République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur de la **R**echerche **S**cientifique
Université **M**'hamed **B**ougara – **B**oumerdes

Faculté des Sciences
Département d'Informatique

Domaine : Mathématiques Informatique
Filière : Informatique Académique
Spécialité : Ingénierie du Logiciel et Traitement de l'Information

Master I
Rapport de projet tuteuré

Thème

Etude des attaques par injection de fautes contre les cartes à puce

Réalisé par :

MOKHTARI Tarek
AMZAL Hanane

Soutenu le 05.06.2014 Devant le jury composé de :

Mlle YAHIAOUI *Examinatrice.*

Mlle HAMADOUCHE *Promotrice.*

Année universitaire : 2013/2014.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

AES	Advanced Encryption Standard
AID	Application IDentifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
ATR	Answer To Reset
CAD	Card Acceptance Device
CAP	Converted APplet
CCD	Charge-Coupled Device
CISC	Complex Instruction Set Computer
COS	Chip/Card Operating System
CRC	Cyclic Redundancy Check
DES	Data Encryption Standard
DFA	Differential Fault Analysis
DPA	Differential Power Attack
EEPROM	Electrical Erasable Programmable Read Only Memory
EMA	ElectroMagnetic Attack
EMV	Europay Mastercard Visa
FIB	Focus Ion Beam
GSM	Global System for Mobile Communications
HTTPS	Hyper Text Transmission Protocol Service
ICAO	International Civil Aviation Organization
ISO	International Organization for Standardization
JCAPI	Java Card API
JCRE	Java Card Runtime Environment
JCVM	Java Card Virtual Machine
JVM	Java Virtual Machine
NFC	Near Field Communication
OSI	Open Systems Interconnection
P2P	Peer To Peer
PC	Program Counter
PIN	Personal Identification Number
PIX	Proprietary Identifier eXtension
RAM	Random Access Memory
RID	Ressource IDentifier
RISC	Reduced Instruction Set Computer
RMI	Remote Method Invocation
RNG	Random Number Generator

ROM	Read Only Memory
SCWS	Smart Card Web Server
SIM	Subscribe Identity Module
SP	Stack Pointer
SPA	Simple Power Attack
SRAM	Static Random Access Memory
SWP	Single Wire Protocol
TPDU	Transmission Protocol Data Unit
USB	Universal Serial Bus
UV	Ultraviolet

Introduction

Les cartes à puce ne sont pas que de simples supports de stockage, ce sont des ordinateurs aux capacités et de taille réduites, munis de systèmes d'exploitation tel que Java Card les rendant capables de charger plusieurs programmes issus ou non de leur constructeur. Ces cartes peuvent être transportés facilement et offrent un niveau de sécurité élevé, ce qui en fait un atout pour de nombreux domaines tels que la banque, les services du gouvernement, la téléphonie mobile, la télévision à péage, etc.

Comme les composants de sécurité des cartes à puce contiennent des informations confidentielles, elle font l'objet d'attaques. Ces attaques peuvent prendre le matériel en défaut (on parle d'attaques physiques), soit le l'applicatif en défaut (attaques logiques), soit en combinant les deux.

Dans le domaine de la sécurité informatique, lorsqu'une faille est découverte dans un système, la communauté scientifique ou industrielle met tout en œuvre pour fournir des mécanismes permettant de détecter son exploitation ou en la comblant. En faisant le rapprochement avec notre domaine d'activité, la faille serait l'attaque par injection de fautes, le système à protéger est la carte à puce, et dans l'ordre des choses il existe des mécanismes visant à enrayer la mise au point des attaques par injection de fautes ou à les détecter.

L'objectif de notre travail est de faire une étude sur les attaques par injection de fautes. Mais dans un premier temps nous allons présenter les cartes à puce et les différentes attaques existantes contre ces dernières.

Le présent document est composé de quatre chapitres.

On a commencé par introduire le domaine des cartes à puce dans le **chapitre 1** en présentant les différents concepts de base (typologie, standards, protocoles, etc).

Par la suite, on a présenté la plateforme Java Card dans le **chapitre 2** en mettant l'accent sur l'aspect sécurité.

Dans le **chapitre 3** nous avons présenté une classification des attaques existantes contre les cartes à puce(Attaques physiques, logiques et combinées) en mettant l'accent sur les attaques physiques. Nous avons aussi catégorisé ces attaques physiques selon deux critères (leur caractère intrusif et l'activité de l'attaquant).

Pour finir nous avons présenté les attaques par injection de fautes, sujettes de notre travail, dans le **chapitre 4**, en essayant de couvrir les points les plus importants (les techniques d'injection de fautes, les types et les modèles de fautes, les contremesures existantes,etc).

Chapitre 1

La Carte à Puce

Sommaire

1.1 Qu'est ce qu'une carte à puce ?

Une carte à puce est un support électronique, portable et sécurisé en vue de conserver des données personnelles [11]. Pratiquement, c'est une carte en matière plastique (figure 1.1) de quelque centimètres de coté et moins d'un millimètres d'épaisseur, incorporant un circuit électronique de taille très réduite. Ce circuit peut contenir un microprocesseur, capable de traiter des informations, généralement sensibles, de façon sécurisée, ou être limité à des circuits de mémorisation non volatile et, éventuellement, un composant de sécurité.

Bien que les ressources d'une carte à puce soient toujours limitées, en terme d'espace mémoire et capacité de calcul, ses récentes générations, qualifiées de cartes intelligentes (*smart card* en anglais), sont capable d'exécuter du code et d'héberger diverses applications se rapprochant ainsi d'un ordinateur personnel.

FIGURE 1.1 – Une carte à puce

1.2 Historique

Il y a environ quarante ans que le premier prototype de carte à puce a été conçu ; cela n'empêche pas la carte d'être un objet méconnu du grand public. Toutefois, elle a toujours été considérée comme étant un équipement électronique à la pointe de la technologie. En effet, depuis les années 70, les technologies intégrées à la carte à puce reflètent les dernières avancées en matière de microprocesseurs de même qu'en matière du nombre et de la diversité de ses applications.

Plusieurs dates s'inscrivent dans le livre d'or de l'histoire de la carte à puce, ainsi que quelque illustres inventeurs ayant marqué cette industrie de leur empreinte. Le tableau suivant (Tableau 1.1) en est un résumé.

Année	Évènement
1968	Les deux inventeurs allemands Jürgen Dethloff et Helmut Gröttrup introduisent en premier un circuit intégré dans une carte plastique.
1970	Le chercheur japonais Kunikuta Arimura de l'Institut de Technologie Arimura dépose un brevet sur la carte à puce.
1974 - 1978	Le Français Roland Moreno dépose 47 brevets (dans 11 pays) sur la carte à puce. Il a créé une mémoire portative dotée de moyens de protection (matériels et logiciels) pour restreindre l'accès en lecture et en écriture.
1977	Dethloff dépose un brevet où il introduit une carte à mémoire portative dont les moyens de protection sont constitués par un microprocesseur.
1979	En France, la première carte programmable à microprocesseur (nommée CP8) est créée et assemblée à Toulouse par Motorola pour le groupe Bull.
1983 - 1984	Arrivée des premières cartes téléphoniques et cartes bancaires en France et en Allemagne.
1984 - 1987	Introduction des premières normes ISO (sous la référence 7816) en Europe, régissant le domaine des cartes à puce.
1991	Lancement des premiers réseaux et services GSM et des premières cartes SIM (Subscribe Identity Module).
1994	Arrivée de la première carte à puce sans contact (carte MIFARE de la société Mikron).
1995	Apparition de la carte EMV (Europay, Visa MasterCard), une carte de paiement portant un niveau de sécurité jamais atteint, et offrant des capacités multi-applicatives mais encore mono-opérateur.

1996	Conception de la toute première carte Java par les équipes de Bertrand du Castel au sein de Sun Microsystems. Cette carte adopte le choix de l'environnement ouvert pour anticiper l'avènement des cartes multi-applicatives et multi-opérateurs.
2002	Philips et Sony développent la technologie NFC (Near Field Communication), la possibilité de transformer une carte en lecteur et inversement, et de réaliser des échanges P2P (Peer To Peer). Apparition des "*smart objects*" capables de marier le sans contact de proximité et le sans fil longue distance ou des accès Internet (des téléphones mobiles et des clés USB en particulier).
2004	Avènement des e-passeports ICAO (International Civil Aviation Organization), et mettre au point le "match-on-card" : comparer une empreinte stockée à celle acquise sur un capteur extérieur.
2007	Arrivée des cartes SIM-USB-SWP-SCWS. SWP pour Single Wire Protocol ; permet à la carte de communiquer sur un seul fil avec un circuit NFC dans un "*smart object*", et utiliser les autre contacts pour connecter un lien de type USB avec le processeur central de ce "*smart object*", la carte se transforme alors en serveur web (SCWS : Smart Card Web Server).

TABLEAU 1.1 – Historique de la carte à puce
[11][21][7]

1.3 Typologie des cartes à puce

Les différentes cartes à puce existantes peuvent être classées soit suivant les technologies utilisées en interne (leurs composants constitutifs), soit suivant les possibilités de communication (leur interface). La figure suivante (Figure 1.2) résume cette classification.

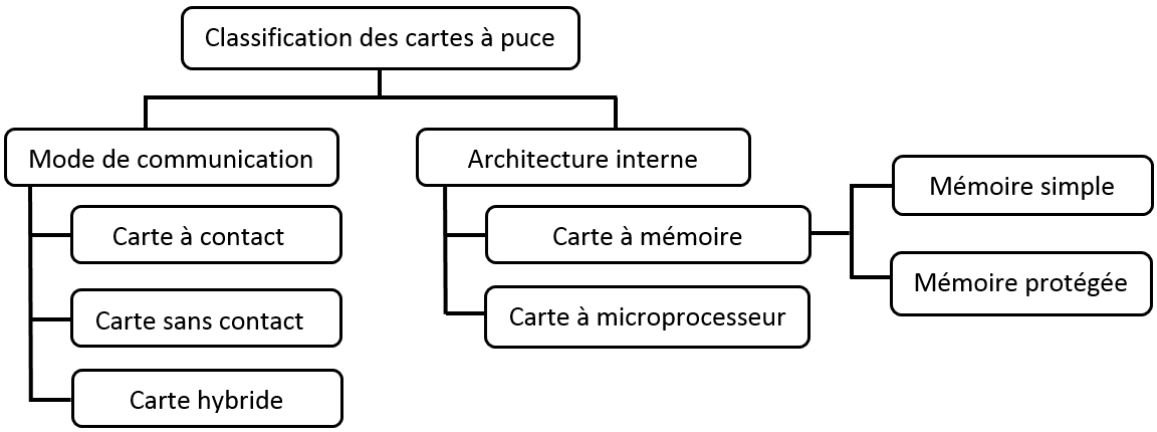

FIGURE 1.2 – Typologie des cartes à puce

1.3.1 Selon les composants

1.3.1.1 La carte à mémoire

Elle constitue la famille la plus ancienne des cartes à puce. Ne contenant que de la mémoire, de telles cartes sont limitées à des applications relativement simples. Cette catégorie de cartes peut être scindée à son tour en deux sous-catégories [11] :

- Les cartes à **mémoire simple** ne contiennent qu'une zone mémoire et un minimum de logique nécessaire pour pouvoir y accéder (Figure 1.3). Le niveau de sécurité offert par ces cartes est très faible et ne peut reposer que sur des artifices relativement simples (la mémoire peut être lue sans protection, cependant, l'écriture peut-être rendue impossible [19]).

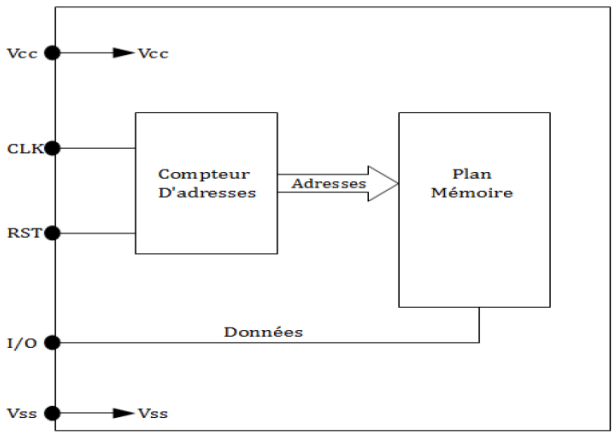

FIGURE 1.3 – Architecture interne générale d'une carte à mémoire simple [11]

- Les cartes à **mémoire protégée** associent de la mémoire, dont certaines zones sont accessibles seulement en lecture ou seulement en phase de personnalisation de la carte, et de la logique câblée non programmable (instruction programmée une fois pour toutes pour une application particulière) permettant l'exécution d'automates simples allant jusqu'à la présentation de mots de passe ou autres succédanés de codes PIN. Il s'agit typiquement des télécartes contenant un compteur d'unités, un numéro de série et une donnée secrète permettant d'authentifier la carte [11]. Ce type de carte ne possède pas de capacité propre de calcul, est aisément duplicable, et est très fortement dépendant du lecteur. [21]
  Bien que les cartes à mémoire protégée soient plus sûres que celles à mémoire simple, elles ne permettent pas la mise en place des applications les plus complexes que sont les cartes bancaires, les cartes SIM ou bien encore les cartes de décryptage TV. Il faut en effet pour cela que la carte dispose d'une "intelligence" locale que n'ont pas les cartes à mémoire.

1.3.1.2 La carte à microprocesseur

Ce type de carte constitue réellement ce que l'on peut appeler une *smart card*. En effet, la puce de cette carte embarque un microprocesseur, ce qui la rend autonome (c.-à-d. intelligente). Les dimensions de la puce sont au maximum de 25 mm^2 pour sa surface et de 200 micromètres pour son épaisseur [21]. D'une part, ces dimensions sont imposées par les contraintes de flexion induite par le support en plastique. Et d'autre part, cette dimension limitée réalise un compromis entre sécurité physique et complexité du composant.

En effet, c'est l'association en un seul circuit [11][21](Figures 1.4 et 1.5) :

- d'un **microprocesseur** : de 8, 16 ou 32 bits avec des architectures CISC [1] (Complex Instruction Set Computer) ou RISC [2] (Reduced Instruction Set Computer) travaillant à des fréquences internes comprises entre 5 et 40 MHZ.
  Le standard ISO 7816 oblige à respecter une certaine plage de fréquences d'horloge afin d'assurer la compatibilité avec le matériel déjà en place (lecteurs). Pour cette raison, les fabricants utilisent des multiplicateurs de fréquence en interne afin d'effectuer les opérations dans la carte plus rapidement. Actuellement, il existe ainsi des cartes à microprocesseur de dernière génération RISC fonctionnant avec une horloge interne de 100 à 200 MHz en utilisant un multiplicateur x20 ou x40.
- d'une **mémoire morte** (ROM pour Read Only Memory) : c'est une mémoire persistante et figée (en usine). Elle n'a donc pas besoin d'énergie pour sauvegarder l'information qu'elle contient et son contenu (qui est constitué du système d'exploitation [3] ainsi que des données permanentes) n'est pas modifiable. Sa taille varie de 32 Ko jusqu'à 256 Ko et même plus pour les cartes haut de gamme.
- d'une **mémoire vive** (RAM pour Random Access Memory) : cette mémoire est utilisée comme espace de stockage temporaire pour les données lors de leur traitement. Ceci est dû à sa rapidité d'accès (essentielle pour fournir rapidement les données au microprocesseur), et à son caractère non persistent. Sa capacité va de 1 ko à 24 ko (pour les cartes haut de gamme).
- d'une **mémoire EEPROM** (Electrical Erasable Programmable Read Only Memory) : C'est une mémoire persistante avec les mêmes caractéristiques que la ROM, sauf que son contenu peut être modifiable. Elle est chargée du stockage des informations qui peuvent évoluer dans le temps tels que les applications, les clefs cryptographiques et le code PIN de l'utilisateur ; c.-à-d. les informations qui doivent être conservées même lorsque la carte n'est plus sous tension. Sa capacité varie de 16 ko à 256 ko.
- d'une **interface d'entrée/sortie série** : Pour les échanges de données.
- de toute **la logique** nécessaire pour faire fonctionner l'ensemble.

