Programmation pour cartes à puce
Cours 1 - Introduction

Halim Djerroud

LISV
Laboratoire d'ingénierie
des systèmes de Versailles

UVSQ
université PARIS-SACLAY

révision : 0.1

Bibliographie

- Smart Card Handbook (Wolfgang Rankl, Wolfgang Effing)
- 7816 Part 4 : `http://www.ttfn.net/techno/smartcards/iso7816_4.html`

Ce cours est basé sur de nombreux supports et codes sources fournis par Philippe Guillot

Ont participé à l'élaboration de ce cours :

- Philippe Guillot
- Christelle OU
- Kévin CARRIER
- Mehdy Tounsi

LISV Laboratoire d'ingénierie des systèmes de Versailles | UVSQ université PARIS-SACLAY

Histoire de la carte à puce

- Première carte à puce (Carte avec une mémoire) : 1974
  - Breveté par un : Roland Moreno (français)
  - Au départ, c'était une carte à mémoire
  - Objectif : Principalement sécuriser les cabines téléphoniques au moyen de DES
- Première carte à puce intelligente *Smart Card* (+ processeur) : 1976
  - Michel Ugon (français) Employé chez Bull : ajoute un processeur donnant ainsi la faculté aux cartes à puces de traiter l'information
  - L'année 1976 correspond également à l'appel d'offres DES (Data Encryption Standard)1978, le premier algorithme de chiffrement à clé publique, RSA, est publié.
- Entrée des carte à puces dans le domaine publique : 1983
  - Télécartes (cartes à puce pour cabines téléphoniques)
  - Diminution du taux de fraude de 0.2% à 0.032%, c'est 80% de 0.2%
- Utilisation des cartes à puce dans le domaine bancaire : 1983

Histoire de la carte à puce (suite)

- Le marché de la carte à puce a failli disparaître : 1993 et 2000
  - durant cette période, la télévision à péage qui utilise les cartes à puce a subi une série de piratages qui entraîna une forte baisse du marché
  - Des rumeurs désignent la NSA comme suspect...
- La carte SIM a sauvé le marché de la carte à puce : 2000
  - Les cartes SIM (Subscriber Identity Module) représente plus de 75% du marché des cartes à puces
- De nos jours, les cartes à puces sont omniprésentes
  - Exemples : Carte vitale, pièce d'identité, permis de conduire, carte SIM, carte de transport (Navigo), carte bancaire, carte d'étudiant, télé à péage, etc.

Exercice :

- Comptez le nombre de cartes à puce que vous avez dans vos poches.

Marché des cartes à puces

- 2006 : 2.6 Milliards
- 2010 : 6.0 Milliards (88.6 millions de cartes bancaire en France)
- 2012 : 6.9 Milliards
- 2013 : 7.5 Milliards
- 2015 : 8.2 Milliards
- 2020 : 12 Milliards
- Marché en constante évolution

Conclusion

Il y a donc un travail important de cryptologie à effectuer autour de cette technologie

Architecture d'une carte à puce

- La carte à puce qu'on utilisera dans ce cours est basé sur le microcontrôleur ATMega 328, inspirée de la carte *FunCard ATMega163*
- C'est un processeur 8 bits qui utilise une architecture Harvard c'est-à-dire que la mémoire de données et la mémoire programme sont physiquement séparées.
- La mémoire de données est reliée à plusieurs périphériques : une mémoire volatile RAM (Random Access Memory) de 2 Ko, une mémoire non volatile EEPROM (Electrically Erasable Read-Only Memory) de 1 Ko, c'est-à-dire qu'il s'agit d'une mémoire programmable et effaçable de manière électronique, d'un périphérique compteur et d'une entrée/sortie branchée sur les contacts de la puce.

Architecture d'une carte à puce

Une carte à puce contient donc trois types de mémoire :

- une mémoire volatile RAM
- une mémoire non volatile EEPROM
- une mémoire programme non volatile FLASH (effaçable par page)

