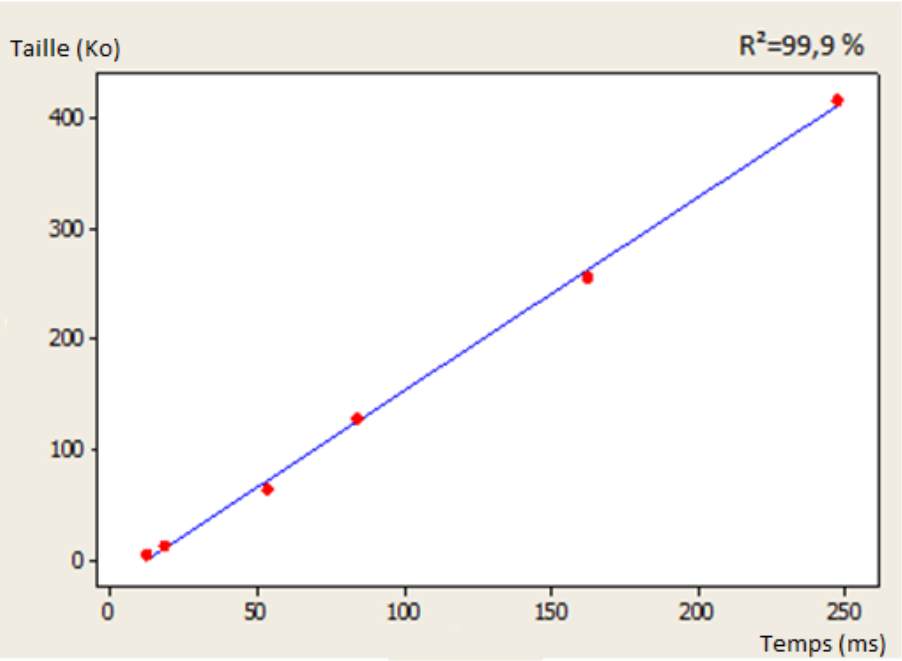


Figure 7-2: Régression en prenant le temps Moyen comme référence Pour le CPU Cortex A8 (Galaxy Tab) (58)

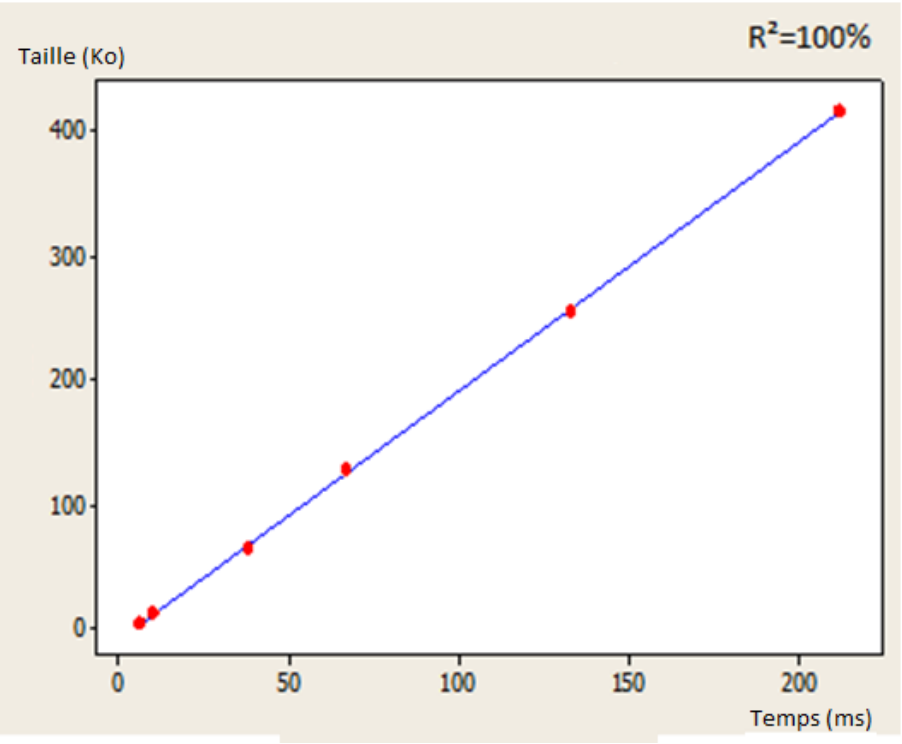


Figure 7-3: Régression en prenant le temps Min comme référence Pour le CPU Cortex A8 (Galaxy Tab) (58)