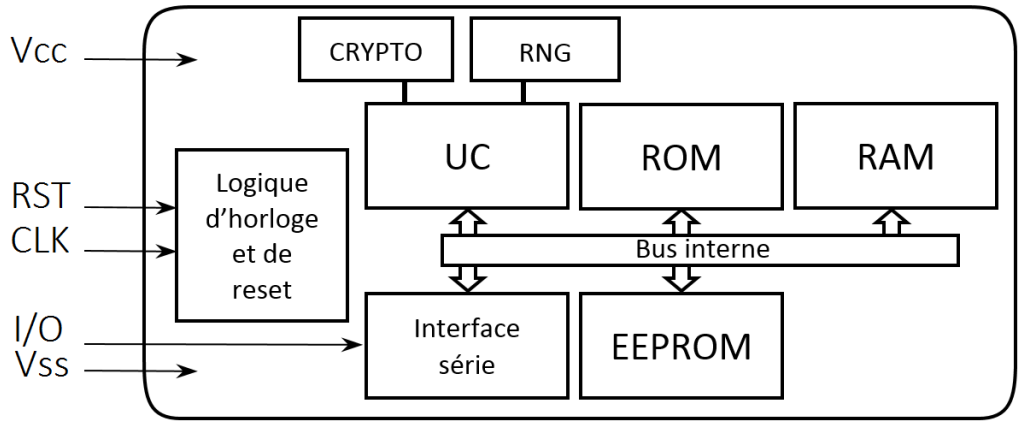

FIGURE 1.4 – Architecture simplifiée d'une carte à microprocesseur [11][21]

La puce possède éventuellement un coprocesseur cryptographique qui réalise les opérations de chiffrement et déchiffrement de façon matérielle ce qui permet d'améliorer les performances. Ce

1. Un microprocesseur à jeu d'instructions étendu.
2. Un microprocesseur à jeu d'instructions réduit.
3. COS ou Card Operating System.

coprocesseur cryptographique est associé à un générateur de nombres aléatoires RNG (Random Number Generator).
Les cartes à microprocesseur conviennent aux applications les plus sensibles où le degré de sécurité des données est le facteur prédominant : contrôle d'accès sécurisé, carte bancaire, télécommunication, etc.
Dans tout le reste du document, l'utilisation du terme "carte à puce" fera référence à ce type de cartes i.e. les cartes à microprocesseur.

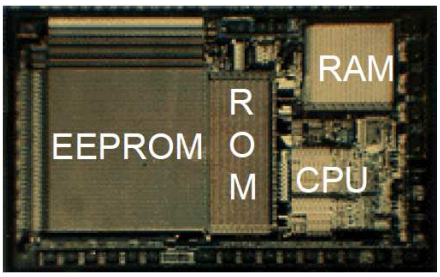

FIGURE 1.5 – Puce d'une carte à microprocesseur : le chip 21 (1988)
[23]

1.3.2 Selon l'interface

1.3.2.1 La carte à contact

Pour interagir avec un environnement extérieur, la carte à contact doit être insérée dans un lecteur (appelé aussi CAD pour Card Acceptance Device)[11]. Elle communique par un microcontact relié à la puce (microchip) par des fils d'or. Cet assemblage se nomme micromodule [21] (Figure 1.6). Cette carte utilise une communication série avec huit contacts (C1-C8) définis dans le standard ISO 7816, et dont l'utilité est résumée dans le tableau suivant (Tableau 1.2).
N'ayant pas d'alimentation électrique propre, et pour pouvoir fonctionner, ce type de cartes doit être inséré dans un lecteur qui lui fournit l'énergie nécessaire à son fonctionnement.

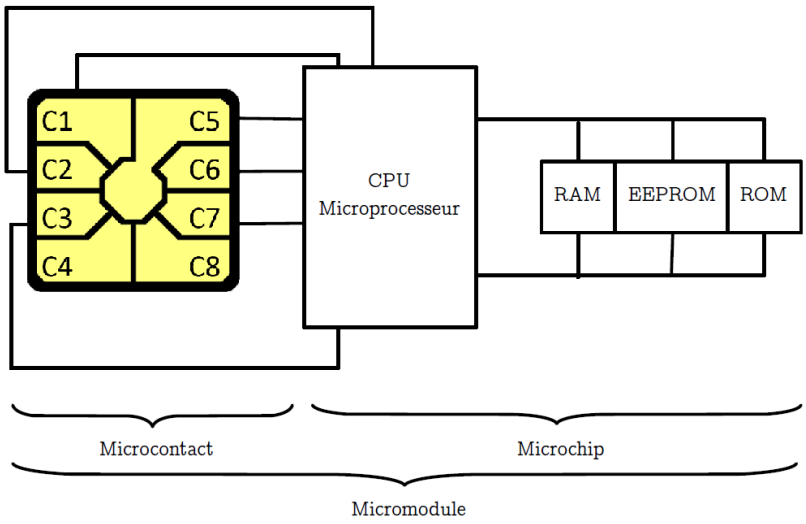

FIGURE 1.6 – Le micromodule
[21]

Contact	Appellation	Utilisation
C1	Vcc	tension d'alimentation positive de la carte
C2	RST	commande de remise à zéro
C3	CLK	horloge fournie à la carte
C5	GND	masse électrique
C6	Vss	tension de programmation (n'est plus utilisée)
C7	I/O	entrée/sortie de données (bidirectionnelle)
C4 et C8	RFU	à l'origine, contacts réservés pour utilisation future mais actuellement ils servent à communiquer en USB (Universal Serial Bus)

TABLEAU 1.2 – Utilisation des contacts selon la norme ISO 7816-2
[11]

1.3.2.2 La carte sans contact

Les cartes sans contact communiquent grâce à une antenne reliée à la puce. Cette carte n'utilise pas de contact physique avec le lecteur. Plusieurs technologies existent, mais de façon générale, la carte sans contact contient une puce électronique avec un émetteur hyperfréquence et une antenne intégrée dans le plastique [11] (Figure 1.7). La portée entre la borne de lecture et la carte est limitée (environ 10 cm), et le temps de transaction ne doit pas excéder les 200 millisecondes, ce qui limite la taille des données échangées [21].
L'énergie, nécessaire à la puce, provient soit d'un couplage capacitif qui consiste par exemple en une batterie, soit d'un couplage inductif distant collecté par l'antenne.
Certaines cartes sans contact utilisent les ondes radio et peuvent donc fonctionner à plus grande distance. Cette carte est un peu plus complexe car elle doit intégrer un régulateur de tension, un modulateur/démodulateur ainsi qu'un mécanisme d'anti-collision, un générateur d'horloge et bien entendu une antenne [11].
Les cartes sans contact sont privilégiées dans le domaine du transport ainsi que pour le contrôle d'accès aux bâtiments, domaines dans lesquels les transactions doivent être faites à une vitesse assez élevée.

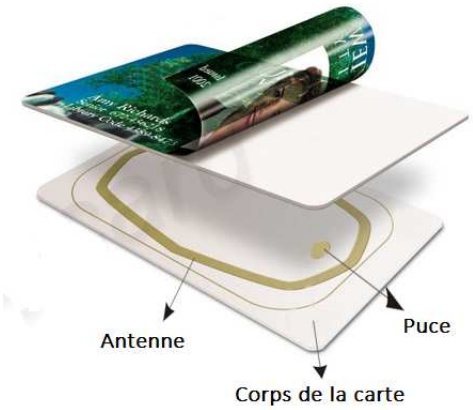

FIGURE 1.7 – Carte à puce sans contact

1.3.2.3 La carte hybride

Une carte à interface hybride est simplement une carte combinant les technologies sans contact et celles avec contact. Les technologies étant différentes, une puce ne peut pas être à la fois avec et sans contact. De ce fait les cartes hybrides embarquent deux puces, la première reliée aux contacts, la deuxième à l'antenne (Figure 1.8).
Ces cartes sont le meilleur compromis car elles offrent les avantages des deux types de carte à puce mais leur prix est beaucoup plus élevé.

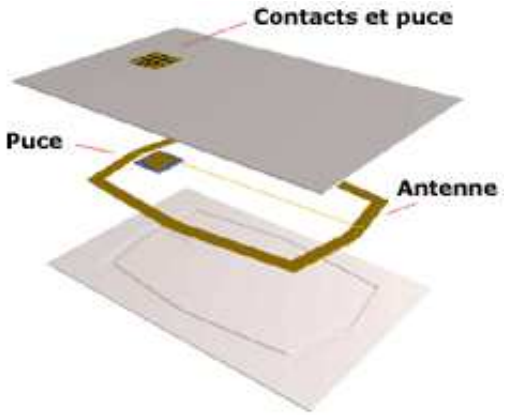

FIGURE 1.8 – Carte à puce hybride

1.4 Systèmes d'exploitation pour cartes à puce

Les cartes à microprocesseur sont supportées par un puissant système d'exploitation appelé "COS" (Chip Operating System ou Card Operating System). Cependant, vu les ressources limitées des cartes à puce, leur système d'exploitation doit être le plus léger possible, tout en étant riche, afin qu'il puisse être embarqué dans sa mémoire. A ce niveau là, trois sous-familles de cartes peuvent être distinguées [11] :

1.4.1 Les cartes à puce *"spécifiques"*

Dans lesquelles le contenu de la majeur partie du système d'exploitation est définie par le fabriquant de la carte. En respectant les normes définies dans les standards, ce dernier peut définir ses propres instructions propriétaires, ses propres fichiers avec le contenu de son choix. Un tel développement conduit à écrire le programme i.e. le système d'exploitation, qui est ensuite programmé par masque dans le microcontrôleur pour produire des cartes "sur mesure". Ce type de cartes est réservé aux grands groupes industriels, commerciaux ou financiers puisque cette approche impose de produire la carte en très grande série afin d'amortir les coûts d'étude et de développement qui sont très élevés.

1.4.2 Les cartes à puce *"personnalisables"*

Elles contiennent un système d'exploitation non modifiable, programmé par le fabriquant. ce système supporte un certain nombre de commandes, définies dans les standards, mais bien souvent complétées par des commandes propriétaires crées par le fabriquant de la carte et adaptées aux différentes applications pouvant être visées par la carte. Lorsqu'elles sont fournies par le fabriquant,

certaines parties de la mémoire sont accessibles afin de définir le comportement global de la carte, les propriétés des fichiers, etc. C'est le stade de la "personnalisation". Ceci revient au développement de l'application visée par la carte. Cette personnalisation est d'autant plus souple que le fabriquant aura prévu de nombreux fichiers de types différents et un jeu d'instruction riche. Une fois cette phase de personnalisation achevée, il y a une opération de "verrouillage" du contenu de la carte afin de le rendre impossible à modifier par la suite. A ce niveau là, la carte peut être mise en circulation afin de faire fonctionner l'application.

1.4.3 Les cartes à puce à *"Système ouvert"*

Pour ces cartes, le développement des applications est plus simple car il se fait en langage évolué (Java, Basic, C) et non pas en langage machine. Ainsi le programme obtenu est traduit en un code intermédiaire qui sera chargé dans la carte puis exécuté par le microcontrôleur qui est doté d'un interpréteur du code intermédiaire. Dans un tel schéma, l'évolution du code est désormais possible (à l'inverse des deux types de système précédents où ce programme contenait à la fois le système opératoire et l'application rendant ainsi toute évolution du code difficile voire impossible). Ceci revient à dire que les cartes à système ouvert autorisent le chargement des applications après délivrance de la carte et peuvent même héberger plusieurs applications différentes (dans le cas des cartes multi-applicatives). Il existe sur le marché plusieurs Systèmes ouverts et qui sont totalement incompatibles entre eux. Parmi les systèmes existants : MULTOS, Basic Card et Java Card.
Le chapitre suivant (2) est dédié pour donner une présentation détaillée de Java Card.

1.5 Standards de normalisation

Les cartes à puces sont très standardisées car elles doivent être utilisables avec la gamme la plus large possible de lecteurs dans le monde entier. D'ailleurs l'intérêt majeur d'une carte à puce aujourd'hui est que son niveau de normalisation est remarquable. À ce propos, la normalisation ne concerne pas la puce seulement, mais aussi les dimensions **physiques** de la carte, les paramètres **électriques**, ainsi que les paramètres **logiciels**.
Les principaux standards en matière de carte à puce sont le fruit des travaux de l'ISO. Le tableau ci-dessous (Tableau 1.3) en est un résumé.

Norme ISO	Titre officiel de la norme
Normes relatives aux cartes en général	
	Cartes d'identification :
ISO 7810	- Caractéristiques physiques
ISO 7811-1	- Techniques d'embossage
ISO 7811-2	- Techniques d'enregistrement magnétique
ISO 7811-3	- Emplacement des caractères embossés sur les cartes de type ID 1
ISO 7811-4	- Position des pistes magnétiques à lecture seule - Pistes 1 et 2
ISO 7811-5	- Position des pistes magnétiques à lecture/écriture - Piste 3
ISO 7812-1	- Identification de l'émetteur - système de numérotation
ISO 7813	- Cartes pour transactions financières
Normes relatives aux cartes à puce avec contact	
	Cartes d'identification - Cartes à circuits intégrés avec contacts :
ISO 7816-1	- Caractéristiques physiques
ISO 7816-2	- Dimension et position des contacts
ISO 7816-3	- Signaux électroniques et protocoles de transmission

ISO 7816-4	- Commandes inter-industries
ISO 7816-5	- Enregistrement des fournisseurs d'applications
ISO 7816-8	- Commandes pour des opérations de sécurité
ISO 7816-9	- Commandes pour la gestion des cartes
ISO 7816-10	- Signaux électroniques et réponse à la mise à zéro des cartes synchrones
ISO 7816-11	- Vérifications personnelles par méthodes biométriques
ISO 7816-15	- Application des informations cryptographiques
Normes relatives aux cartes à puce sans contact	
	Cartes d'identification - Cartes à circuits intégrés sans contacts :
ISO 14443-1	- Caractéristiques physiques
ISO 14443-2	- Interface radio fréquence et des signaux de communication
ISO 14443-3	- Initialisation et anti-collision
ISO 14443-4	- Protocole de transmission

TABLEAU 1.3 – Principales normes relatives aux cartes à puce
[11]

1.6 Protocoles de communication

Pour que la carte à puce communique avec le monde extérieur, elle doit être placée dans un lecteur (carte avec contact) ou à proximité de ce dernier (carte sans contact). Le lecteur est connecté lui même à un autre ordinateur (hôte).
La communication entre la carte et le lecteur se fait suivant le modèle client/serveur où le lecteur joue le rôle de client et c'est d'ailleurs ce dernier qui fournit la carte en énergie. La communication client/serveur se fait au travers d'une pile protocolaire que l'on retrouve dans la figure 1.9 et dont les différentes couches s'alignent au mieux sur le modèle OSI. Dans ce modèle, à chaque couche correspond un protocole de communication.[21]

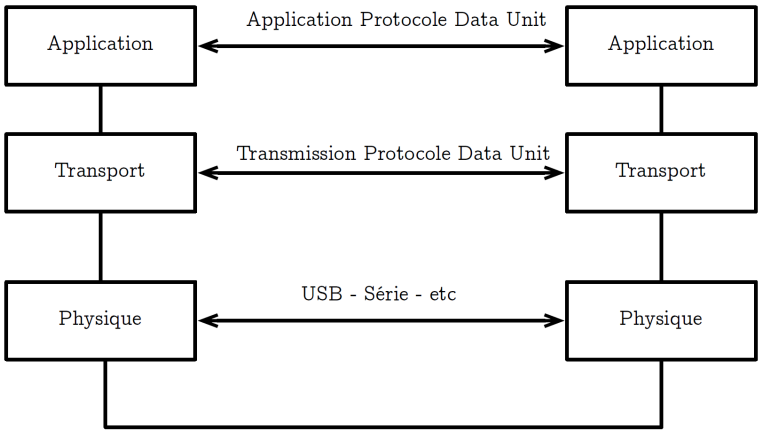

FIGURE 1.9 – Pile de communication entre le lecteur et la carte
[21]

La couche physique : La couche physique est normalisée par l'ISO 7816-3 (qui définit les signaux électroniques et les protocoles de transmission). Elle définit notamment une fréquence d'horloge

comprise entre 1 MHz et 5 MHz, ainsi qu'une vitesse pour les communications pouvant aller jusqu'à 115200 bauds.