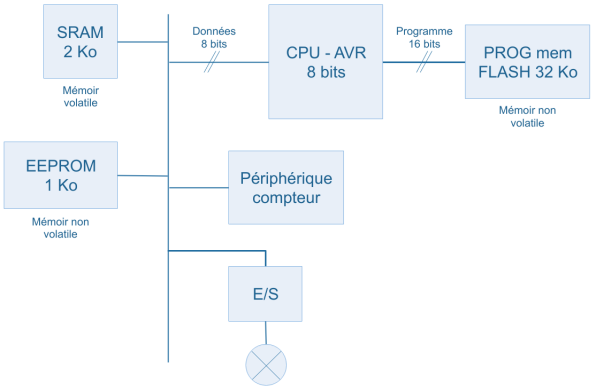
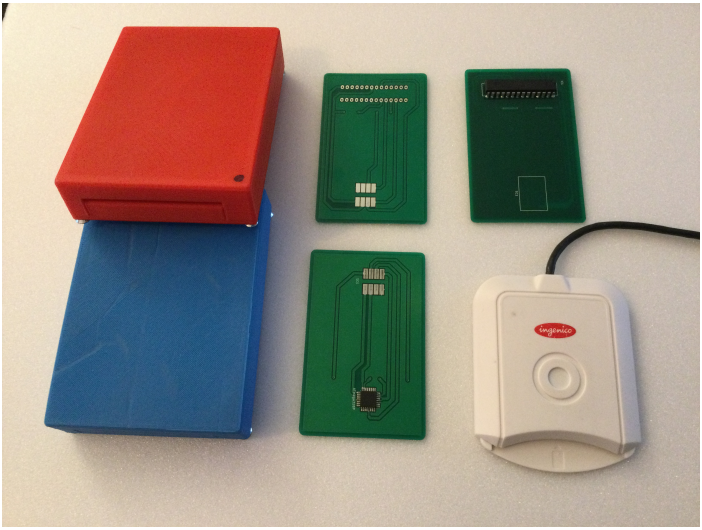


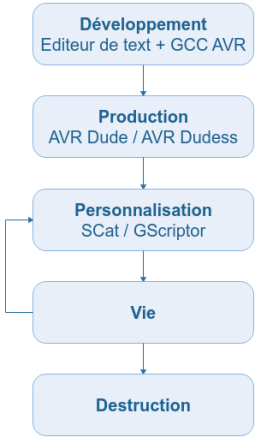
Figure – Architecture interne simplifiée

Carte à puce utilisé dans ce cours



Cycle de vie d'une carte à puce

1. **Développement** : C'est là que l'on imagine le programme en s'aidant de simulateurs ou en faisant des tests en programmation Flash
2. **Programmation / Production** : de la carte si la carte possède une mémoire programme Flash. Réalisation de la carte si cette mémoire est une ROM
3. **Personnalisation** : Cette étape consiste à intégrer de l'information dans l'EEPROM pour rendre une carte unique. Cela peut être un mot de passe, des identifiants, etc
4. **Vie** : La mémoire EPPROM peut être modifié au cours de cette vie
5. **Destruction**



Cartes utilisées dans ce cours

Dans ce cours, nous utiliserons des cartes reprogrammables, le programme réside dans la mémoire FLASH

1. Étape 1 : Nous développerons un code en C sur un IDE comme Emacs, CodeBock etc. Puis AVR-GCC pour compiler notre code
2. Étape 2 : Nous utiliserons le logiciel AVRDUDE (avec un programmateur à base d'arduino connecté par USB)
3. Étape 3 et 4 : Nous allons utilisé le logiciel **SCat**. Bien évidement, nous aurons aussi besoin d'un lecteur de cartes à puce

La norme ISO 7816

- Une carte à puce sans environnement n'a aucun sens.
- Il faut donc définir une norme pour que l'environnement de la carte comprenne cette dernière et inversement
- C'est entre autre ce qui fait son intérêt car ainsi, on peut par exemple utiliser sa carte bancaire n'importe où dans le monde !
- La normalisation des cartes à puce concerne :
  - dimensions de la carte,
  - la position de la puce sur carte
  - les tensions d'alimentation et le brochage de la puce sur la carte
  - **Les paramètres logiciels qui permet de définir comment dialoguer avec la carte : les commandes que celle-ci peut comprendre et comment elle doit réagir face à ces commandes (ce qui nous intéresse dans ce cours)**

La norme ISO 7816

La norme que nous allons utiliser est la norme ISO 7816 disponible sur le site `http://www.iso.ch/`. C'est une norme internationale et payante
Cette norme se divise en 4 parties :

- la norme ISO 7816 – 1 : donnant les caractéristiques physiques
- la norme ISO 7816 – 2 : précisant la position et le brochage des contacts de la carte
- la norme ISO 7816 – 3 : donnant les caractéristiques électriques
- la norme ISO 7816 – 4 : définissant les commandes de base de la carte permettant ainsi de comprendre le dialogue entre une carte et un lecteur

Dans ce cours nous allons essentiellement nous intéresser à la norme ISO 7816 – 4

Aujourd'hui, tous les utilisateurs de cartes rendent plublique leur norme pour des raisons de sécurité.