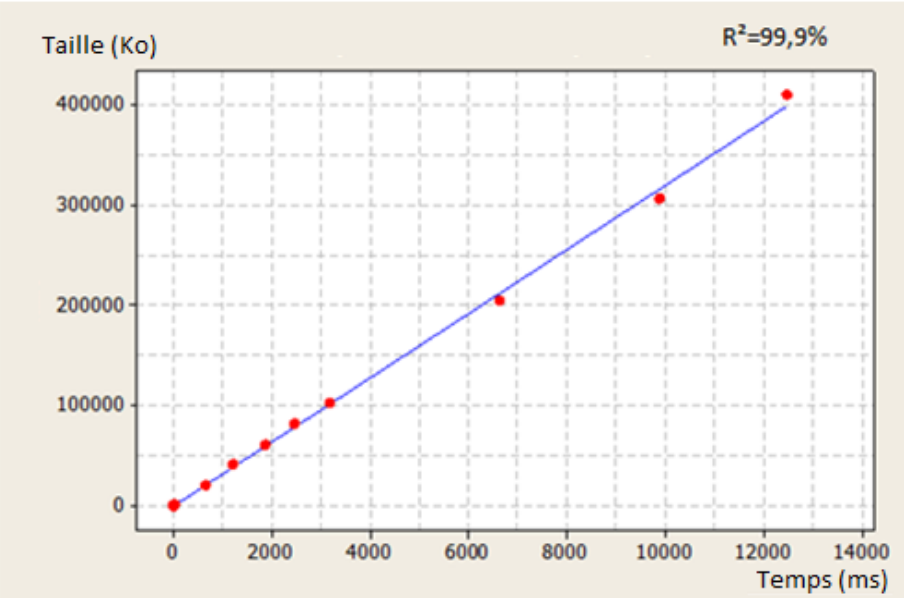


Figure 7-4: Régression en prenant le temps Moyen comme référence pour le CPU AMD Athlon (HpPavillion) (58)

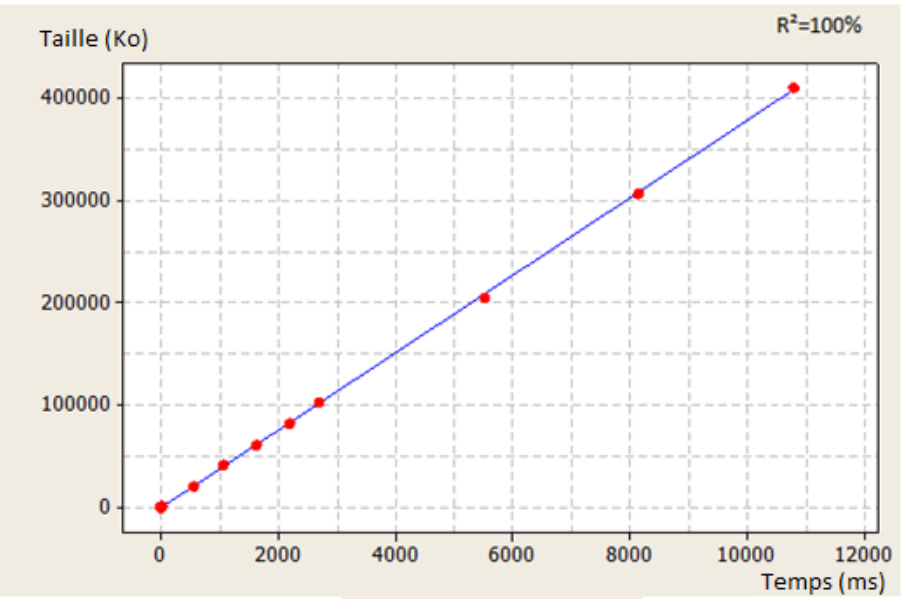


Figure 7-5: Régression en prenant le temps Min comme référence pour le CPU AMD Athlon (HpPavillion) (58)

7.5 Evaluation des performances

La performance de certains algorithmes cryptographiques nécessite une analyse particulière pour les utiliser dans les systèmes embarqués afin de déterminer l'impact des ressources souvent limitées dans ce type de systèmes.

7.5.1 Evaluation du Temps de calcul

Les résultats des tests des deux algorithmes l'AES (256) et le RC5 (128) sur les différentes architectures sont illustrés dans les Figure 7-6 et Figure 7-7.

La Figure 7-6 montre une différence considérable dans le temps de calcul entre l'AMD athlon et les autres architectures.

Le processeur AMD athlon est équipé d'une unité de calcul en virgule flottante FPU (*Floating Point Unit*) alors que les autres n'en ont pas, elle est émulée par le noyau linux de l'Android(ce qui est largement moins performant).