La couche transport - TPDU : Le protocole utilisé pour le transport des données est le protocole TPDU (Transmission Protocol Data Unit). Tous les échanges de données pour ce protocole sont également définis dans l'ISO 7813-3. Ce protocole définit deux modes principaux de transmission qui sont :

- le T=0 qui utilise une transmission par octet.
- le T=1 qui utilise une transmission par paquet.

Il y a aussi certaines cartes qui utilisent le T=14 qui est réservé aux protocoles propriétaires. On trouve également des cartes qui utilisent T=CL pour les communications sans contact.
Cette norme définit aussi la sélection du type de protocole si plusieurs protocoles sont disponibles ainsi que le format de **ATR** (Answer To Reset) qui est le message envoyé par la carte au lecteur juste après sa réinitialisation afin de se synchroniser pour les échanges futurs. C'est un paquet de données contenant la description des paramètres de transmission entre autre : protocole de transport supporté par la carte, vitesse de transmission des données, paramètres matériels de la carte, etc.

La couche application - APDU : Le protocole ici utilisé échange des données au format APDU (Application Protocol Data Unit) qu'on retrouve également normalisé dans l'ISO 7816-4. Il existe principalement deux types d'APDU utilisées comme une paire pour chaque échange (Tableaux 1.4 et 1.5) :

Entête obligatoire				Corps optionnel		
CLA	INS	P1	P2	Lc	Champ de données	Le

TABLEAU 1.4 – APDU de commande

Corps optionnel	En-queue obligatoire	
Champ de données	SW1	SW2

TABLEAU 1.5 – APDU de réponse

- les commandes APDU sont émises par le CAD [4] en direction de la carte.
- les réponses APDU sont émises par la carte en direction du CAD.

La commande APDU est constituée (voir le Tableau 1.4) d'un entête et d'un corps. L'entête obligatoire contient les quatre champs (un octet chacun) suivants :

- **CLA :** l'octet de classe qui identifie la catégorie de la commande et de la réponse APDU, correspond généralement à un domaine d'application donné (par exemple, CLA = A0h pour les commandes des cartes SIM).
- **INS :** indique l'instruction à exécuter.
- **P1** et **P2** : les paramètres de l'instruction.

Le corps de la commande est variable et peut être omis, il contient les champs suivants :

- **Lc :** spécifie la taille du champ de données.
- Le **Champ de données** qui contient les données à envoyer à la carte pour exécuter l'instruction spécifiée dans l'entête (INS).
- **Le :** nombre d'octets attendu par le CAD pour le champ de données de la réponse APDU de la carte.

4. Card Acceptance Device : Lecteur de cartes.

La réponse APDU est constituée des éléments du Tableau 1.5, donc d'un corps optionnel de taille variable :

– Le **champ de données** de taille Le déterminée dans la commande APDU correspondante.

Et une zone de fin obligatoire comportant deux octets :

– **SW1** et **SW2** (Status Word), deux champs d'un octet précisant l'état de la carte après exécution de la commande APDU. Par exemple, SW1 = 90h et SW2 = 00h signifie que l'exécution s'est déroulée jusqu'à son terme et avec succès.

1.7 Domaines d'application

La carte à puce a désormais pris une grande place dans notre vie quotidienne, du fait qu'elle est utilisée dans plusieurs domaines où ses capacités de stockage sécurisé des informations sont mises en valeurs. Parmi ces applications on peut citer :

- **Applications monétaires :**
  - Carte bancaire.
  - Prépaient de télécommunications.
  - Porte monnaie électronique.
  - Titres de transport.
  - télévision à péage, etc.
- **Applications d'identification :**
  - En entreprise : contrôle d'accès au locaux, contrôle horaire.
  - Sur un réseau téléphonique : carte SIM.
  - Dans le secteur du gouvernement : Cartes d'identité nationales, E-passeports, carte de santé, etc.
- **Applications de sécurisation :** la carte est utilisée comme un support de stockage des clefs cryptographiques servant à l'authentification et la sécurisation des communications :
  - contrôle d'accès logique à un serveur par authentification.
  - protection des messages et documents numériques par signature électronique et chiffrement des données.
  - sécurité des transactions sensibles, etc.

1.8 Conclusion

Les cartes à puces sont de plus en plus présentes autour de nous, elles ont leur place dans la vie de tous les jours. Le nombre d'applications spécifiques à ce support se multiplie de jour en jour. Pour les programmeurs il est nécessaire de pouvoir utiliser un langage évolué et objet pour remplacer le C ou l'assembleur. C'est pourquoi JavaCard est un standard qui tend à devenir une référence.

Dans le chapitre suivant nous allons présenter la plateforme JavaCard en détail.

Chapitre 2

La plateforme Java Card

Sommaire

2.1 Présentation

Toutes les cartes à puces sont caractérisées par des ressources très limitées disponibles pour l'exécution d'applications. En outre, l'explosion des réseaux de télécommunication et des transactions électroniques ont augmenté le besoin en applications sans négliger la sécurité.

Aujourd'hui, le moyen le plus sûr pour assurer un niveau de sécurité satisfaisant reste la carte à puce. Cependant, le développement d'applications pour carte à puce a toujours été difficile et réservé à des experts.Il a donc fallut développer un langage qui soit à la fois fiable, robuste, peu gourmand en ressources et bien sûr simple. C'est en 1996 que Java Card a été proposé .

La plateforme Java Card a pour but de faire fonctionner la technologie Java sur des équipements fortement contraints tels que les cartes à puce ou d'autres équipements avec peu de mémoire et peu de puissance de calcul. Une Java Card est donc une carte à puce sur laquelle on est capable de charger et d'exécuter des applications Java du type applets ou servlets (depuis la version 3.0 de la spécification). Contrairement aux cartes à puce traditionnelles, ce sont des cartes ouvertes, c.-à-d. que les programmes qui y sont contenus ne sont pas nécessairement fournis par l'émetteur de la carte [21].

La technologie Java Card peut être considérée comme étant une plateforme fournissant un environnement sécurisé pour cartes à puce, interopérable et multi-applicatif qui jouit des avantages du langage Java [21].

2.2 Historique

- En 1996, Shlumberger à Austin au Texas un groupe d'ingénieurs cherchent à simplifier le modèle de programmation existant pour cartes à puce tout en préservant sa sécurité en utilisant une plateforme simple et universel à l'époque était le langage Java qui vérifier ces conditions. Mais, en raison de la faible puissance des cartes à puce, ils décident alors d'adapter la plateforme Java et de n'utiliser qu'un sous-ensemble de Java. Ainsi, la machine virtuelle Java communément appelée JVM3 ainsi que le système de runtime 4 ne devait pas dépasser les 12 ko. C'est ainsi l'initialisation de Java Card 1.0.
- En février 1997, Bull et Gemplus se joignent à Schlumberger pour fonder le Java Card Forum. Il a pour but de promouvoir des API 5 Java Card afin de permettre son adoption en masse comme standard pour l'industrie de la carte à puce.
- En novembre 1997, Sun Microsystems fournit la spécification 2.0 de Java Card qui consiste en un sous-ensemble du langage et de la machine virtuelle Java. Cette spécification définit les concepts de base de la programmation et des API très différents de ceux de la version de Schlumberger (version 1.0). Mais il n'y a encore rien sur le format des applets à charger sur la plateforme.
- En mars 1999, Java Card 2.1 est sorti, composer de trois sous ensembles :
  - une spécification pour les APIs Java Card ; la cryptographie et la gestion des exceptions.
  - une spécification pour l'environnement d'exécution des applets ; la standardisation.
  - une spécification pour la machine virtuelle ; la définition explicite de la machine virtuelle Java Card (JCVM) et du capfile (fichier binaire contenant l'application java compilée).
- En juin 2002, Java Card 2.2 a été publiée par Sun Microsystems dont les principales nouveautés sont la prise en charge des canaux logiques ainsi que l'invocation de méthodes à distance (RMI), l'ajout de quelques éléments sur l'installation et l'effacement des applications sur la carte et quelques nouveaux algorithmes cryptographiques.
- En octobre 2003, Java Card 2.2.1 est publiée, cette dernière corrige l'API de la version précédente et apporte quelques clarifications.
- En mars 2006, Java Card 2.2.2 qui n'apporte pas de changement majeur mis à part quelques nouveaux algorithmes cryptographiques et une clarification des API.
- En 2008,Java Card 3.0 arrive avec des spécifications qui apporte une révolution dans les spécifications, car elle introduit deux types de spécification Java Card :
  - Java Card 3 Classical Edition qui est juste une évolution de la Java Card 2.2.2 faite pour fonctionner avec des cartes ayant de faibles ressources. Elle contient les fonctionnalités nécessaires aux supports des applets.
  - Java Card 3 Connected Edition qui introduit de vraies nouveautés, car en plus des applets classiques, elle supporte un nouveau type d'application : les servlets [1]. Elle nécessite donc des cartes avec des contraintes de ressources moins fortes (cartes modernes haut de gamme). Elle introduit un quatrième sous-ensemble dans la spécification celui des servlets. Ainsi pour cette édition, on a :
    - Java Card Runtime Environment Specification, Connected Edition.
    - Java Card Application Programming Interface Specification, Connected Edition.
    - Java Card Virtual Machine Specification, Connected Edition.

1. Les servlets sont des applications web qui nécessitent d'avoir un serveur web embarqué dans la carte.

– Java Card Servlet Specification, Connected Edition [21].

2.3 Avantages de Java Card

La plateforme Java Card présente des avantages pour les développeurs des applications pour les carte à puce, ces avantages sont résumés dans les points suivant :

1. *La simplicité* : la plateforme Java Card introduit un langage de programation simple, facile à utiliser, à compiler, à déboguer, et à apprendre. De plus, tous les développeurs Java sont potentiellement capables de concevoir des applications Java Card.

   La facilité de Java Card se représente en :
   – Le langage Java Card est orienté objet donc il possède tous les avantages de ce type de langage .
   – L' existance des outils de développement puissants aussi bien libre que commerciaux, qui facilite la conception et le chargement des applications au sein de la carte .
   – La présence d'une plateforme ouverte avec des API et un environnement d'exécution standardisé.
2. *L'indépendance matériel* : Le code d'un programme JavaCard est universel au sens où il ne dépend pas du matériel où il sera exécuté tant qu'il existe la machine virtuelle JavaCard (Écrire une fois, exécuter n'importe où)[22].
3. *La plate-forme multi applicative* : JavaCard possède un modèle de sécurité qui permet à plusieurs applications de coexister en sécurité sur la même carte. Chaque applet possède son espace de mémoire propre (sandbox). Une carte peut contenir une application bancaire, un accès pour télévision satellite, un carnet de santé,etc. [22].
4. *Le partage de données entre applications* : Les différentes applications présentes sur une carte peuvent accéder à des données d'une autre application si le programme a été envisagé via le firewall. Seul le JCRE peut accéder à toutes les données. Une applet peut contenir les identifiants de sécurité sociale qui peuvent être utiles pour une applet mis par la mutuelle du détenteur [22].
5. *La sécurité des données* : est divisé en 2 parties, une partie est offerte par le langage Java (un langage fortement typé, plusieurs niveaux de contrôle d'accès aux méthodes et aux variables, pas de construction de pointeurs, etc) et l'autre partie est assuré par l'environnement d'exécution Java Card JCRE qui présente plusieurs mécanismes de sécurité tel que le pare-feu qui isole les applications hébergées dans la carte pour empêcher tout comportement hostile de leur part [11].
6. *La souplesse* : Le fait que les Applets soient chargées dans une EEPROM, permet de les mettre à jour en effaçant l'ancienne version et en chargeant la nouvelle. Cette évolutivité rend une carte pleinement réutilisable, alors qu'auparavant les applications étaient directement inclues dans le masque et donc non modifiables. Les modifications et chargement d'Applet font appel à des mécanismes sécurisés par le "Card Manager" [1].

2.4 Architecture de la plateforme

Java Card a été conçue pour les équipements fortement contraints tels que les cartes à puce vu ses ressources limitées spécialement en espace mémoire pour sauvegarder et charger les application.

La 1^{ere} solution adoptée a été de supporter seulement un sous-ensemble du langage Java, adapté au développement d'applications pour cartes à puce. Mais actuellement, avec les grands progrès

technologiques qu'a connu l'industrie des cartes à puce, de nouvelles cartes haut de gamme (microprocesseur 32 bits avec architecture RISC, beaucoup plus de mémoire et des interfaces de communication multiples pouvant fonctionner en USB 2.0) commencent à voir le jour. De ce fait, de nouvelles applications plus sophistiquées, tel qu'un serveur web, peuvent désormais être hébergées dans de telles cartes.

Le nouveau défi pour Java Card est de trouver une solution qui prend en charge à la fois les cartes à faibles ressources (représentant la majorité des cartes actuellement disponibles) et les cartes de nouvelle génération. C'est l'avènement de la version 3.0 de Java Card qui a apporté cette solution en introduisant deux versions de la spécification. L'une pour les cartes les plus communes (Classic Edition) et l'autre pour les cartes de nouvelle génération (Connected Edition). Chacune de ces deux éditions est compatible avec les applications développées pour les éditions précédentes (i.e. antérieurs à l'édition 3.0) [11].

2.4.1 Java Card 3 Classic Edition

Elle est basée sur l'évolution de la version 2.2.2 de la plateforme Java Card.Elle cible les cartes à faibles ressources qui supportent uniquement les applets comme modèle d'application. L'édition Classic adopte la même architecture (figure 2.1) de la plateforme Java Card que les versions précédentes (architecture utilisée depuis la version 2.1). Les changements apportés par cette version portent essentiellement sur le support des derniers algorithmes cryptographiques (standards de sécurité) ainsi que les standards régissant les communications sans contact [4].

La figure suivante (Figure 2.1) montre cette architecture.

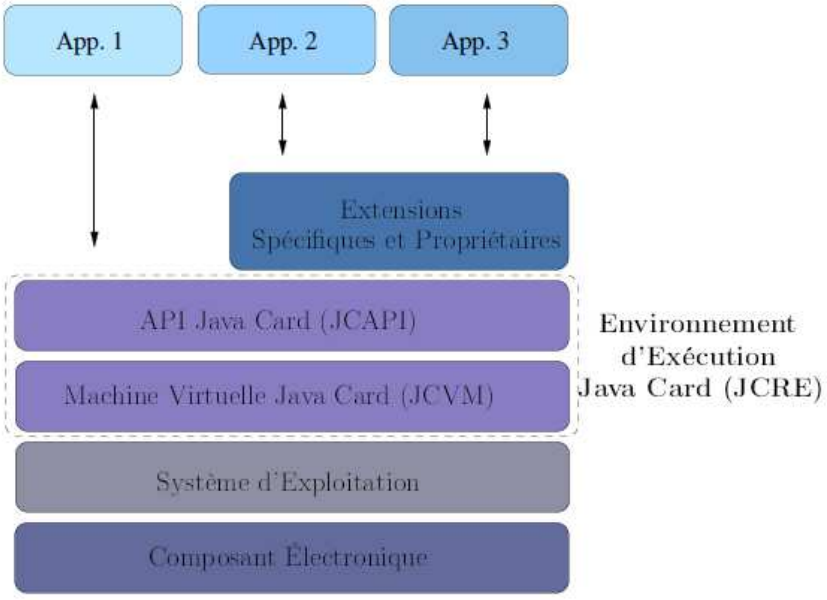

FIGURE 2.1 – Architecture de Java Card 3 Classic Edition
[3]

– Java Card 3.0.1 API Specification. Les APIs Java Card consistent en un ensemble de classes spécialisées dans la programmation d'applications de smart card conformément à la norme ISO 7816.Les classes dans ces APIs sont compactes et succinctes.Elles incluent des classes adaptées à la plate-forme java fournissant un support sur le langage et des services de cryptographie. Les

APIs contiennent trois packages principaux java.lang, javacard.framework, javacard.security et un package d'extension javacardx.crypto.