Fonctionnement du dispositif de travail

1. Reset
2. ART
3. APDU

Figure – Dispositif de travail

Reset

- Une carte est passive, c'est-à-dire qu'elle ne fait rien de sa propre initiative. Le seul moment où elle agit seule, c'est lors du **RESET**, c'est-à-dire lorsqu'on l'insère dans le lecteur.
- Une fois alimentée par le lecteur, elle lui envoie un ATR (Answer To Reset), c'est-à-dire une réponse au **RESET** : elle lui indique son type de carte, les protocoles qu'elle connaît et leurs détails
- Cela permet au lecteur d'identifier la carte qu'il est en train de lire et en cas de non compatibilité, de la rejeter (c'est ce qui se passe quand on insère une carte Navigo dans un distributeur bancaire).

ATR

- Les échanges entre la carte et l'ordinateur se fait octet par octet.
- Dès que la carte est mise sous tension, elle envoie un message de réponse d'initialisation appelé ATR, il peut atteindre une taille maximale de 33 octets.
- L'ATR contient deux octets :
  - `TS=0x3b`, appelé octet initial
  - `T0=0x0n` où **n** désigne un nombre, appelé **octet de format**
- Les quatre premiers bits de l'octet de format sont à 0 et le dernier correspond au nombre d'octets d'historique limité à 15 ($0 \leq n \leq F$)
- Dans l'exemple du programme `Hello.c`, l'octet de format est `0x05` car l'historique "Hello" contient 5 caractères ASCII, donc 5 octets.

ATR

- Il existe deux protocoles de communication :
  - `T=0` qui fonctionne **par octets** (envoi d'octets de données)
  - `T=1` qui fonctionne **par blocs** (envoi de blocs de données)

Dans ce cours, on utilisera le protocole `T=0`. Celui-ci est inscrit dans l'un des symboles binaires de l'octet 0x3b. (Pour plus de détails, voir tiret 3 de la norme ISO 7816).

APDU

- Une fois l'ATR envoyé, la carte se met en attente de commandes. Dans le vocabulaire des cartes à puce, les commandes se nomment APDU (Application Protocole Data Unit).
- Les APDU sont constituées d'un entête contenant 5 octets :

CLA	INS	P1	P2	P3

1. **CLS** : correspond à la classe d'instruction
2. **INS** : désigne le numéro d'instruction
3. **P1** : désigne le paramètre 1
4. **P2** : désigne le paramètre 2
5. **P3** : désigne la taille des données entrantes

- La carte attend une commande, l'exécute, puis envoie une réponse
- Une fois qu'elle a envoyé une réponse, elle se met encore en attente d'une autre commande.

Acquittement et status word

- La réponse de la carte correspond soit à un :
  - acquittement
  - message d'erreur appelé **status word d'erreur**.
- Si les cinq octets de l'entête que reçoit la carte sont tous corrects, elle **acquitte**
  - L'acquittement consiste à renvoyer `INS` ou son complémentaire. Par exemple, si `INS = 0000 0001` alors le complémentaire de `INS` est `1111 1110`. Dans ce cours, on n'utilise pas le complémentaire de `INS`.
- Si la carte ne connaît pas la classe, elle va renvoyer un **status word d'erreur : `6x xx`**

Acquittement et status word

Un numéro d'instruction ne doit commencer ni par 6, ni par 9 car si c'était le cas, le lecteur interprèterait l'acquittement comme une erreur ou comme un status word d'erreur.

Le tableau ci-dessous liste les différents status word d'erreurs qui pourront nous intéresser par la suite

Status Word d'erreur dans la norme ISO	**Signification**
`6e 00`	CLA inconnue
`6d 00`	CLA connue, INS inconnue
`6b 00`	CLA, INS connues mais P1 et P2 incorrects
`6c xx`	CLA, INS, P1 et P2 corrects mais P3 incorrect et xx désigne le P3 attend

Les classes

- Les classes inférieures à `0x80` sont réservées dans la norme. On devra donc utiliser des classes toujours supérieures à `0x80` qui sont des classes librement accessibles pour des applications propriétaires.
- Il existe deux types de commandes : les **commandes entrantes** et les **commandes sortantes**. Une commande peut être soit entrante, soit sortante mais pas les deux à fois.

Les commandes entrantes

- Ce sont des commandes qui vont faire entrer des données dans la carte. Un exemple de ce type de commande est la commande de personnalisation que nous allons programmer. Elle consistera à introduire le nom de l'utilisateur de la carte dans la mémoire non volatile **EEPROM**.
- Après avoir envoyé l'**ATR**, la carte reçoit un entête. Si l'entête est correct, la carte fait un acquittement. Elle reçoit alors des données et les traite. Si tout se passe bien, c'est-à-dire si les données ont bien été écrites dans la mémoire, elle envoie le status word `90 00` qui indique que tout s'est passé convenablement.
- Si l'écriture en mémoire ne s'est pas déroulée comme prévue, elle envoie un status word d'erreur `6x xx`.
- Si l'entête reçu est incorrect, elle envoie un status word d'erreur et se met en attente de nouvelles commandes.