La FPU est fortement sollicitée par toutes les opérations mathématiques complexes (compression, manipulation d'images, cryptographie, cheksum, etc.) ce qui explique ces résultats

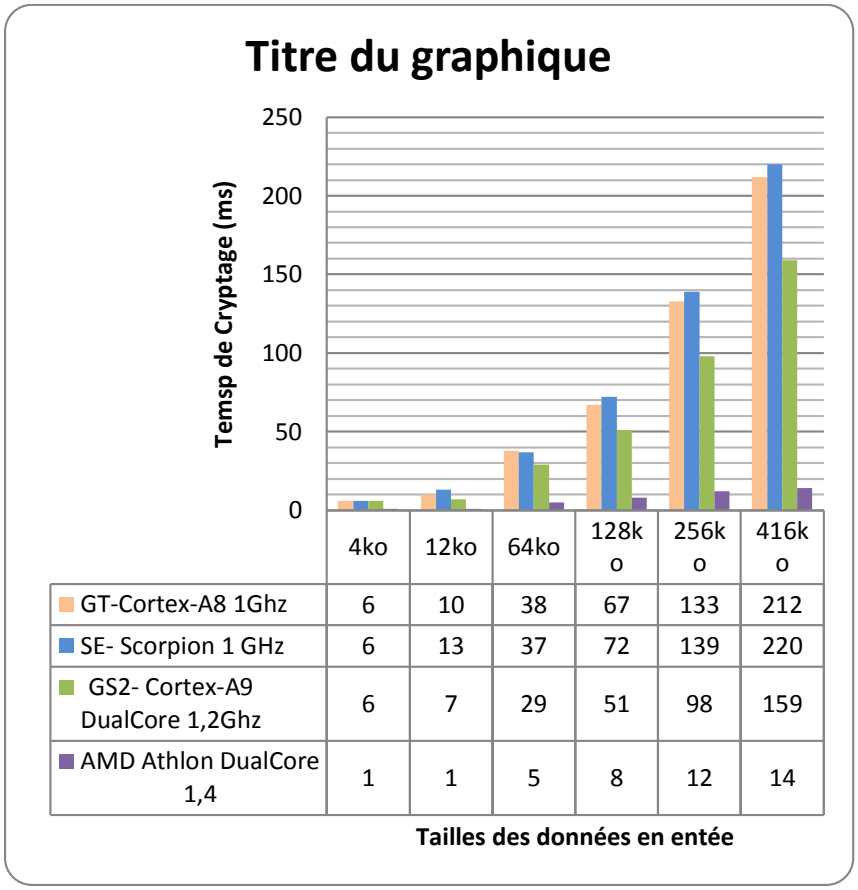
Malgré cette différence, le temps de calcul des différents processeurs embarqués reste raisonnable, sourtout pour les systèmes qui ne demandent pas de fortes contraintes temps réel.

Les Figure 7-6 et Figure 7-7 montrent aussi des petites différences de calcule les autres processeurs ARM. Cela est signifiable vue que le processeur Cortex A9 est plus rapide et plus performant que les deux autres.

En comparant nos résultats avec d'autres études[xxiii], on conclut que les processeurs embarqués dans les Smartphones modernes sont en grande évolution, et sont devenus capables d'implémenter des protocoles avec différents schémas cryptographiques, surtout avec l'apparition des processeurs ARM Cortex A57 qui peuvent accélérer le traitement cryptographique jusqu'à dix fois.[xxiv]