– Java Card 3.0.1 Virtual Machine Specification. La machine virtuelle Java Card (Java Card Virtual Machine ou JCVM) est basée sur les notions de piles pour la manipulation des valeurs et de tas pour le stockage des objets. Elle comporte un contexte d'exécution par méthode appelée.

  Tout comme la machine virtuelle Java, elle exécute du code sous forme de codes octet (byte-code). Comme les ressources sont limités dans la carte, alors que la différence entre la machine virtuelle javacard (JCVM) et la machine virtuelle java (JVM) est que la JVCM est subdivisée en deux parties séparées : l'une sur la carte, l'autre sur l'ordinateur.

  – Partie hors carte (Off-Card) : comprend un vérifieur de byte code et un convertisseur situé sur l'ordinateur qui prend en entrée des fichiers .class qui sont produits par le compilateur Java à partir du code source et un ou plusieurs fichiers d'exploitation.Le convertisseur pré-traite tous les fichiers .class et les convertit en un fichier .cap(Converted APplet).Le convertisseur ne produit pas seulement un fichier CAP mais aussi un fichier d'exportation.

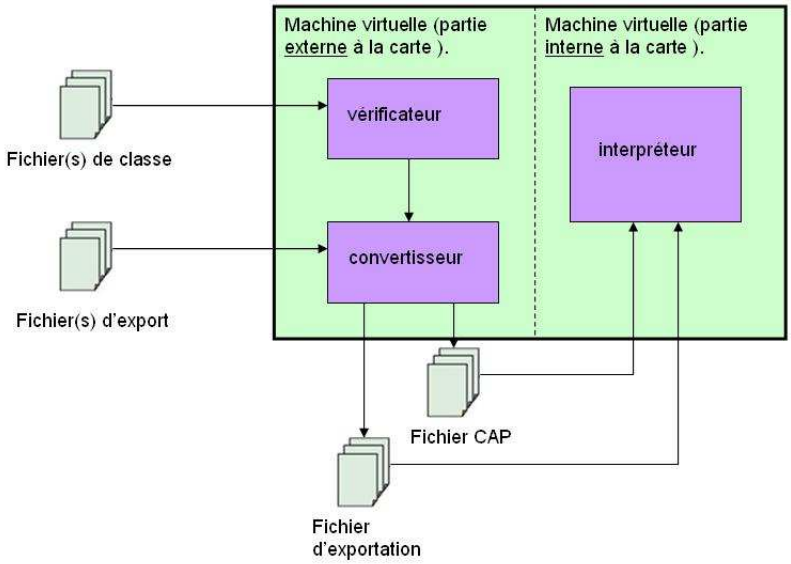

FIGURE 2.2 – La machine virtuelle Java Card JCVM

  – Partie sur carte (On-Card) : c'est la partie embarquée et elle comprend l'interpréteur de byte code qui se charge de l'exécution du code chargé. Le fichier .cap généré par le convertisseur, sera exécuté par l'interpréteur, il s'agit de faciliter l'installation et prévoir un format plus simple à exécuter pour une platet-forme disposant de peu de ressource [11][5].

  Pris ensemble ils implémentent toutes les fonctions de la machine virtuelle java nécessaire au sous-langage java des java cards.

– Java Card 3.0.1 Runtime Environment Specification. Décrit précisément le comportement de l'exécution de la Java Card. Java Card Runtime Environment (JCRE) gère les ressources de la carte, la communication avec le réseau, l'exécution des applets et leur sécurité.En fait c'est le système d'exploitation des smart cards.Le JCRE est situé au sommet de l'architecture d'une smart card.

  Il consiste en la machine virtuelle Java (l'interpréteur de bytecode), la structure des classes d'applications Java Card (APIs), des extensions industrielles spécifique, et les classes systèmes [4].

2.4.2 Java Card 3 Connected Edition

Elle est conçue pour les cartes de nouvelle génération disposant de plus de ressources, supportant ainsi un sous-ensemble plus large et plus riche du langage Java.

L'édition connectée de la plateforme introduit dans son architecture (Figure 2.3) une nouvelle machine virtuelle ainsi qu'un nouvel environnement d'exécution qui supporte maintenant trois modèles d'application en opposition à l'unique modèle proposé par l'édition classique des précédentes versions de la plateforme [22].

Ces trois modèles d'application sont :

- Les applets classiques basé sur celui hérité des précédentes versions de Java Card et à ce titre a les mêmes caractéristiques. Ces applications utilisent le modèle de communication avec la carte basé sur le protocole APDU.
- Les applets étendues est aussi basé sur celui des applets, mais avec des améliorations parmi lesquelles la gestion du multithreading, du format de fichier class, la gestion des chaines de caractères, etc. Elle utilise aussi le protocole de communication APDU.
- Les applications Web utilise le modèle d'application basé sur la construction de servlets . Il hérite des nouvelles API introduite avec cette édition de la plateforme. Et elle utilise le protocole HTTPS pour effectuer les communications avec la carte.

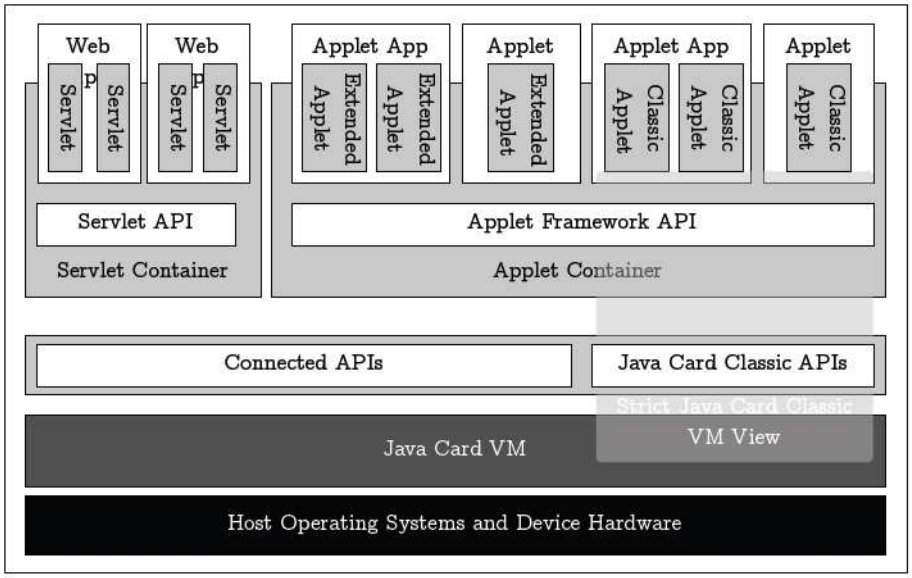

FIGURE 2.3 – Architecture du Java Card 3 Conncted Edition
[21]

2.4.3 Java Card et le sous-ensemble du langage Java

La quantité de ressources mémoires et processeurs disponible sur la carte étant faible, la plateforme Java Card ne contient qu'un sous-ensemble de Java. Cette section explique les différences qui existent avec Java [11].

L'édition Classic et Java

Les éléments non supportés du langage Java	Les éléments supportés du langage Java
Le chargement dynamique de classes : il n'existe aucun mécanisme permettant de télécharger des applications pendant l'exécution.	Paquetages.
Le gestionnaire de sécurité ou Security Manager.	Création dynamique d'objet.
La finalisation.	Méthodes virtuelles : l'invocation de méthodes virtuelles sur des objets comme dans un programme écrit pour la plate-forme Java (l'héritage et le mot clé super).
Les threads.	Interfaces.
Le clonage d'objet.	Exceptions.
Le contrôle d'accès : avec quelques restrictions.	les types simples : boolean, short et byte.
Les types énumérés : et le mot clef enum.	les classes Object et Throwable.
Les types char, double, float et long ainsi que les tableaux de plus d'une dimension.	Le type int et le mécanisme de suppression d'objets sont supportés en option.
De façon générale aucune des classes des API principales de java n'est supportée entièrement.	
Les fichiers Class.	

TABLEAU 2.1 – Le langage Java dans l'édition Classic

L'édition Connected et Java

Les éléments non supportés du langage java	Les éléments supportés du langage Java
Les types float et doubles.	Tous les types sauf les types float et doubles.
Les classloader : définit par l'utilisateur car ces classes ne peuvent pas être modifiés.	Le multi-threading.
La finalisation des instances de classes.	Le support des fichiers Class comme format de chargement avec l'édition des liens qui se fait sur la carte.
La gestion d'erreurs : sur la plateforme est très limitée. Il y a une classe d'erreurs qui est définie dans la spécification. En dehors de ces erreurs la machine virtuelle doit soit s'arrêter, soit renvoyer l'erreur la plus proche possible de la super classe de l'erreur représentant la condition d'erreur à lever.	La destruction automatique des objets non utilisés ou Automatic Garbage Collection.

TABLEAU 2.2 – Le langage Java dans l'édition Connected

2.5 Applets Java Card

Les applets sont des programmes écrit en langage java qui s'exécutent dans la carte en interagissant avec le JCRE. Elles sont identifiées par un identifiant unique AID (Application IDentifier).De même, chaque paquet est identifié par un AID unique. Ceci est la convention de nommage définie par la norme ISO 7816.Un AID est une séquence d'octets utilisée pour l'identification unique d'applets et de paquets. Cette séquence d'octets est composée de deux pièces distinctes. Voici comment elles sont organisées :

Application IDentifier AID	
RID (Ressource IDentifier)	PIX (Proprietary Identifier eXtension)
5 octets	0-11 octets

TABLEAU 2.3 – Structure de AID.
[11]

C'est l'ISO qui procède à l'attribution des RIDs aux entreprises afin que chacune ait son propre RID et ensuite chaque entreprise contrôle la gestion des PIXs. L'AID d'un paquet et l'AID par défaut d'une applet sont spécifiés dans le fichier CAP. Ils sont fournis au convertisseur lorsque le fichier CAP est généré [11][5].

2.6 Sécurité de Java Card

La plate-forme Java Card est un environnement de cartes multi-applicatives dans laquelle les données sensibles d'une applet doivent être protégées contre l'accès malveillant pouvant être réalisé par d'autres applets [1][21][11].

Il existe plusieurs types de mécanismes de sécurité fournis par Java Card 3, que l'on peut classifier en deux catégories principales :

- ceux qui sont intégrés au langage Java.
- ceux qui sont intégrés à la plateforme elle même.

2.6.1 Sécurité du langage Java

Le fait que Java Card soit un sous ensemble de Java, il intègre tous les mécanismes de sécurité existant dans ce langage de programmation à savoir [21][11] :

- Java est un langage fortement typé ce qui permet d'éviter les conversions de types non autorisées.
- Java n'utilise pas d'arithmétique sur les pointeurs, il n'a pas un moyen de forger des pointeurs.
- Le niveau d'accès de toutes les classes, méthodes et champs est strictement contrôlé (Public, Private, Protected).
- Interdiction de la conversion des types non prévue, il n'y a pas donc aucun moyen de falsifier un pointeur.
- L'opération de cast suit des règles strictes, cast implicite d'un sous-type vers un super-type, et cast explicite obligatoire d'un type vers un sous-type,

2.6.2 Sécurité de la plateforme

La sécurité de la plateforme est assurée par plusieurs mécanismes qui collaborent entre eux chaqu'un assure une partie et complète les autres pour offrir une sécurité maximale à la carte à puce.

Les principaux mécanismes sécuritaires intégrés aux cartes Java Card 3.0 Classic Edition et versions antérieures sont :

Le vérifieur de byte code La vérification du bytecode est un composant crucial de la sécurité dans le modèle d'isolation de Java qui garantit le type du code pour qu'il puisse être exécute sans risques par la machine virtuelle et ne peut pas outrepasser le mécanismes de sécurité de haut niveau.

La vérification consiste à effectuer une analyse statique du code mobile à charger. Cette analyse assure que le fichier contenant l'applet (fichier CAP) soit un fichier bien formé, qu'il n'y a pas de débordement de pile, que le flot d'exécution reste confiné sur du byte code valide, que chaque argument d'une instruction soit d'un type correct et que les appels de méthodes soient effectués conformément à leurs attributs de visibilité (public, protected, private).

Comme les ressources des cartes à puce sont limitées, le processus de vérification de byte code, coûteux en termes de temps d'exécution et d'espace mémoire, se fait dans la partie hors carte pour toutes les versions de Java Card antérieures à la version 3.0 Classic Edition y compris celle-ci [1][21][11].

Pare-feu Le pare-feu permet d'apporter une protection contre la fuite d'informations vers d'autres applications dues aux erreurs de programmation et de conception.Il impose l'isolement entre les applets à l'intérieur de la carte, il n'autorisera que les interactions entre les applets appartenant au même contexte.La notion du contexte est basée sur la structure des packages . Si deux applets sont des instances de classes provenant du même package JavaCard, ils sont considérés comme appartenant au même contexte.

Ainsi, chaque objet est assigné à un contexte unique propriétaire qui est le contexte de l'applet qui a créé l'objet. Une applet a le droit d'accéder à ses objets (et à chaque objet situé dans le même paquetage), mais le pare-feu vérifie toujours qu'une applet n'essaye pas d'accéder illégalement à un objet n'appartenant pas à son contexte. Ainsi, le pare-feu isole les contextes de telle manière qu'une méthode étant exécutée dans un contexte ne puisse accéder à aucun champ ou méthodes d'objets appartenant à un autre contexte [1].

Mécanisme de partage les différentes classes d'un même contexte peuvent échanger librement des informations, ceci n'étant pas possible pour deux applications appartenant à deux contextes différents. Pour cela Java Card introduit la notion d'objets partagés qui sont définis comme des classes héritant de l'interface Shareable. Lorsqu'une classe est définie comme partagée, elle autorise les applets de contextes différents à invoquer les méthodes définies dans son interface partageable sans qu'elles ne soient bloquées par le pare-feu, comme le montre la figure 2.4.

Un tel mécanisme est réalisé sous le contrôle du JCRE qui veille à ce que les règles de partage soient respectées. Le mécanisme de partage d'objets permet donc d'établir une communication inter-applications en dépit du fait qu'elles ne partagent pas un même contexte [13].

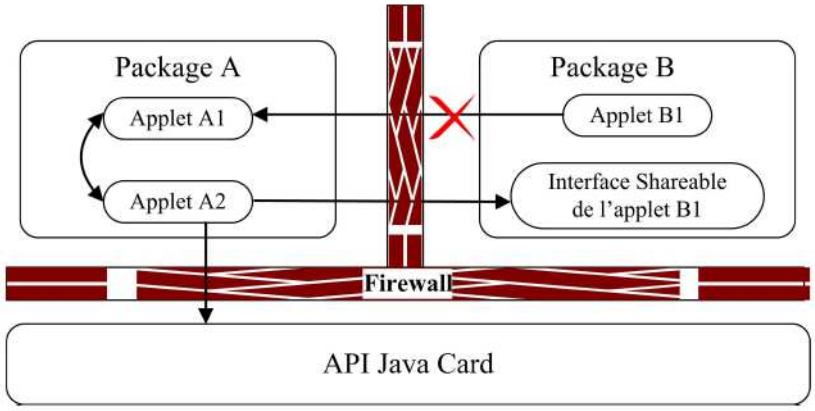

FIGURE 2.4 – Le principe du pare-feu Java Card.
[14]

Atomicité et mécanisme de transaction Afin de garantir l'intégrité des données lors de la mise à jour des objets persistants, les notions d'atomicité et de transaction ont été introduites dans l'environnement Java Card.

