Partie **D. Sauveron** 19 décembre 2011

45 min ***Répondre sur des intercalaires !***

Exercice 1 : RSA

Voici une implémentation d'une signature RSA : $y^a \bmod n$ où `y` est le message à signer, `n` est public et `a`, l'exposant peut être considéré comme la clé secrète (`L` est la taille binaire de la clé).

```
s = 1;                  // s contiendra en fin de calcul une première signature du message y
for (i = L - 1; i >= 0; i--) {
    s = s² mod n;
    if (a[i] == 1)
        s = s*y mod n;
}
t = 1;                  // t contiendra en fin de calcul une seconde signature du message y
u = 1;
for (i = L - 1; i >= 0; i--) {
    t = t² mod n;
    if (a[i] == 1)
        t = t*y mod n;
    else
        u = t * u mod n;
}                                   // Comparaison des signatures
for (i = 0; i <= size(s)-1; i++)
{
    if (s[i] != t[i])
        return u;
}
return s;
```

Remarque : `a[L-1]` contient donc le bit de poids le plus fort et `a[0]` celui de poids le plus faible.

1. Sachant que les signatures en courant de l'opération de *carré modulaire* et de *multiplication modulaire* ont les signatures en consommation en courant présentées ci-dessous, **si cela est possible**, illustrez par un schéma représentant la consommation en courant la faille de cet algorithme qui permet « de lire » les bits de la clé secrète lors d'une signature avec une clé de 4 bits (oui ce n'est pas très solide une clé RSA de 4 bits) dont la valeur est `1101`. Représentez sur un schéma la consommation en courant de l'implémentation de l'algorithme de signature ci-dessus où l'on peut « lire » chaque bit de la clé.

Remarque : On rappellera que la représentation de la consommation se fait en fonction du temps.

2. À quelles attaques l'algorithme ci-dessus est-il sensible et pourquoi ?
3. Quelle(s) contre-mesure(s) proposez-vous pour sécuriser ce calcul (aux niveaux physique et logiciel – d'ailleurs surtout logiciel) ?

Partie **D. Sauveron** 19 décembre 2011

Exercice 2 : Java Card – Nos secrets Favoris !

Documents autorisés : Le cours et les TDs

Le but de cet exercice est d'implanter une applet Java Card qui permet de sauver des favoris (bookmarks). La carte ne supportant pas les caractères, nous stockerons les caractères formant les URLs sous la forme de leur représentation ASCII. Une telle applet, SecureBookmarkApplet, reçoit des URLs de l'extérieur, les stocke et renvoie un index. Elle permet également à l'application extérieure de pouvoir récupérer un bookmark précédemment stocké pour peu qu'elle fournisse un index valide.

On recevra de l'extérieur des URLs avec une taille maximal de 255 octets.

On sauvera toutes les URLs reçues de l'extérieur dans un tableau d'une taille maximale de 32767 octets. Ce tableau sera structuré comme une suite d'enregistrements respectant le format suivant :

- le numéro de l'enregistrement sur un octet
- la taille de l'enregistrement
- les données de l'enregistrement : c'est-à-dire l'URL

index	taille	données	index	taille	données	index	taille	données	...
0x01	0x16	0x68 0x74 0x74 0x70 0x3A 0x2F 0x2F 0x77 0x77 0x77 0x2E 0x67 0x6F 0x6F 0x67 0x6C 0x65 0x2E 0x63 0x6F 0x6D 0x2F	0x02	0x22	...	0x03	0x31	...	...

http://www.google.com/

On pourra donc au plus stocker 127 fois 255 octets de données et on limitera le nombre maximal d'enregistrement à 256 (soit la valeur 0x00 pour le dernier index disponible).

- L'applet acceptera 2 commandes :
  - « SAVE BOOKMARK» permettra d'envoyer l'URL que l'on veut sauver et retournera un `byte` représentant l'index des données sauvées ;

Commande « SAVE BOOKMARK »						
Commande APDU						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0x20	0x30	N/A	N/A	longueur	URL codée en ASCII	1
Réponse APDU						
Optional data	**Status word**	**Meaning of status word**				
index	0x9000	Successful processing				
N/A	0x6999	Card is full				

  - « GET BOOKMARK » permettra de donner l'index (`byte`) du favori qu'on veut récupérer et retourna l'URL ;

Commande «GET BOOKMARK »						
Commande APDU						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0x20	0x32	index	N/A	N/A	N/A	0
Réponse APDU						
Optional data	**Status word**	**Meaning of status word**				
données	0x9000	Successful processing				
N/A	0x6998	Index is not available				

- « DELETE BOOKMARK » permettra de donner l'index (`byte`) du favori qu'on veut effacer. Il devra déplacer favoris qui suivent celui à effacer (sauf si c'est le dernier) afin d'optimiser le tableau afin qu'il reste contiguë et il devra par conséquent les réindexer en enlevant 1 à la valeur de leur index d'origine.

Commande «DELETE BOOKMARK »						
Commande APDU						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0x20	0x34	index	N/A	N/A	N/A	0
Réponse APDU						
Optional data	**Status word**	**Meaning of status word**				
N/A	0x9000	Successful processing				

1. Le code de départ de l'applet est le suivant :

```
package fr.mastercsi.bordeaux;
import javacard.framework.*;
public class SecureBookmarkApplet extends Applet
{
    public static void install(byte[] bArray, short bOffset, byte bLength) throws ISOException
    {
        new SecureBookmarkApplet().register();
    }

    public void process(APDU apdu) throws ISOException
    {
        // Insérer ici le code métier
    }
}
```

2. Écrire le code minimal à placer dans la méthode `process(APDU apdu)` pour traiter les commandes APDU spécifiées ci-dessus et pour renvoyer les codes de retour spécifiés après traitement.
3. Écrire les fonctions réalisant les traitements demandés par les commandes APDUs donnés ci-dessus.