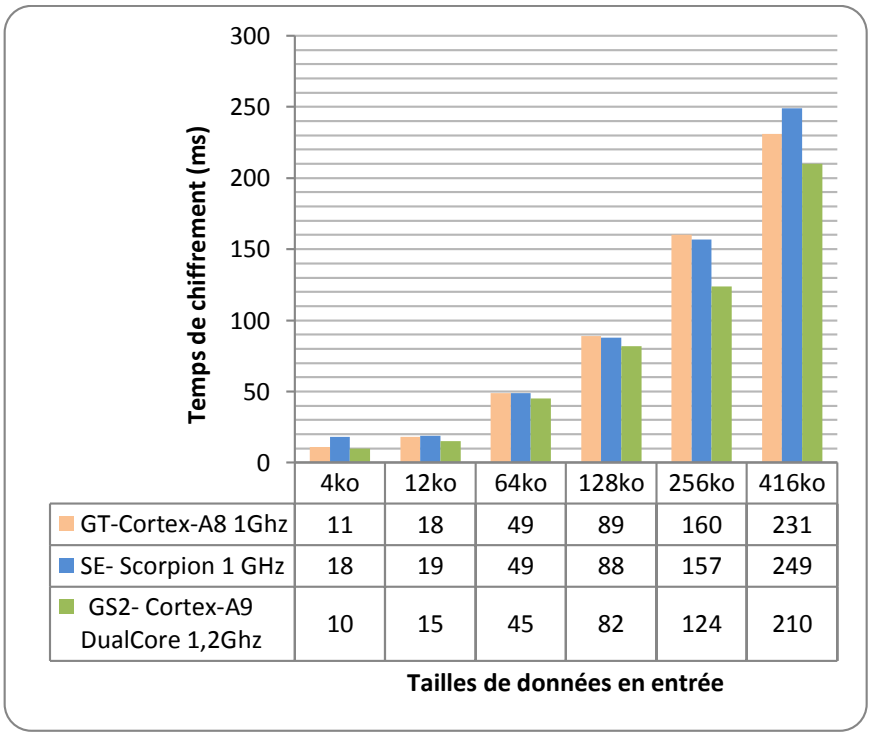
[xxiii] Comme celles de (60) et (42)

[xxiv] Voir Chapitre 5



	4ko	12ko	64ko	128ko	256ko	416ko
GT-Cortex-A8 1Ghz	6	10	38	67	133	212
SE- Scorpion 1 GHz	6	13	37	72	139	220
GS2- Cortex-A9 DualCore 1,2Ghz	6	7	29	51	98	159
AMD Athlon DualCore 1,4	1	1	5	8	12	14

Figure 7-6: Temps d'exécution de l'algorithme RC5 (128) pour les différentes architectures (58)



	4ko	12ko	64ko	128ko	256ko	416ko
GT-Cortex-A8 1Ghz	11	18	49	89	160	231
SE- Scorpion 1 GHz	18	19	49	88	157	249
GS2- Cortex-A9 DualCore 1,2Ghz	10	15	45	82	124	210

Figure 7-7: Temps d'exécution de l'algorithme AES (256) pour les différentes architectures (58)

7.5.2 Saturation de la RAM

On a aussi constaté une dégradation des performances dans le cas où il y'a une saturation de la RAM, pour bien éclaircir cela, on a fait des tests de l'algorithme RC5 (128) sur le HP Pavillion avec le processeur AMD Athlon qui a une RAM de 1GB en considérant des tailles de données suffisamment grandes pour mieux voir l'effet de saturation comme le montre les Figure 7-8 et Figure 7-9.

La Figure 7-9 représente la régression en testant l'algorithme RC5 sur des tailles de données assez grandes pour saturer la RAM (500 MB, 600 MB, et 700 MB). Elle Montre que l'utilisation de ces données donne une régression non linéaire (erreur égale 22% en la comparant à une régression linéaire) alors que la Figure 7-8 montre la linéarité de la régression en éliminant les tailles de données qui saturent la RAM.

Ainsi, l'utilisation des données qui peuvent saturer la RAM, ou l'utilisation d'une RAM de taille réduite peut ralentir le traitement d'une façon signifiable.

7.5.3 Evolution des Processeurs

Selon la Figure 7-10, la vitesse des processeurs s'améliore avec un rythme rapide, ainsi les processeurs embarqués sont de plus en plus aptes à gérer les tâches de chiffrement nécessaires pour protéger les données.

Les processeurs embarqués sur les Smartphones modernes sont des processeurs très puissants, on trouve par exemple un processeur Dual-core 1.2 GHz pour l'iPhone 5 ou même un processeur Quad-core 1.4 GHz Cortex-A9 pour le Samsung Galaxy SIII, et par conséquent, ces systèmes sont de plus en plus capables d'implémenter les schémas cryptographiques modernes.

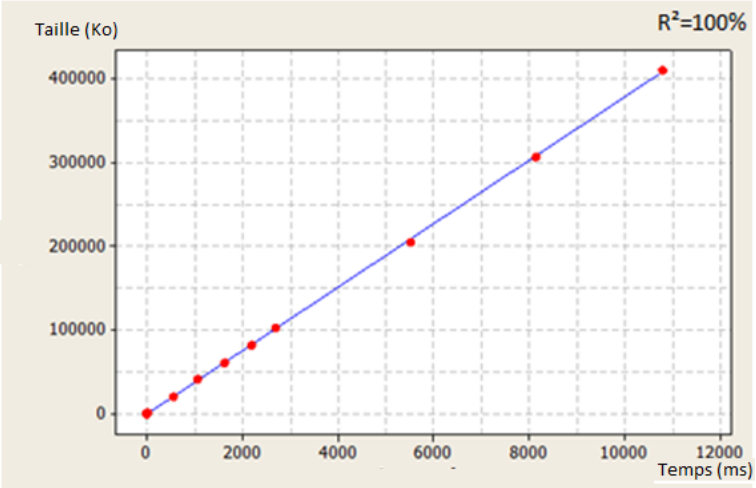


Figure 7-8: Régression en évitant la saturation de la RAM (58)