Une transaction est un mécanisme permettant de rendre un ensemble d'opérations atomique (i.e totalement exécutée ou pas du tout). Par exemple, il est important que lors d'une transaction bancaire les données sur la transaction soient mises à jour entièrement, ou bien pas du tout. Le JCRE offre alors un support aux applettes pour restaurer les données à leur état avant la transaction lorsque celle-ci n'a pas terminé normalement (la carte a été arrachée du lecteur par exemple). Le déclenchement, la terminaison ou l'annulation des transactions se fait via des méthodes des APIs (respectivement beginTransaction(), commitTransaction) et abortTransaction() de la classe JCSystem). Le mécanisme de rollback des données est utilisé pour restaurer les anciennes données en cas d'échec ou en cas d'annulation de la transaction [11][9].

2.7 Conclusion

Dans ce chapitre, nous avons vu quels étaient les avantages de la plateforme Java Card : à savoir une plateforme offrant de la portabilité aux applications pour cartes à puce, une simplicité de développement du fait de l'existence de nombreux environnements de développement et de la simplicité de l'apprentissage du langage Java, et de nombreuses fonctionnalités de sécurité. De plus, c'est la plateforme pour cartes à puce la plus déployée dans le monde.

Dans le chapitre suivant nous allons présenter les différentes manipulations frauduleuses qui peuvent mettre en question la sécurité des cartes à puce en général, et qui visent les informations secrètes qu'elles contiennent.

Chapitre 3

Attaques contre la carte à puce

Sommaire

Les composants de sécurité, notamment ceux des cartes à puce, et parce qu'ils contiennent des informations confidentielles, font l'objet d'attaques. Celles-ci tentent généralement de porter atteinte à la confidentialité, à l'intégrité ou à l'authenticité des données protégées par les algorithmes de cryptographie embarqués sur ces composants.
La sécurité d'une carte à puce peut être contournée de plusieurs manières : soit en prenant le matériel en défaut, soit en prenant l'applicatif ou le système en défaut, soit en combinant les deux.

3.1 Attaques physiques

Les attaques dites matérielles ou physiques exploitent les faiblesses de l'implantation matérielle des algorithmes embarqués sur cartes à puce. Contrairement aux attaques logiques, l'attaquant doit avoir le composant "entre les mains" pour pouvoir mettre en œuvre ces attaques. Dans cette section nous présentons les différents types d'attaques auxquels un circuit sécurise (celui d'une carte à puce) peut être soumis. Dans la littérature, les attaques sont souvent classées suivant deux axes : le caractère intrusif de l'attaque et l'activité de l'attaquant. En effet, les attaques peuvent se différencier d'une part par leur caractère intrusif par rapport au circuit. On retrouve dans cette catégorie trois types d'attaques : les attaques invasives, les attaques non invasives et les attaques semi invasives. D'autre part elles sont différenciées par rapport à l'activité de l'attaquant lors de la réalisation d'une attaque, et on retrouve deux types d'attaques : les attaques actives et les attaques passives

(voir Tableau 3.1).

Attaques	Actives	Passives
Invasives	Modification physique du circuit.	micro-sondage.
Semi invasives	Attaques par injection de fautes (Perturbation lumineuse).	
Non invasives	Attaques par injection de fautes (Perturbation de l'alimentation, de signal d'horloge ou de température)	Attaques par canaux cachés (attaques temporelles, attaques en puissance SPA/DPA, attaques électromagnétiques EMA).

TABLEAU 3.1 – Catégorisation des attaques physiques contre les cartes à puce. [6]

3.1.1 Selon l'activité de l'attaquant

3.1.1.1 Attaques actives

Les attaques actives requièrent diverses actions de la part de l'attaquant soit sur le circuit lui-même soit sur son environnement. Le but est de perturber le fonctionnement normal du composant afin de se retrouver dans un mode imprévu ou bien afin d'obtenir des résultats erronés.

3.1.1.2 Attaques passives

Lors d.une attaque passive, le pirate ne fait que mesurer certaines grandeurs physiques émanant du circuit pendant le fonctionnement normal du système.

3.1.2 Selon le caractère intrusif

3.1.2.1 Attaques invasives

Les attaques invasives nécessitent d'accéder physiquement aux éléments constitutifs des circuits intégrés (tels que les pistes, les plots d'entrées/sorties, les bus d'interconnexion, etc.) en utilisant des équipements de laboratoire très spécifiques. L'accès aux éléments constitutifs nécessite une phase de préparation du circuit,une phase de compréhension de l'organisation des éléments constitutifs puis une phase de récupération des informations. Celle-ci est réalisée soit par **sondage**, soit par **modification du circuit**.

1. **Préparation**
   La première étape de préparation du circuit est la décapsulation du module qui l'intègre et l'enlèvement des couches de passivation qui protègent le circuit contre l'environnement et la migration ionique. Cette étape met en œuvre des procédés physico-chimiques relativement simples telles que l'abrasion mécanique ou la dissolution par un acide. La photo de gauche de la figure 3.1 représente une puce de carte à puce qui a été préparée en vue d'une attaque en injections par faute.

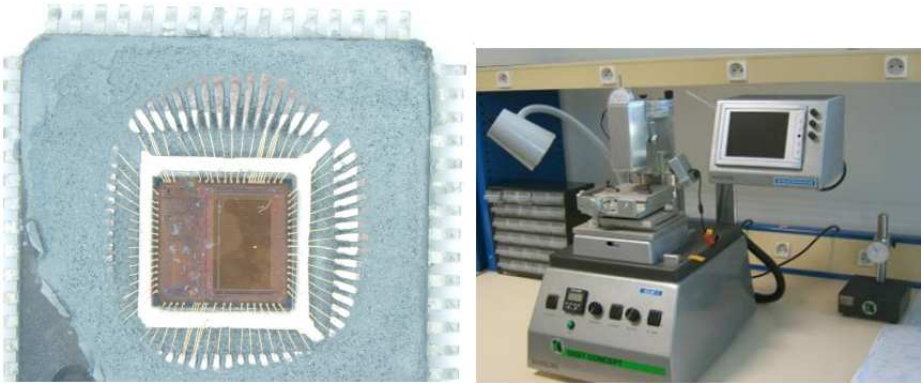

FIGURE 3.1 – Puce décapsulée et machine de décapsulation
[18]

2. **Rétro-conception matérielle**
   Cette phase vise à connaître la structure des éléments constitutifs du circuit pour en déduire sa fonction. Tous les niveaux de matériaux déposés durant la fabrication sont enlevés en ordre inverse et sont respectivement photographiés (voir figure 3.2) à l'aide d'un microscope associé avec une caméra CCD[1]. En traitant toutes les informations ainsi acquises, ce qui peut se révéler extrêmement long et coûteux pour un circuit récent, l'attaquant peut en déduire le schéma du circuit puis les algorithmes implantés. Ces connaissances l'aideront à choisir les positions de ses micro-sondes.

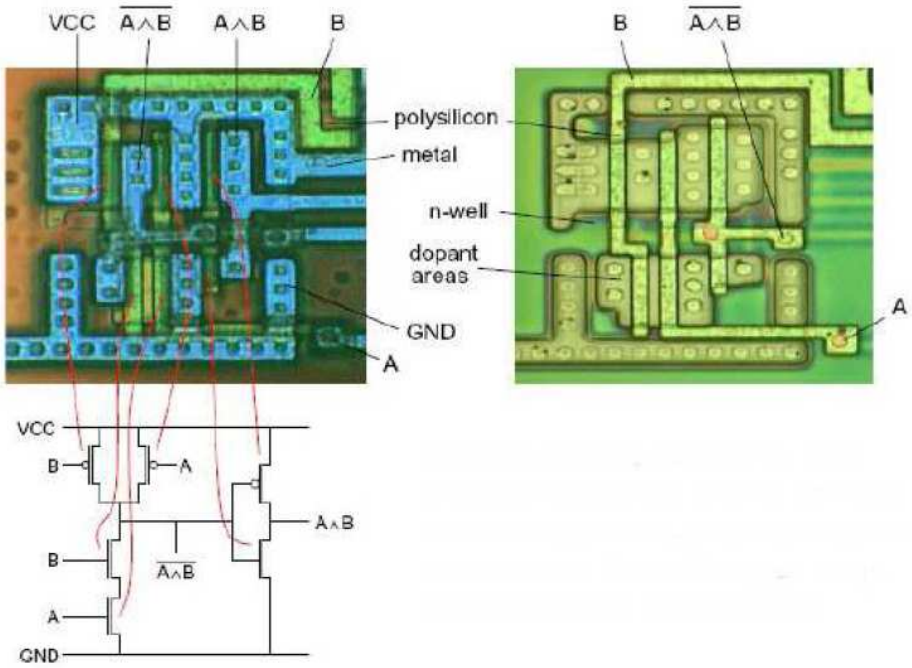

FIGURE 3.2 – Reconstruction d'un schéma original à partir des images acquises
[18]

1. *Charge-Coupled Device*, Capteur photographique (le composant de base des caméras numériques).

3. **Attaque**
   - **Micro-sondage**
     Le micro-sondage (ou *"micro-probing"*) consiste à poser des sondes très fines sur certaines pistes du circuit et d'accéder ainsi à la valeur des signaux internes du composant. L'attaquant peut ainsi examiner, grâce à une **"sonde passive"**, les échanges qui ont lieu sur le bus de communication entre le processeur et la mémoire ou entre le processeur et un bloc de cryptographie. Il peut même forcer un fil de ce bus à une valeur particulière (on parle alors de **"sondes actives"**). La pose des micro-sondes nécessite généralement la réalisation de plots de contact et parfois celles de pistes verticales (appelées "via"). Ces reprises de connexion sont généralement faites à l'aide d'un FIB (pour *Focus Ion Beam*). La figure 3.3 représente des points de test posés sur un bus de communication.

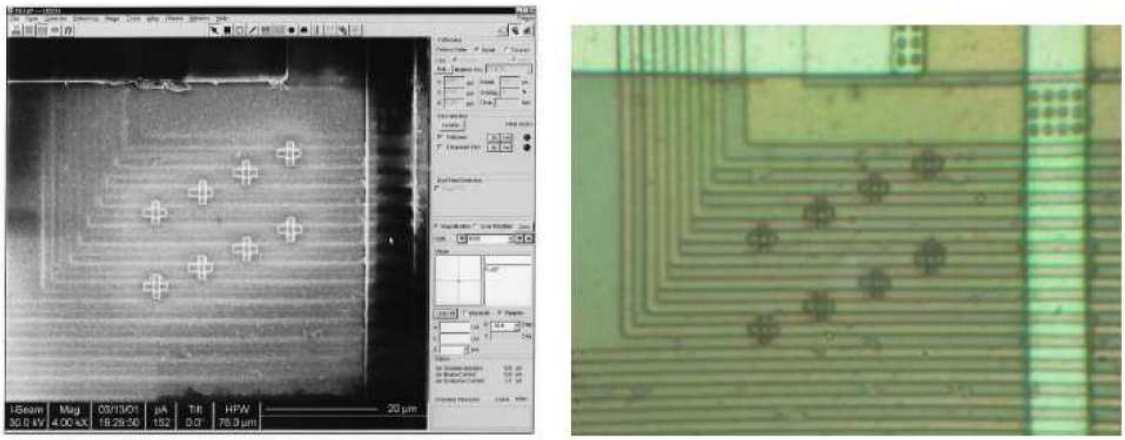

FIGURE 3.3 – Création des points de tests par FIB
[18]

   - **Modification du circuit**
     Certains circuits sont munis de protections (tels que des capteurs de tension d'alimentation, de lumière, de fréquence d'horloge, etc.) contre les attaques en faute [section 3.1.2.3]. L'attaquant peut être amené à les désactiver en coupant les fils de métal d'interconnexion, par exemple, avec un scalpel laser ou un FIB tel que représenté sur la figure 3.4.

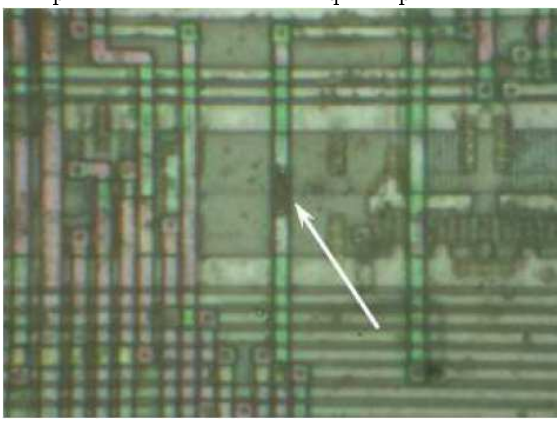

FIGURE 3.4 – Coupure d'un simple fil par scalpel laser
[18]

3.1.2.2 Attaques semi-invasives

Selon l'auteur de [6] la notion d'attaques semi intrusives a été introduite dans [20]. Dans ce type d'attaques la puce n'est pas détruite mais le package contenant la puce est enlevé afin de pouvoir réaliser l'attaque et observer de plus près le comportement de la puce. Les attaques semi-invasives peuvent être effectuées en utilisant des outils tels que la lumière UV, les rayons X et d'autres sources de rayonnements ionisants, les lasers et les champs électromagnétiques. Ils peuvent être utilisés individuellement ou en combinaison les uns avec les autres. Les auteurs de [20] par exemple utilisent un flash de caméra pour cibler un transistor et faire changer l'état d'une cellule mémoire SRAM dans un microcontrôleur.

3.1.2.3 Attaques non invasives

Les attaques non invasives sont considérées comme une menace réelle en raison de leur caractère même. En plus, elles nécessitent souvent beaucoup moins d'équipement que les attaques invasives. L'attaquant observe ici les paramètres externes ou les phénomènes physiques liés au fonctionnement de la puce et liés aux données traitées. L'attaquant peut ainsi en déduire des informations sur la clef de chiffrement.
Deux types d'attaques peuvent être distinguées à ce niveau : les attaques basées sur l'observation de l'environnement de la carte (**attaques par canaux cachés**) et celles basées sur la perturbation de cette dernière (**attaques par injection de fautes**).

1. **Attaques par canaux cachés**
   Les attaques par canaux cachés sont basées sur l'observation de grandeurs physiques dépendantes du calcul exécuté au sein du circuit. Ces grandeurs sont appelées canaux cachés car bien qu'elles soient mesurables sur le circuit, elles ne constituent pas son résultat et ne devraient pas être porteuses d'informations sur l'état interne du circuit. Leur observation peut néanmoins donner de nombreuses informations sur les clefs de chiffrement utilisées. Les grandeurs les plus utilisées sont le temps d'exécution, la consommation électrique ainsi que le rayonnement électromagnétique. A partir de cette liste de canaux cachés, on obtient naturellement la liste correspondante d'attaques : les **attaques temporelles** (*Timing Attack*), les **attaques en puissance simples ou différentielles** (***SPA/DPA*** *: Simple/Differential Power Attack*) et les **attaques électromagnétiques** (**EMA** : *ElectroMagnetic Attack*) [16].

   – Les **attaques temporelles** sont basées sur l'analyse de données obtenues lors de la mesure précise du temps de calcul du circuit attaqué (ou le nombre de cycles nécessaires à cette opération dans le cas d'un circuit synchrone) pour un ensemble de valeurs. La mesure du temps d'exécution est faite par le dispositif pilotant le circuit attaqué, qui lui fournit entre autres son signal d'horloge, permettant ainsi de mesurer la durée entre le début du calcul et l'obtention du résultat. Une analyse de ces données aboutit à des informations sur la clef de chiffrement. Cette attaque n'est valide que sur des implantations en nombre de cycles non constant, étant donné que si aucune variation n'est observable, aucune information sur la clef n'est déductible. Les différences de temps d'exécution sont souvent dues à certaines instructions ou opérations qui ne sont réalisées que dans certaines branches de l'algorithme. Ces branches dépendent souvent des données traitées et sont donc la source des fuites sur ce canal caché. Selon [16] Cette attaque a été pour la première fois présentée par l'auteur de [15] et réalisée par les auteurs de [8].