Les commandes entrantes

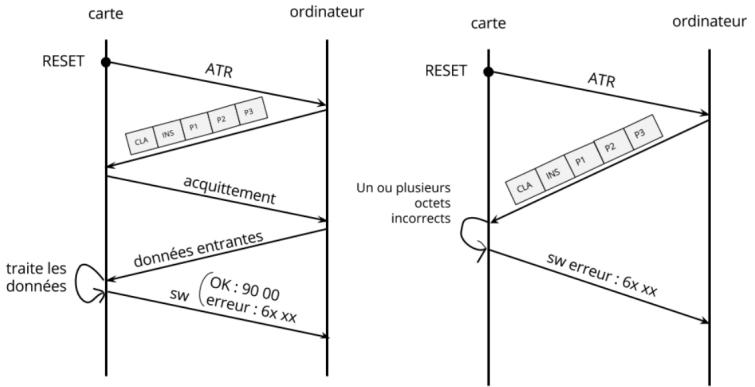


Figure – Exemple d'échanges pour une commande entrante
(à gauche un déroulement sans erreurs, à droite l'entête est incorrect)

Les commandes sortantes

- Ce sont des commandes qui vont faire sortir des données de la carte.

Par convention, une commande sortante se nomme **`getResponse()`** et l'octet **`INS`** doit être égal à 0xC0. Cette commande consiste à lire des données de la carte.

- Après avoir envoyé l'**ATR**, la carte reçoit un entête. Si l'entête est correct, la carte fait un acquittement. Elle « sort » ensuite les données demandées et si tout s'est bien passé, envoie `90 00`, sinon `6x xx`. Si l'entête est incorrect, comme précédemment, elle envoie `6x xx`.

Spécification du paramètre P3

- Quelque soit le type de commande, le paramètre P3 qui correspond à la taille des données, doit toujours être spécifié. Dans le cas d'une commande entrante, il n'y a pas de problèmes. En revanche, pour le cas sortant, c'est moins évident.
- On procèdera en deux temps : par exemple, si la commande sortante est « Quel est votre nom ? », on mettra d'abord P3 à 0. Si la taille attendue est 5, on aura comme erreur `6c 05` et à ce moment-là on pourra tester de nouveau la commande avec le paramètre attendu. Dans cet exemple, on voit donc l'utilité du **satus word d'erreur 6c xx**.

Les commandes sortantes

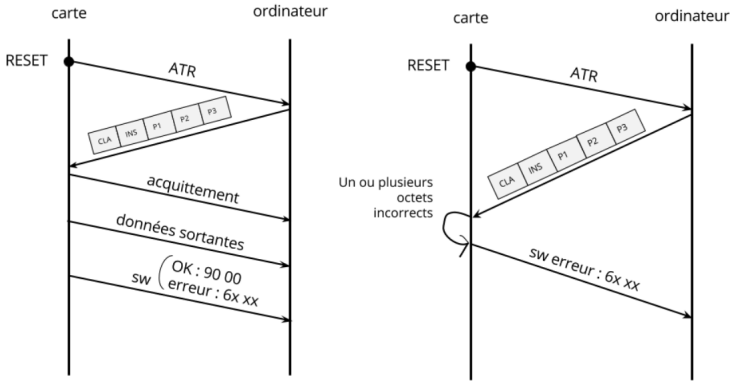


Figure – Exemple d'échanges pour une commande sortante
(à gauche un déroulement sans erreurs, à droite l'entête est incorrect)

Déroulement d'une exécution avec la norme ISO 7816 – 4

1. Initialisation : Rest
2. (carte) → (ordinateur)

   ATR (Anwer To Reset)
   - Octet de protocole : 0x3b (protocole t=0)
   - Octet de format : 0x0n (n= nombre d'octets d'historique)
   - Historique : $h_1 \ldots h_n$
3. Attente des APDU
4. (carte) ← (ordinateur)
5. Si APDU sortante
   - Si OK alors on acquitte : on renvoie INS
   - Sinon, on envoie un code d'erreur 6x xx
6.
7. (carte) → (ordinateur)

   Status word :
   - 6x xx : Erreurs
   - 9x xx : Pas d'erreurs
   - 90 00 : Pas d'erreur du tout