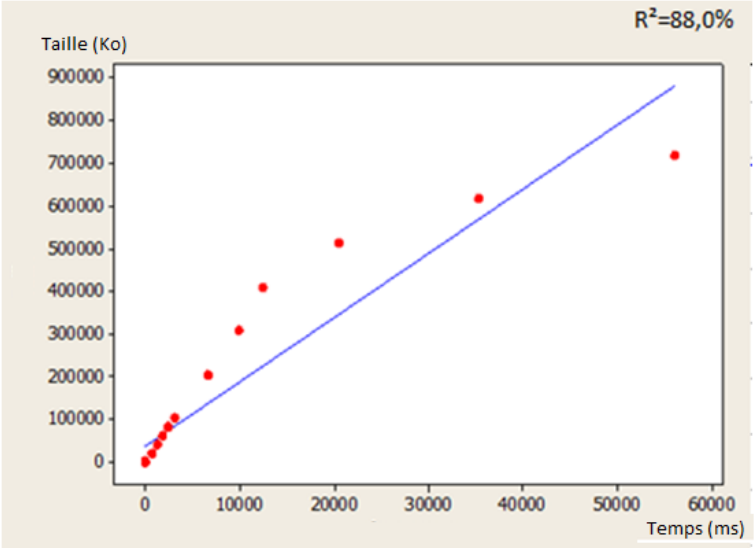


Figure 7-9: Comparaison de régression dans le cas de saturation de la RAM avec une régression linéaire (58)

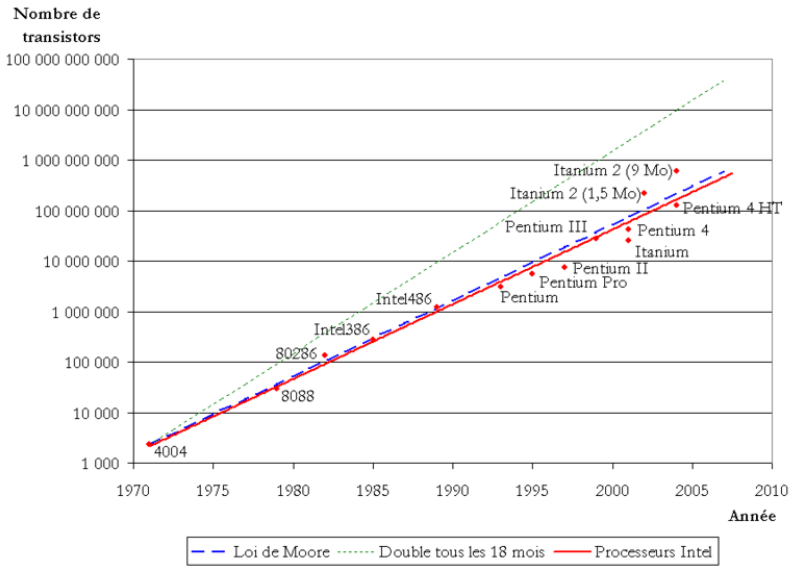


Figure 7-10: Vitesse des CPU selon le temps (61)

7.6 Conclusion

Beaucoup d'études ont abordé le problème des limitations de calcul dans les systèmes embarqués et ont montré le cout élevé pour garantir la sécurité dans ces systèmes.

En implémentant les deux algorithmes AES et RC5 dans des Smartphones Android et en étudiant les valeurs retournées, on a montré la capacité des architectures embarquées dans les Smartphones modernes à implémenter des schémas cryptographiques puissants pour sécuriser ces systèmes.

L'évolution des processeurs embarqués a donné une autre vue sur l'utilisation de ces systèmes, ils sont devenus capables de sécuriser les données des utilisateurs à condition de prendre en considération quelques paramètres comme le type de Processeur et des coprocesseurs intégrés, la taille de la mémoire cache, la taille de la RAM ainsi que la taille des données à traiter.