   – Les **attaques par analyse de la consommation** sont basées sur la variation de consommation électrique du circuit en fonction des données traitées. Ceci est lié aux différences de consommation d'une porte logique suivant ses transitions. Ainsi, il est possible à partir du profil de consommation du circuit pour un ensemble d'entrées différentes d'obtenir les

valeurs de la clef. On peut distinguer plusieurs types d'attaques en puissance : l'attaque en puissance simple, l'attaque en puissance différentielle et l'attaque en puissance différentielle d'ordre supérieur [16].
  - Le but de l'attaque par analyse simple de consommation **(SPA)** est d'essayer d'obtenir directement de l'information sur la clef secrète à partir d'une seule mesure de consommation[11].
  - L'attaque par analyse différentielle de consommation **(DPA)** permet d'obtenir de l'information en utilisant des méthodes statistiques permettant de distinguer la faible corrélation qui existe entre la valeur des données manipulées et les mesures de consommation électrique relevées [11].

- Les **attaques électromagnétiques (EMA)** (trés similaires à celles par analyse de consommation) sont basées sur l'étude du rayonnement électromagnétique au voisinage proche du circuit. Ce rayonnement électromagnétique est une image de l'activité interne du circuit donc corrélé de la même manière aux données traitées. En effet, le courant utilisé par la puce crée un champ électromagnétique qui dépend lui aussi de l'activité de la puce. Ce champ peut être mesuré grâce à une sonde spécifique reliée à un oscilloscope et l'analyse de ce signal s'effectue ensuite de manière similaire à l'analyse de la consommation électrique. Cependant, une telle analyse permet d'isoler l'activité d'une toute petite partie de la puce. En effet, suivant la position de la sonde à la surface du composant, l'attaquant va pouvoir analyser le rayonnement électromagnétique de telle ou telle partie du composant. Par conséquent, le bruit induit par les autres parties de la puce lors d'une analyse de consommation électrique peut être grandement diminué voire même supprimé.
  Tout comme les attaques en puissance, les attaques électromagnétiques peuvent se décliner en attaques simples et différentielles [16]. Contrairement aux attaques en puissance, ces attaques peuvent avoir une résolution de l'ordre de quelques portes en utilisant des sondes adaptées. L'auteur de [16] a pu trouvé un exemple de mise en place d'une telle attaque dans [10].

2. **Attaques par injection de fautes**
   Cette attaque consiste à changer le comportement d'un composant en le perturbant afin de créer une erreur exploitable [12]. De tels défauts peuvent être induits par différents moyens, y compris des fautes transitoires de courte durée (altération rapide de l'alimentation), l'ajout d'énergie par rayonnement laser, etc. L'attaque vise à rendre des opérations cryptographiques moins sûres, obtenir plus facilement des clefs, ou à modifier les flots de contrôle du programme en altérant la mémoire, les informations transitant sur les bus ou les registres du processeur. Cette classe d'attaques sera abordée en détail dans le chapitre suivant (4).

3.2 Attaques logiques

Les attaques logiques présentent une menace majeure pour les systèmes embarqués, surtout pour les systèmes qui ont la capacité de télécharger et exécuter le code des applications. Les attaquants logiciels malveillants exploitent les faiblesses et les lacunes présentés dans l'implémentation et l'architecture des systèmes (vulnérabilités)[1].

Elles consistent à s'attaquer à la partie logicielle de la plateforme par injection de données ou chargement d'applications malicieuses dans le but de contourner l'exécution des applications installées dans la carte à puce, (par exemple des applications pour lesquelles les instructions ne respectent pas les règles de typage de Java ou les règles de construction des fichiers d'entrée, ou encore utilisent des imprécisions d'une ou plusieurs des spécifications Java Card [12]), ou de dévoiler ses secrets (clés, code Pin, etc.).

Il s'agit souvent d'exploiter des erreurs de conception ou de codage qui permettent à l'attaquant de contourner certains mécanismes de protection, ou de détourner l'utilisation de certaines fonctionnalités de la carte à puce[5].

Des attaques bien connues comme le cheval de Troie ou le débordement de pile ont montré toute leur efficacité aussi bien sur les systèmes d'exploitation des ordinateurs personnels que sur des systèmes d'exploitation moins complexes comme ceux des systèmes embarqués ou des cartes à puce [17].

E. Poll et W. Mostowski proposent plusieurs attaques par confusion de type. L'objectif est de transformer un objet d'un type donné pour lequel un nombre limité d'opérations est possible en un objet pour lequel d'autres opérations deviennent éligibles. Par exemple :transformer un tableau d'octets en un tableau d'entiers :
permet de pouvoir lire deux fois plus de données et potentiellement des données sortant du domaine de l'applet.

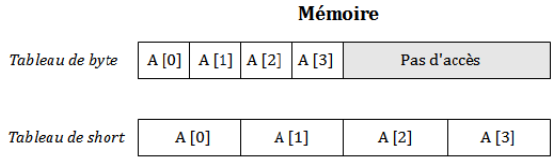

FIGURE 3.5 – Confusion de type entre les tableaux.

Les auteurs J.Iguchi-Cartigny et J.L.Lanet ont présenté une approche étendant celle de Hyppönen dans le but de lire et écrire n'importe où dans la mémoire de la carte (pas seulement à une adresse mémoire). Pour cela, ils ont développé un code auto-modifiable (le cheval de Troie) qui vise la recherche et le remplacement de motifs afin de remplacer des parties de code, dans la carte, qui n'appartiennent pas au contexte de sécurité de l'applet développée en contournant les mécanismes de sécurité. Cette attaque a été menée sur plusieurs cartes Java Card. Ces dernières ont présenté des réactions différentes : résistance complète à l'aide des contremesures implémentées, exécution partielle du code en contournant certaines contremesures, aucune résistance face à l'attaque.

J.Hogenboom et W.Mostowski ont présenté une nouvelle attaque qui utilise une confusion de type en exploitant un bug dans l'implémentation du mécanisme de transaction. Mais contrairement à J. Iguchi-Cartigny et J.L. Lanet, cette attaque ne nécessite aucune manipulation du fichier à charger dans la carte (i.e. le fichier CAP) d'où sa simplicité de réalisation. Cette attaque a conduit à une lecture/écriture de la totalité du contenu de la mémoire, offrant ainsi la possibilité de manipuler des données sensibles d'autres applets ou même propres à la carte [13][12][11].

3.3 Attaques combinées

Sur la catégorie des attaques précédentes, la spécification Java Card 3.0 change considérablement la donne. Grâce aux nouveaux mécanismes de sécurité disponibles sur l'édition connectée, en particulier la vérification de bytecode qui est pratiquée systématiquement lors du chargement des programmes sur la plateforme et le ramasse-miette qui est automatique. Les attaques logiques visant spécifiquement Java Card deviennent plus difficiles à mettre en œuvre. Cependant pour réussir à contourner le vérificateur de bytecode, un attaquant peut faire une combinaison d'attaques matérielle et logicielle, c'est l'attaque combinée. Cette combinaison consiste à charger dans la carte des applications sémantiquement correctes qui sont par la suite modifiées par une attaque physique pour générer un code malicieux permettant de réaliser des attaques logiques. L'application modifiée est appelée mutant.

G.Barbu et al. ont introduit une nouvelle approche combinant une injection de fautes avec une attaque logique. Deux cas d'étude, réalisés sur des cartes Java Card 3.0 Connected Edition, ont été présentés. Le premier cas montre comment introduire un code mal-formé bien qu'un vérifieur de byte code embarqué soit présent dans la carte. Alors que le second exemple expose comment transformer une méthode quelconque présente dans la carte en un code malicieux.

G.Bouffard et al. ont proposés une nouvelle approche combinant aussi les attaques logiques et physiques. Une modification du flux d'exécution à travers un rayon laser, permet de contourner le vérifieur de byte code embarqué et ainsi avoir le contrôle de la mémoire (dump de l'EEPROM) [11][13].

3.4 Conclusion

Les cartes à puces présentent comme tout système informatiques un nombre de failles de sécurité qui les rendent vulnérables aux différents types d'attaques ; physiques, logiques ou combinées.

Trois familles de vulnérabilités matérielles (physiques) sont distinguées. La première est la famille de vulnérabilités aux attaques non-invasives, ces attaques n'entrainent pas la destruction du matériel (i.e. la carte à puce). La seconde famille est celles des vulnérabilités aux attaques invasives. Dans ce cas, l'attaque opérée permet effectivement de voler l'information, mais le matériel est détruit. Enfin, la troisième famille de vulnérabilité est celle des attaques semi-invasives.

Dans le chapitre suivant nous allons détailler les différents aspects de l'une des attaques matérielles non-invasives ; l'attaque par injection de fautes.

Chapitre 4

Attaques par injection de fautes

Sommaire

Ce chapitre est consacré à la présentation des attaques en faute et aux contre mesures proposées jusque là. Il présente tout d'abord les différents moyens utilisés pour perturber le circuit d'une carte à puce et provoquer une faute, puis les différents modèles et types de fautes qui peuvent être engendrés par ces perturbations, ensuite leurs conséquences. Ce chapitre se termine par une présentation des différentes contre mesures proposées dans la littérature pour protéger les systèmes contre les attaques par injection de fautes.

4.1 Méthodes d'injection de fautes

4.1.1 Variation de tension d'alimentation

Un circuit intégré est composé d'éléments de mémorisations (bascules ou Flip-Flops) qui comparent, pendant un court intervalle de temps, leurs entrées à la tension d'alimentation. Cet échantillonnage, nécessaire au bon fonctionnement du circuit, et qui tolère une certaine variation d'alimentation électrique, typiquement de +/-10% par rapport à la tension nominale [6], peut donc être altéré par une variation de la tension d'alimentation [18]. En conséquence, et lors d'exécution cela peut par exemple mener un processeur à mal interpréter ou omettre des instructions [2].

Diverses variations se sont avérées efficaces : de l'impulsion brève mais d'amplitude élevée à des variations de faible amplitude mais quasi-statique [18].

Dans le circuit de la figure 4.1, un glitch[1] négatif sur l'alimentation Vdd[2] ralentit la partie combinatoire du circuit, à tel point que le front validant de l'Horloge survient avant que la nouvelle valeur ne soit prête en sortie de la partie combinatoire D2. La valeur mémorisée dans le registre de sortie est donc erronée. La figure 4.2 illustre le comportement des différents signaux en fonctionnement normal et en fonctionnement avec application de glitch de tension. On remarque que le glitch sur Vdd entraine un retard sur D2 et une erreur sur Sortie.

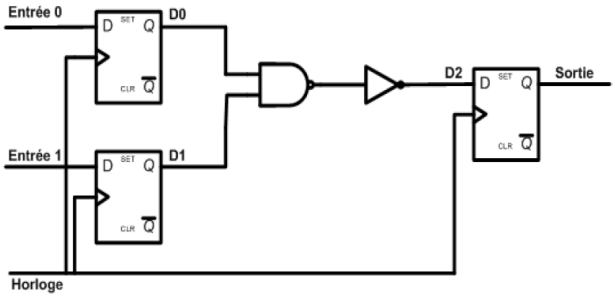

FIGURE 4.1 – Circuit séquentiel synchrone
[6]

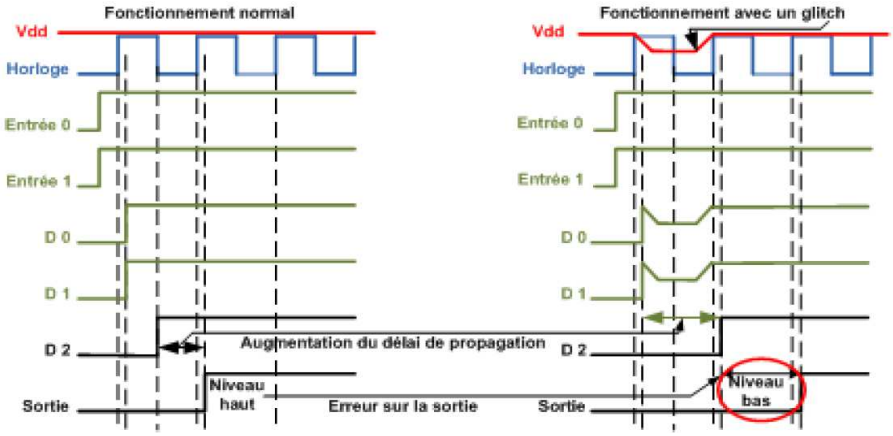

FIGURE 4.2 – Simulation d'un glitch négatif sur Vdd
[6]

1. Un glitch est une défaillance électronique ou électrique.
2. Désignation de la broche d'alimentation positive d'un circuit intégré.

Dans cet exemple particulier, le glitch produit une erreur sur le seul bit de sortie. Bien entendu, le glitch peut produire plusieurs erreurs sur les différentes sorties liées à toute la partie combinatoire alimentée par la tension Vdd sujette au glitch.

4.1.2 Modification de la fréquence d'horloge

La majorité des circuits actuels fonctionnent au pas cadencé d'une horloge (on parle dans ce cas de circuits "synchrones"). Ces circuits possèdent une fréquence de fonctionnement maximale, liée à la durée de transfert des données dans leurs blocs combinatoires. Si le circuit est cadencé par une horloge externe, l'attaquant peut imposer une fréquence d'horloge supérieure à la fréquence maximale du circuit. La période d'horloge est alors inférieure à la durée du transfert des données dans les blocs combinatoires (on parle d' "overclocking") et le registre en aval stocke soit une valeur instable soit la valeur précédente. Dans ces deux cas, des opérations réalisées par le circuit peuvent s'avérer incorrectes. La figure 4.3 représente une réduction sur un seul cycle de la période de l'horloge [18].

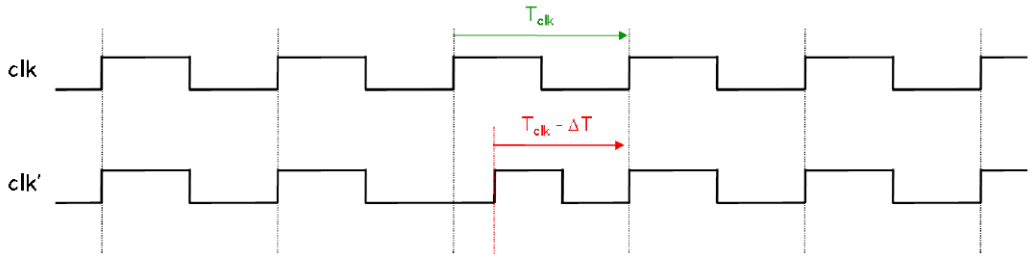

FIGURE 4.3 – Réduction d'un cycle d'horloge
[18]

En augmentant globalement la fréquence de fonctionnement du circuit, le nombre d'erreurs augmente [6].

4.1.3 Perturbation de la température

La modification de la température opérationnelle d'un circuit, au-delà des bornes spécifiées par le constructeur, modifie son comportement. Les blocs de mémoire sont les éléments les plus sensibles à ce type de modifications. En effet, ces changements peuvent altérer les temps de lecture et d'écriture dans la mémoire et donc être la source d'incohérences dans les données lors de l'exécution [16].

Contrairement aux attaques précédentes qui nécessitent une parfaite localisation dans le temps (choix d'un cycle "fauté"), les attaques par perturbation de la température demandent une localisation géométrique. En effet, sans une maitrise expérimentale parfaite, tout le système attaqué devient défaillant. Il est donc extrêmement difficile de ne produire que des erreurs exploitables [6].

4.1.4 Champ électromagnétique

Il s'agit de générer un champ électromagnétique intense à proximité du composant. Cela modifie la tension de seuil [3] du transistor [4] de telle sorte qu'il ne peut plus commuter pendant la perturbation électromagnétique. Ainsi, suivant le type du transistor, ceci permet de s'assurer qu'une cellule

3. La tension de grille (voir note suivante) qui fait la transition entre le comportement bloqué du transistor et son comportement conducteur.

4. Un dispositif semi-conducteur à trois électrodes permettant de contrôler un courant sur une des électrodes de sorties grâce à une électrode d'entrée (base/grille).

mémoire contient la valeur 0 ou 1 [6]. La figure 4.4 représente une bobine et un générateur d'impulsions qui peuvent être utilisés pour générer un tel champ. Un tel dispositif permet de modifier les valeurs stockées dans des cellules mémoire [18].

FIGURE 4.4 – Dispositif d'injection électromagnétique
[18]

Toutefois, le principal problème d'une telle approche est de cibler des bits précis. Cela exige de l'attaquant de connaitre la disposition du dispositif attaqué afin de pouvoir contrôler la zone ciblée [6].

4.1.5 Illumination (Attaque optique)

En utilisant une source lumineuse concentrée sur la puce, il est possible d'apporter suffisamment d'énergie à une cellule mémoire pour lui faire changer son contenu. C'est en se basant sur cette propriété physique des mémoires que cette attaque est née [21].

Ce type d'attaques semi-invasif nécessite l'extraction des couches protectrices et une diminution de l'épaisseur du silicium pour avoir une bonne pénétration des éléments perturbateurs tels que la lumière, et comprend trois techniques d'injection [6] :

Injection de lumière D'après [6], en 2002, Sergei Skorobogatov et Ross Anderson présentèrent un équipement assez simple et peu coûteux permettant d'injecter en pratique des fautes en utilisant un simple flash d'appareil photo. Cela leur a permis de modifier la valeur d'un bit d'une cellule mémoire [20].

Le flash lumière ne permet généralement pas une localisation précise de l'injection de faute et provoque donc un disfonctionnement en de nombreux points du circuit attaqué [6].

Injection de particules Cette technique d'injection de fautes peut être réalisée en utilisant des accélérateurs de particules. Cependant, cet équipement coûte cher et reste réservé à certains laboratoires [6].

En raison des contraintes de fabrication des canons à particules utilisés pour l'injection, il est difficile d'assurer un bon contrôle spatial et temporel avec cette technique [6].

Injection laser L'un des principaux avantages du laser est qu'il permet une bonne maîtrise temporelle et spatiale de l'injection de fautes favorisant ainsi la reproductibilité de l'attaque. Un autre avantage est que le coût des bancs laser est nettement inférieur à celui des accélérateurs de particules par exemple [6].

La figure 4.5 reporte deux photos d'un banc d'injection de fautes par laser.

FIGURE 4.5 – Banc laser pour l'injection de fautes
[18]

4.2 Modèles de fautes

Pour se prémunir contre une attaque par injection de fautes, il faut avant tout identifier quels sont les éléments qui permettent de pouvoir caractériser une faute, et ensuite définir une modélisation qui permet de déduire l'impact de la faute sur les applications à protéger.

Les différents modèles distinguent généralement la localisation de la faute dans le temps et l'espace, la précision et le type de fautes engendrées [21] :

- La **fenêtre temporelle** pendant laquelle l'attaque s'effectue. En effet, une attaque réussie doit pouvoir être synchronisée avec une instruction ou alors avec la manipulation d'une donnée importante.
- La **localisation spatiale** de la faute à la surface de la puce. Si l'on veut effectuer une attaque efficace, il faut savoir à quel endroit de la carte on l'effectue. Il faut donc arriver à distinguer les différents types de mémoire.
- La **précision** est la quantité d'information modifiée par l'attaque. Car en fonction des propriétés physiques de l'attaque, et des protections intégrées à la puce, il est plus ou moins difficile d'affecter des zones mémoires très petites. Plus la carte contient des mécanismes de protection, plus il est difficile de modifier une zone suffisamment petite et précise de la mémoire.
- Le **type de faute** est l'état de la zone mémoire modifiée après l'attaque. Ici, il faut savoir si l'attaque a la capacité de changer l'information vers une valeur donnée dans le cas des mémoires non chiffrées ou si elle est soumise à un aléa dans le cas des mémoires chiffrées.
  Il existe plusieurs types de fautes, définies ci-dessous et représentées sur la figure 4.6 :
  - La valeur du bit est inversée par rapport à la valeur qu'il aurait eu sans perturbation. On parle alors de **bit flip**.
  - La valeur du bit est forcée à une valeur fixe. On parle de **set** si la valeur est fixée à 1 et de **reset** si la valeur est fixée à 0.
  - La valeur du bit est forcée à sa valeur précédente. On parle de **collage** à 1 si la valeur précédente était égale à 1 et collage à 0 sinon.
  - La valeur du bit est forcée à 0 ou 1 avec une probabilité de 50% **random**.

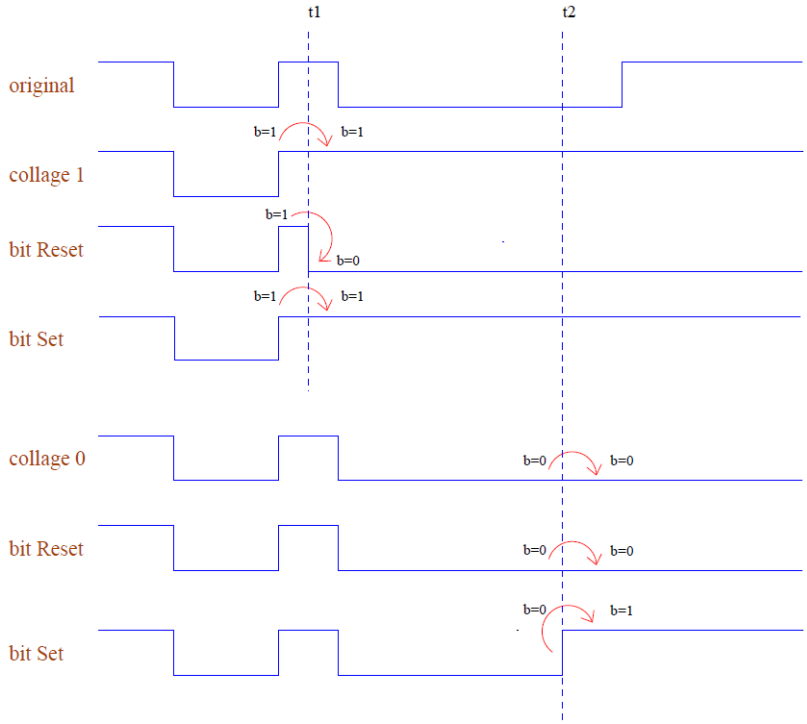

FIGURE 4.6 – Classification des types de fautes
[18]

Grâce à ces différents critères, on peut en déduire les modèles de fautes cités dans le tableau 4.1. Pour les critères de fenêtre temporelle et de localisation, l'attaquant a trois niveaux de contrôle :

- *Total* : il a un contrôle total sur le critère.
- *Moyen* : il a un contrôle peu précis sur le critère.
- *Aucun* : il n'a aucun contrôle du critère.

Modèle de fautes	Timing	Localisation	Précision	Type de faute
Precise bit error	total	total	bit	set (0) ou reset (1)
Precise byte error	total	total	octets	0x00 ou reset (0xFF) ou aléatoire
Unknown byte error	moyen	moyen	octets	set(0x00) ou reset (0xFF) ou valeur aléatoire
Unknown error	aucun	aucun	variable	aléatoire

TABLEAU 4.1 – Les différents modèles de fautes
[21]

Dans ce tableau, les modèles sont cités par ordre décroissant de puissance de l'attaquant, c.-à-d. que le modèle de fautes "precise bit error" est plus compliqué à mettre en œuvre que le modèle "unknown error".

4.3 Types de fautes

Suivant l'effet recherché par l'attaquant, on peut classer les fautes en destructives, permanentes ou transitoires [6] :

Fautes destructives Les fautes destructives modifient définitivement le comportement du circuit, car une fois injectées, le circuit ne pourra plus être utilisé suivant son comportement initial. Cela peut être provoqué par une émission laser à haut niveau d'énergie sur une cellule mémoire par exemple. Dans ce cas, la cellule mémoire ne peut plus être réécrite. Changer la valeur d'une cellule mémoire (contenant des variables ou du code) définitivement peut être extrêmement efficace, en particulier lorsqu'elles sont liées aux objets sensibles de la carte, tels qu'un PIN ou une clef. Ceci peut être exploite pour contourner des testes de vérification d'un code par exemple.

Fautes Permanentes Les fautes permanentes persistent jusqu'à réinitialisation du circuit ou réécriture dans l'élément mémoire contenant l'erreur provoquée par la faute.

Fautes Transitoires Les fautes transitoires n'affectent le comportement du circuit que sur une durée limitée (un ou plusieurs cycles d'horloge). Le circuit reprend son fonctionnement correct après arrêt de la source de perturbation. L'erreur découlant du comportement fautif d'un ou plusieurs éléments du circuit peut se propager pendant plusieurs cycles d'horloge. Ce type de fautes est le plus communément exploité par les attaques en fautes.

4.4 Scénarii d'attaques

L'injection de fautes est utilisée pour affecter l'opération normale d'une carte à puce. Grâce à des perturbations, l'attaquant peut [18] :

- Modifier un branchement conditionnel pour accéder à des segments de code de sécurité. Une telle modification de branchement peut notamment permettre de réduire le nombre des rondes d'un algorithme de chiffrement par bloc (DES [5], AES [6]). Cette réduction peut aller jusqu'à une seule ronde, rendant triviale la récupération de la clé secrète de chiffrement.
- Étendre le nombre d'itérations d'une routine d'écriture pour présenter des données sur le port série. Ces données peuvent alors être du code ou des clefs stockées dans la mémoire.
- Corrompre des données transférées entre la mémoire et le processeur ou modifier des instructions en cours d'exécution.

D'autres attaques en faute sont, par contre, spécifiques aux algorithmes de cryptographie telles que DFA (Differential Fault Analysis) et les attaques en « safe-error » détaillées dans [18].

4.5 Conséquences

Les conséquences de l'attaque par injection de fautes peuvent être plus ou moins graves. On peut citer des perturbations [21] :

- Dans le **microprocesseur** : en effet, les perturbations peuvent entrainer des modifications dans les registres contenus dans le microprocesseur tels que le pointeur de pile (ou SP pour Stack Pointer), ou le pointeur de code (ou PC pour Program Counter). Une modification du PC peut par exemple faire pointer le code dans une zone mémoire différente de celle qui devait être exécutée.

5. Data Encryption Standard
6. Advanced Encryption Standard

- Dans les **différentes mémoires** : en effet, l'attaque peut aussi avoir lieu dans l'EEPROM ou la mémoire flash, par exemple où peuvent être stockés le code des applications, les clefs cryptographiques, ou même le code PIN de l'utilisateur. Dans la RAM, où l'on peut modifier la structure des objets temporaires manipulés. Le résultat obtenu peut être une inconsistance des données, par contre, notons que les données importantes peuvent être protégées par des techniques de signature.
- Sur les **bus** : la faute sert à modifier les données qui transitent sur le bus. C'est un moyen d'attaquer les informations contenues dans la carte, car les données sauvegardées ou manipulées circulent sur les bus entre les différents composants de la carte à puce. Cela peut dans certains cas faciliter l'attaque des données sauvegardées dans la carte.

En somme, une attaque en faute entraîne des modifications du code, des perturbations dans le flot de contrôle et ou de données, ou une modification des données.

4.6 Contre mesures

Les protections contre les attaques par fautes sont de nature très variées. Elles ont pour but soit de détecter l'attaque soit de la rendre inefficace, et peuvent être déployées à tous les niveaux entre le matériel et l'application.

La première solution pour se protéger contre les attaques par injection de fautes est de s'assurer que les techniques d'induction de ces fautes soient inefficaces ou plus difficiles à mettre en place. Ces techniques tentent de protéger le circuit dans son intégralité et se répartissent en deux catégories : les protections passives et les protections actives [16].

- Les **protections passives** ont pour but de rendre l'induction de fautes plus difficile sans savoir s'il y a vraiment une attaque en cours. Elles sont principalement basées sur la mise en place de boucliers autour du circuit. Ces boucliers permettent l'atténuation de l'effet de tous les moyens d'induction optiques ou électromagnétiques. Contourner ces protections demande donc d'endommager le circuit plus profondément.
- Les **protections actives** ont pour but de détecter qu'une attaque est en cours. Elles sont basées sur l'utilisation de différents capteurs pour détecter toute modification anormale du milieu d'opération. Ceci concerne aussi bien les conditions environnementales telles que la température ou l'éclairage ou les signau. Si ces paramètres sortent des bornes prédéfinies, une attaque est détectée et un signal envoyé au bloc décisionnel pour appliquer la réaction voulue.

Ces protections peuvent être contournées en utilisant des moyens d'induction plus puissants et précis. La précision d'un tir laser permet de passer à côté des détecteurs de lumière et de générer des fautes. Pour cette raison, il faut ajouter des mécanismes de détection d'erreur au sein du circuit. Cette détection est réalisée en utilisant de la redondance [16].

4.6.1 Redondance matérielle

Le principe de la redondance matérielle est de réaliser la même opération sur plusieurs copies d'un même bloc de calcul et d'en comparer les résultats. Ce principe a été utilisé pour développer plusieurs schémas de protection :

- La duplication simple avec comparaison est basée sur l'utilisation de deux copies en parallèle du même bloc, suivies par la comparaison des deux résultats (Figure 4.7).

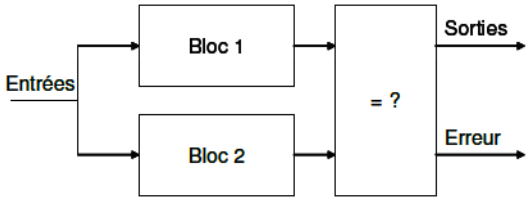

FIGURE 4.7 – Duplication simple avec comparaison
[16]

- La duplication multiple avec comparaison est une extension de la duplication simple à un nombre quelconque de copies du bloc de calcul. La triplication est une des protections les plus utilisées basées sur ce schéma.
- La duplication simple avec redondance complémentaire est une optimisation de la duplication simple en utilisant pour second bloc de calcul la fonction duale du premier. Pour ce faire, il faut donc inverser les entrées et les sorties du second bloc. Cette technique a l'avantage que si la fonction protégée n'est pas autoduale, il sera plus difficile pour un attaquant de générer deux fautes qui aboutiront au même résultat (Figure 4.8).

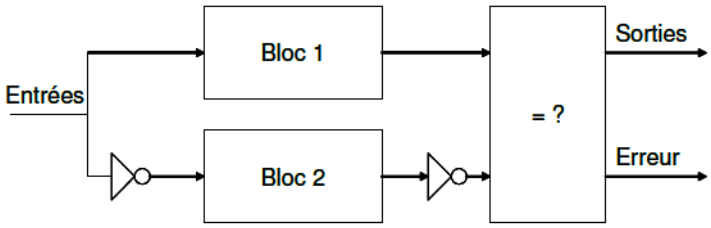

FIGURE 4.8 – Duplication simple avec redondance complémentaire
[16]

- La duplication dynamique est une optimisation de la duplication multiple dans laquelle un bloc qui a donné un résultat erroné (déterminé par vote) ne sera plus utilisé jusqu'à une réinitialisation du dispositif (Figure 4.9).

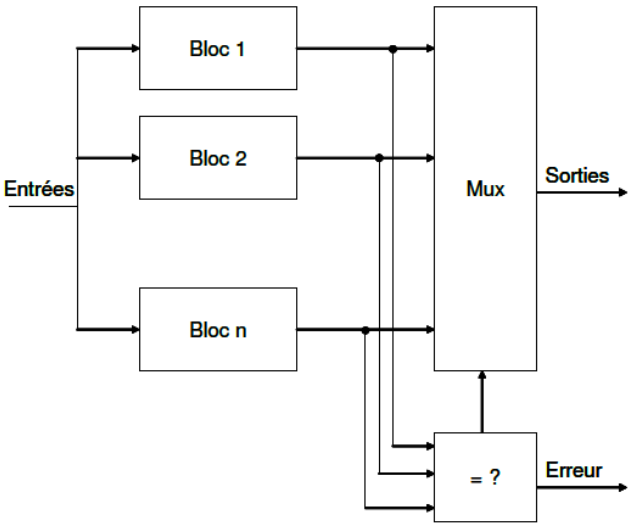

FIGURE 4.9 – Duplication dynamique
[16]

– La duplication hybride est un schéma utilisant la duplication dynamique où les différents blocs redondants sont partiellement des blocs duaux avec les inversions d'entrées sorties nécessaires.