Conclusion Générale

La sécurité constitue une exigence importante dans les systèmes modernes de l'informatique embarquée. L'utilisation très répandue de ces systèmes dans des services comportant des informations sensibles, en conjonction avec leurs ressources limitées ont conduit à un nombre important d'attaques qui exploitent les caractéristiques innovantes de ces systèmes et entraînent une perte d'informations critiques. Le développement des systèmes embarqués sécurisés est un domaine émergent dans l'ingénierie informatique nécessitant des compétences dans la cryptographie, dans la technologie de communication, etc.

La diversité des systèmes embarqués a engendré une diversité de solutions pour résoudre le problème de sécurité, allant des modification au niveau physique tel que le rajout de coprocesseurs dédiés à la cryptographie, jusqu'au niveau application où on fait appel à des protocoles de sécurité implémentant différents algorithmes cryptographiques, et passant par les supports de communication qui doivent être temps réel dans la plus part des systèmes embarqués, et par le niveau système d'exploitation où sont implémentés des packages cryptographiques qui sont utilisés au niveau applicatif.

Dans ce mémoire (qui est organisé du niveau le plus bas au plus haut), nous avons mentionné les exigences de sécurité des systèmes informatiques embarqués et nous avons décrit les technologies qui sont plus critiques par rapport aux systèmes informatiques à usage générale. Nous avons aussi montré les approches de sécurité dans les systèmes embarqués aux différents niveaux.

L'implémentation de la sécurité au niveau des applications nécessite une analyse soigneuse et détaillée, c'est pour cette raison on a fait une étude sur la sécurité des Smartphones, en analysant leur système d'exploitation et en faisant une étude expérimentale de l'implémentation des algorithmes cryptographiques les plus puissants.

Dans notre travail, on a choisi de faire des études sur des dispositifs réels, plutôt que de faire des simulations afin d'avoir des résultats réels sur ce domaine.

Dans la partie pratique, nous avons choisi le système d'exploitation Android ainsi que les deux algorithmes AES et RC5 comme cible pour nos tests.

Pour l'implémentation de l'algorithme AES, nous avons travaillé avec des packages cryptographiques en Java et nous avons utilisé Eclipse comme outil de développement (après installation du plug-in Android).

Pour l'implémentation de l'algorithme RC5, nous avons plutôt choisi l'AGL WinDev Mobile afin que notre application soit multi-plate-forme, et ainsi pouvoir la tester sur plusieurs type de Smartphones (même ceux qui utilisent d'autre systèmes d'exploitation, tel que iPhone avec iOS).

Le domaine de la sécurité des Systèmes embarqués est très vaste et doit être décomposer selon les domaines d'application ; Dans notre étude, nous nous sommes concentrés sur les Smartphone à cause de leurs large déploiement.

Une de nos perspectives est de continuer à travailler sur les Smartphones en faisant des statistiques avancées sur le temps de calcul et la consommation d'énergie, afin d'établir un modèle mathématique (pour estimer les performances) sur lequel se baseront d'autres études des systèmes embarqués et étudier plus en détail les facteurs de sécurité des Smartphones. Aussi étudier plus en détail le système d'exploitation Android en faisant des modifications pour l'adapter à d'autres types de systèmes embarqués.

De toute évidence, le domaine technique des systèmes embarqués sécurisés est loin d'être mature. Les attaques et contre-mesures innovantes qui réussissent sont sans cesse émergentes, promettant un secteur attrayant et riche pour la recherche et le développement.

Bibliographie

1. **T Noergaard.** *Embedded Systems Architecture.* s.l. : Elsvier, 2005.

2. **Developpez.com.** Le dictionnaire des développeurs. [En ligne] 19 MAI 2005. [Citation : 10 Janvier 2013.] http://dico.developpez.com/html/1222-Systemes-systeme-embarque.php.

3. **CISCO.** *Cisco Visual Networking Index: Forecast and Methodology 2011-2016.* s.l. : CISCO, 2013.

4. **T Martin, M Hsiao, D Ha, J Krishnaswami.** Denial-of-service attacks on battery-powered mobile computers. *In Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04).* 2004.

5. **Y Li, CS Chen, YQ Song, Z Wang.** Real-time QoS support in wireless sensor networks: a survey. *7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems - FeT'2007* . 2007.

6. **P Ficheux.** *Linux embarqué.* s.l. : Eyrolles, 2011.

7. **A Silberschatz, P Galvin, G Gagne.** *Applied Operating System Concepts.* s.l. : Eyrols, 2001.

8. **P Ficheux, P Kadionik.** *Temps réel sous LINUX.* 2003.

9. **W BLACHIER.** *mémoire sur le temps réel.* s.l. : ENSIMAG, ENSIMAG, 2000.