La capacité de détection de ces schémas est dépendante du nombre de copies utilisées : avec 2*n+1 blocs, il est possible de détecter 2*n erreurs et de corriger n erreurs. Dans le cas de la duplication simple et de la duplication simple complémentaire, on détecte une seule erreur sans correction possible.

4.6.2 Redondance temporelle

La redondance temporelle est basée sur la réexécution d'un même calcul sur le même bloc matériel et la comparaison des différents résultats obtenus. Ce principe se décline aussi sur plusieurs schémas de protection.

– La redondance temporelle simple est basée sur la double exécution d'un calcul sur un même bloc de calcul. Les résultats ainsi obtenus sont donc comparés (Figure 2.3).

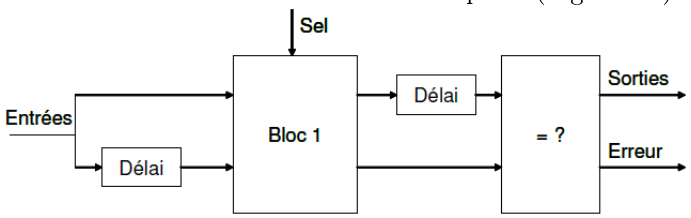

FIGURE 4.10 – Redondance temporelle simple
[16]

– La redondance temporelle multiple se base sur l'exécution multiple de la même opération sur la même unité de calcul.

– La redondance temporelle simple avec opérande inversée utilise le même principe que la redondance simple mais en inversant l'ordre des octets des opérandes (on passe de little endian[7] à big endian[8] et inversement). L'ordre est rétabli avant la comparaison. Il est évident que la fonction ainsi protégée doit avoir certaines propriétés mathématiques permettant cette inversion (Figure 4.11).

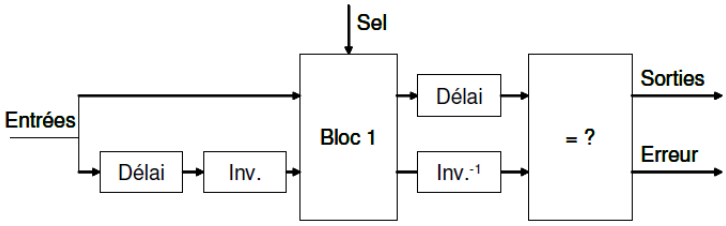

FIGURE 4.11 – Redondance temporelle simple avec opérandes inversées
[16]

– La redondance temporelle simple avec rotation des opérandes utilise certaines propriétés mathématiques de la fonction à protéger permettant l'obtention du même résultat en réalisant une rotation circulaire des entrées et sorties du bloc de calcul (Figure 4.12).

7. Orientation de droite à gauche des octets d'une même donnée.
8. Orientation de gauche à droite des octets d'une même donnée.

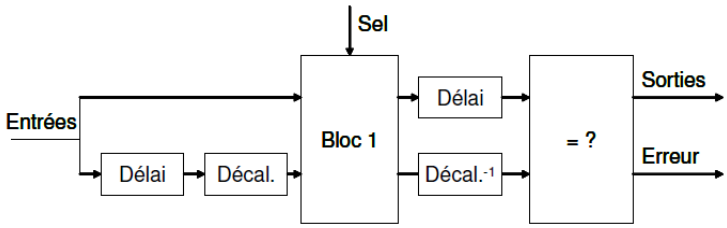

FIGURE 4.12 – Redondance temporelle simple avec rotation des opérandes
[16]

Ces deux grands types de redondance ne sont pas mutuellement exclusifs ; on peut ainsi créer une redondance hybride basée sur l'exécution multiple de la fonction sur plusieurs copies du bloc de calcul. On obtient donc une solution dont le surcoût se répartit sur les deux aspects que sont les performances et la surface (Figure 14).

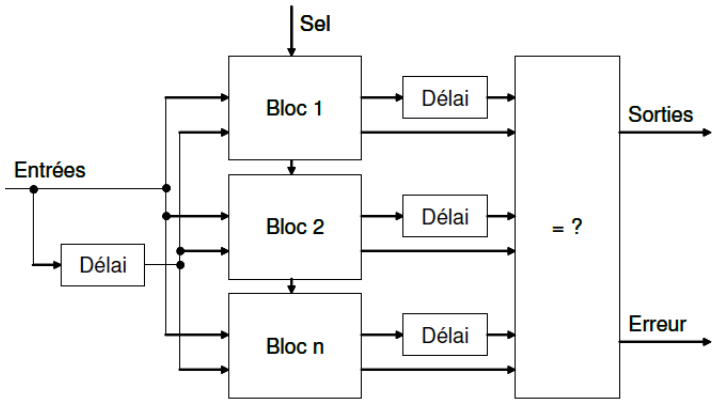

FIGURE 4.13 – Redondance hybride
[16]

4.6.3 Redondance d'information

Les codes correcteurs ou détecteurs d'erreurs sont une autre forme de redondance permettant de vérifier qu'un calcul s'est effectué sans erreur. Le principe est illustré sur la Figure 4.14. Un code est associe au mot obtenu en Sortie du bloc de calcul. Ce code doit être comparé au code attendu pour une Entrée donnée, ce code attendu pouvant être prédit a partir de l'Entrée fournie au Bloc de calcul [6].

Cette approche n'a d'intérêt d'un point de vue coût d'implantation par rapport a une redondance matérielle que si la prédiction et le calcul du code représentent moins de surface qu'une simple duplication du Bloc de calcul [6].

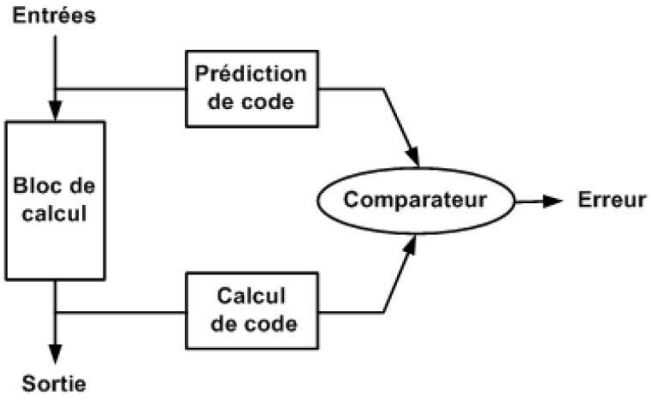

FIGURE 4.14 – Principe de redondance d'information [6]

4.6.4 Protections logicielles

Les protections précédemment citées sont basées sur l'augmentation de la robustesse des coprocesseurs de chiffrement. Les processeurs peuvent aussi être protégés au niveau matériel de la même manière. Cependant, il ne faut pas négliger de protéger le code les pilotant. Les contre mesures logicielles sont utilisées en complément des protections matérielles. Elles permettent d'accroître de manière plus flexible la robustesse du système. Elles visent principalement à protéger le code exécuté sur les processeurs embarqués mais aussi à gérer la prise de décision en cas de détection de faute au niveau matériel. Elles utilisent au niveau logiciel tous les principes énumérés jusqu'ici [16].

- La redondance de variable est une technique basée sur la réexecution d'une opération en utilisant des variables (positions mémoires) différentes. Les différentes variables sont ensuite comparées entre elles pour déterminer s'il y a eu des fautes ou non pendant le calcul. Contrairement à la redondance matérielle, ici les différentes opérations ne sont pas exécutées tout à fait en même temps du fait de la séquentialité du traitement des instructions par le processeur.
- La redondance d'exécution est la simple transposition au niveau logiciel de la redondance temporelle. La fonction est calculée plusieurs fois de suite en ne stockant que les différents résultats obtenus. Les résultats sont ensuite comparés pour détecter la présence de faute.
- L'utilisation de codes correcteurs ou détecteurs est aussi possible pour protéger l'exécution d'un programme. Le but ici est de détecter les erreurs présentes dans les structures de données utilisées dans le programme. Le code le plus utilisé dans ce cas est le contrôle de redondance cyclique (CRC [9]).

Certaines méthodes sont spécifiques à la protection logicielle ; c'est le cas de l'exécution aléatoire. Cette protection est basée sur une modification aléatoire de l'ordre d'exécution des instructions. Elle a pour effet principal de rendre beaucoup plus difficile la prédiction de l'instruction en cours d'exécution par le processeur et donc la détermination de l'instant d'injection de fautes. Il y a cependant des restrictions quant aux modifications qui peuvent être apportées, le but étant bien sûr de conserver la fonctionnalité. Celle ci est la plus efficace si l'attaque requiert plusieurs fautes à des cycles bien précis [16].

La dernière solution utilisée est l'ajout de petites portions de code de test. Régulièrement, ces petits testeurs sont exécutés et si une erreur est détectée, on incrémente un compteur ; au delà d'une certaine valeur, le système est réinitialisé [16].

9. Un outil logiciel permettant de détecter les erreurs de transmission ou de transfert par ajout, combinaison et comparaison de données redondantes, obtenues grâce à une procédure de hachage.

4.7 Conclusion

Les attaques par injection de fautes visant les cartes à puce ont pour principe la création de valeurs logiques erronées au sein du circuit et ont parfois de grave impactes sur le comportement attendu de ce dernier.

Pour réaliser ces attaques, les attaquants modifient certaines conditions environnementales du circuit qui peuvent avoir des effets sur son comportement telles que la tension d'alimentation, la fréquence d'horloge, la température, le voisinage électromagnétique, et l'éclairage.

Les méthodes à leur disposition sont multiples et nombreuses. La plupart d'entre elles ne demandent pas d'investir dans des équipements de pointe et excessivement coûteux. Nous avons vu ici que de nombreuses méthodes de protection peuvent rendre ce type d'attaques plus difficile. Ces protections ne les rendent pas impossibles, et régulièrement les attaques sont affinées pour passer outre les protections existantes, nécessitant ainsi le développement de nouvelles protections.

Conclusion

Les cartes à puce, présentées dans le premier chapitre de ce document, sont de plus en plus présentes autour de nous, elles ont leur place dans la vie de tous les jours. Le nombre d'applications spécifiques à ce support se multiplie de jour en jour. Pour les programmeurs il est nécessaire de pouvoir utiliser un langage évolué et objet pour remplacer le C ou l'assembleur. C'est pourquoi JavaCard est un standard qui tend à devenir une référence.

Après avoir présenté la plateforme Java Card et ces différents aspects (avantages, architecture,etc) dans le chapitre 2, nous pouvons constater qu'elle n'offre pas un niveau de sécurité suffisant pour faire face à toute menace produite par n'importe quel type d'attaque.

Comme les composants de sécurité des cartes à puce contiennent des informations confidentielles, elle font l'objet d'attaques. Celles-ci tentent généralement de porter atteinte à la confidentialité, à l'intégrité ou à l'authenticité des données protégées par les algorithmes de cryptographie embarqués sur ces composants.

Dans le chapitre trois nous avons vu les différents types d'attaques (physiques, logiques ou combinées). Trois familles d'attaques matérielles (physiques) sont distinguées. La première est la famille des attaques non-invasives, qui n'entrainent pas la destruction du matériel (i.e. la carte à puce). La seconde famille est celles des attaques invasives. Dans ce cas, l'attaque opérée permet effectivement de voler l'information, mais le matériel est détruit. Enfin, la troisième famille est celle des attaques semi-invasives.

Dans le quatrième chapitre nous avons présenté les attaques par injection de fautes, ces attaques ont pour principe la création de valeurs logiques erronées au sein du circuit et ont parfois de grave impacts sur le comportement attendu de ce dernier. Pour réaliser ces attaques, les attaquants modifient certaines conditions environnementales du circuit, qui peuvent avoir des effets sur son comportement telles que la tension d'alimentation, la fréquence d'horloge, la température, le voisinage électromagnétique, et l'éclairage. Les méthodes à leur disposition sont multiples et nombreuses. La plupart d'entre elles ne demandent pas d'investir dans des équipements de pointe et excessivement coûteux. Nous avons vu ici que de nombreux méthodes de protection peuvent rendre ce type d'attaques plus difficile. Ces protections ne les rendent pas impossibles, et régulièrement les attaques sont affinées pour passer outre les protections existantes, nécessitant ainsi le développement de nouvelles protections.

Bibliographie

[1] M. BADA : Les problèmes de sécurité dans les systèmes embarqués. , Mémoire de Magister. Université El-Hadj Lakhdar - Batna, 2012.

[2] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall et C. Whelan : The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE, 94(2), 370-382*, Février 2006.

[3] G. BARBU : *De la Sécurité des Plateformes Java Card face aux Attaques Matérielles.* Thèse de doctorat, Institus Mines-Télécom, 2012.

[4] B. BESSON et M. ABERKANE : Les Java Cards. Mémoire Licence. 2004.

[5] M. BOUMASSATA : Vérification de code pour plates-formes embarqués. , Mémoire de Magister. Université El-Hadj Lakhdar-BATNA, 2011.

[6] K. BOUSSELAM : *Résistance des circuits cryptographiques aux attaques en faute.* Thèse de doctorat, Université Montpellier II, Septembre 2012.

[7] J. CARON, K. MONSALLIER et L. ROBET : Histoire de la carte à puce : 25 années d'innovation relayée sur cartes et identification. Paris Nord Villepinte, France, 2010.

[8] J.-F. DHEM, F. KOEUNE, P.-A. LEROUX, P. MESTRE, J.-J. QUISQUATER et J.-L. WILLEMS : A practical implementation of the timing attack. *UCL Crypto Group Technical Report Series*, 1998.

[9] G. DUFAY : *Vérification formelle de la plate-forme Java Card.* Thèse de doctorat, Université DE Nice - Sophia Antipolis UFR Sciences, 2003.

[10] K. GANDOLFI, C. MOURTEL et F. OLIVIER : Electromagnetic analysis : Concrete results. *Cryptographic Hardware and Embedded Systems - CHES 2001, vol. 2162 of Lecture Notes in Computer Science, pp. 251-261, Springer-Verlag*, 2001.

[11] S. HAMADOUCHE : Étude de la sécurité d'un vérifieur de byte code et génération de tests de vulnérabilité. , Mémoire de Magister. Université M'Hamed Bougara De Boumerdes, 2012.

[12] J. IGUCHI-CARTIGNY et J. LANET : Évaluation de l'injection de code malicieux dans une. *Java Card. Proceedings SSTIC 2009, Revues.*

[13] N. KAMEL : *Sécurité des cartes à puce à serveur Web embarqué.* Thèse de doctorat, Université De Limoges, 2012.

[14] A. KARRAY : *Conception, mise en oeuvre et validation d'un environnement logiciel pour le calcul sécurisé sur une grille de cartes à puce de type Java.* Thèse de doctorat, Université Bordeaux I, 2008.

[15] P. C. KOCHER : Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. *Cryptography Research, Inc.*, 1996.

[16] V. MAINGOT : *Conception sécurisée contre les attaques par fautes et par canaux cachés.* Thèse de doctorat, Institut Polytechnique De Grenoble, Juin 2009.

[17] Y. MONNET : *Etude et modélisation de circuits résistants aux attaques non intrusives par injection de fautes.* Thèse de doctorat, Institut National Polytechnique De Grenoble, Avril 2007.

[18] M. Huu NGUYEN : *Sécurisation de processeurs vis-à-vis des attaques par faute et par analyse de la consommation.* Thèse de doctorat, Université Paris 6, 2011.

[19] X. PERRIN, J. JANIER, E. DE CASTRO et E. GOURLAY : , JavaCard. Mémoire de Master. Université des Sciences et Technologies de Lille, 2005.

[20] S. P. SKOROBOGATOV et R. J. ANDERSON : Optical fault induction attacks. *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2002.

[21] A. Al Khary SÉRÉ : *Tissage de contremesures pour machines virtuelles embarquées.* Thèse de doctorat, Université De Limoges, 2010.

[22] Sun Microsystems, Inc, 901 San Antonio Road Palo Alto, CA 94303 USA. *Java Card Applet Developer's Guide*, 1998.

[23] P. URIEN : Introduction à la carte à puce. Telecom ParisTech, 2009.