10. **Conservatoire National des Arts et Métiers.** *Introduction aux systèmes de commande temps réel et aux réseaux de terrain.* 2008/2009.

11. **B Perens.** *The open source definition.* s.l. : Open sources: voices from the open source, 1999.

12. **LinuxDevices.** Embedded Linux market snapshot, 2005. *LinuxDevices.com.* [En ligne] 04 Mai 2005. http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Embedded-Linux-market-snapshot-2005/.

13. **LinuxDevices** . Snapshot of the Embedded Linux market. *LinuxDevices.com.* [En ligne] Mars 2004. http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Snapshot-of-the-Embedded-Linux-market-March-2004/.

14. **C Queinnec.** *Langage C.* s.l. : Technique de l'ingénieur, 2002.

15. **EtherCat.** L'Ethernet de terrain. *J'automatise.* Mai-Juin, 2005, Vol. 40.

16. **ScadaMobile**. *weetwilliamsl.com.* [En ligne] 2011. http://www.sweetwilliamsl.com/iweb/smhome.

17. **J Robert, J Georges, E Rondeau.** *Analyse de performances de protocoles temps-réel basés sur Ethernet.* Nancy France : Sixième Conférence Internationale Francophone d'Automatique CIFA 2010, 2010.

18. **G KONHEIM.** *COMPUTER SECURITY AND CRYPTOGRAPHY.* s.l. : Wiley, 2007.

19. **J Clercq.** Chiffrements symétrique vs asymétrique. [En ligne] 2008. http://www.itpro.fr/a/chiffrements-symetrique-vs-asymetrique/.

20. **X Ding, G Tsudik.** *Topics in Cryptology — CT- RSA.* s.l. : Springer, 2003.

21. **E Barker, W Barker, W Burr.** *Recommendation for Key Management – Part 1: General (Revision 3).* s.l. : NIST Special Publication 800-57, 2012.

22. **S Euschi.** *La sécurité des applications m-commerce.* s.l. : Mémoire de Magistère, Université de Ouergla, 2010.

23. **N Jansma, B Arrendondo.** *Performance Comparison of Elliptic Curve and RSA Digital Signatures.* s.l. : IEEE, 2004.

24. **J Eaves.** ECC and RSA speed comparison. *eaves.* [En ligne] Octobre 2004. http://www.eaves.org/blog/2004/04/ecc-and-rsa-speed-comparison.html.

25. **Wikipedia.** Signature numérique. *Wikipedia.* [En ligne] 2013. http://fr.wikipedia.org/w/index.php?title=Signature_num%C3%A9rique&oldid=88582908. 88582908.

26. **CISCO Networking Academy.** *CCNA Security 1.0 "Implementing Network Security".* s.l. : CISCO Systems, Inc., 2009.

27. **HARARI, S.** *Cryptographie et Procédés de Chiffrement.* s.l. : Hermes Science, 2008.

28. **Zabala, E.** *AES Animation.* Uruguay : Universidad ORT, 2003.

29. **W Wolf.** *Computers as Components - Principles of Embedded Computing Systems Design.* s.l. : Elsevier, 2000.

30. **W Miller, E Freeman.** *An experimental analysis of cryptographic overhead in performance critical systems.* s.l. : In Proceedings of the Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1999.

31. **S Ravi, P Kocher, R Lee, G McGraw, A Raghunathan.** *Security as a new dimension in embedded system design.* s.l. : In Proceedings of the 41st Annual Conference on Design Automation, 2004.

32. **D Abraham, G Dolan, G Double, J Stevens.** *Transaction security system.* s.l. : IBM Systems Journa, 1991.

33. **R Kuhn, M Anderson.** *Tamper resistance - autionary note.* s.l. : In Proceedings of the Second Usenix Workshop on Electronic Commerce, 1996.

34. **H Gary, M Greg.** *Exploiting Software: How to Break Codel.* s.l. : Addison-Wesley Professiona, 2004.

35. **M Robert.** *ASICs et logiciels CAO associés.* [Technique de l'ingénieur] 2006.

36. **H Tokioka, M Agari, M Inoue.** *Low power Consumption TFT-LCD with Dynamic Memory Embedded in Pixels.* s.l. : SID Symposium Digest of Technical Papers, 2012.

37. **authentic .** SafeXcel™-1741 Security Co-processor. *authentic* . [En ligne] 2012 . http://www.authentec.com/Products/EmbeddedSecurity/SecurityProcessors/SafeXcel-1741.aspx.

38. **Intel.** Unveiled: Smartphones with Intel Inside®. [En ligne] 2013. http://www.intel.com/content/www/us/en/smartphones/smartphones.html.

39. **Intel Ltd.** Speed and Security by Design. [En ligne] 2013. http://www.intel.com/design/intarch/celeron440/index.htm.

40. **S Ravi, P Kocher, R Lee, G McGraw, A Raghunathan.** *Security as a new dimension in embedded system design.* s.l. : In Proceedings of the 41st Annual Conference on Design Automation, 2004.

41. **R Anderson, M Kuhn.** *Tamper resistance - a cautionary note.* s.l. : In Proceedings of the Second Usenix Workshop on Electronic Commerce, 1996.

42. **R Venugopalan, P Ganesan.** *Encryption overhead in embedded systems and sensor network nodes: Modeling and analysis.* s.l. : ACM DL, 2003. 1-58113-676-5.

43. **P Kocher, R Lee, G McGraw, A Raghunathan.** *Security as a new dimension in embedded system design.* s.l. : Proceedings of the 41st annual Design Automation Conference, 2004.

44. **S Ravi, A Raghunathan, P Kocher.** *Security in embedded systems: Design challenges.* s.l. : ACM Transactions on Embedded Computing Systems (TECS), 2004.

45. **H Luo, G He, X Lin, X Shen.** *Towards Hierarchical Security Framework for Smartphones.* s.l. : 1st IEEE International Conference on Communications in China: Communications Theory and Security (CTS), 2012.

46. **D Jaggar.** *ARM architecture and systems.* s.l. : IEEE micro, 1997.

47. **ARMLtd.** *ARM7TDMI.* s.l. : Technical Reference Manual Revision: r4p1, 2004.

48. **ARMLtd** . ARM. *ARM.* [En ligne] ARMLtd. www.arm.com.

49. **ARMLtd** . Cortex-A53 Processor. *ARM.* [En ligne] 2013. http://www.arm.com/products/processors/cortex-a50/cortex-a53-processor.php.

50. **ARMLtd** . cortex-A57-processor. *ARM.* [En ligne] 2013. http://www.arm.com/products/processors/cortex-a50/cortex-a57-processor.php.

51. **ARMLtd** . Performance Improvement vs. Cortex-A15. *ARM.* [En ligne] 2013. http://www.arm.com/products/processors/cortex-a/cortex-a15.php.

52. **Authentec.** SafeXcel™ Security Processors. [En ligne] 2012. http://www.authentec.com/Products/EmbeddedSecurity/SecurityProcessors.aspx.

53. **Wikipedia.** Couche d'abstraction matérielle. *Wikipedia.* [En ligne] Decembre 2012. http://fr.wikipedia.org/w/index.php?title=Couche_d%27abstraction_mat%C3%A9rielle&oldid=86585334. 86585334.

54. **D Guignard, J Chable, E Robles.** *Programmation Android.* s.l. : Eyroles, 2010.

55. **F Garin.** *Développer des applications mobiles pour les Google Phones.* s.l. : Dunod, 2009.

56. **S Mongia, K Madisetti.** *Reliable Real-Time Applications on Android OS.* s.l. : IEEE Electrical and Computer, 2010.

57. **C Maia, L Nogueira, L Miguel.** *Evaluating Android OS for Embedded Real-Time Systems.* s.l. : cister.isep.pt, 2010.

58. **R Lebsir, M Benmohamed.** *Analyse Expérimentale d'Algorithmes Cryptographiques pour Systèmes Embarqués.* Annaba : International Conference on EMBEDDED SYSTEMS in TELECOMMUNICATIONS and INSTRUMENTATION (ICESTI'12), 2012.

59. **E jean-Bruno.** *Algorithmes de chiffrement.* s.l. : Centre de Ressource Informatique, 2005.

60. **W Freeman, E Miller.** *An experimental analysis of cryptographic overhead in performance - critical systems.* s.l. : In Proceedings of the Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1999.

61. **Wikipedia.** Loi de Moor. [En ligne] 2011. http://fr.wikipedia.org/w/index.php?title=Loi_de_Moore&oldid=88727503. 88727503.

62. **Motorola Mobility.** *Best practices for encryption in android.* s.l. : Motorola Mobility, 2012.

63. **Wikipedia.** POSIX. *Wikipedia.* [En ligne] 2013. http://fr.wikipedia.org/wiki/POSIX.

64. **J Lee, Y Su, C Shen.** *A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi.* s.l. : Industrial Electronics Society, 2007.

65. **EtherCatGroupTechnology.** *EtherCat, Technical Introduction and Overview.* 2004.