1 Mars 2011

Partie S. Chaumette (Les terminaux du futur)

Question 1.

a. En quoi peut on dire que les téléphones mobiles sont pour partie en échec relativement à l'intégration des technologies qu'ils offrent ?

b. Quel est le mot magique d'une intégration réussi ?

Question 2. Pourquoi, même si des technologies intégrables dans un téléphone mobile ou une carte à puce par exemple existent dès aujourd'hui, ne sont elles pas intégrées par les constructeurs ?

Raison 1 :
Raison 2 :
Raison 3 :

Question 2. Imaginez deux facteurs de forme (différents de ceux vus en cours) de l'ordinateur ou de la carte à puce (ou de ce que j'ai appelé en cours l'alter ego numérique) de demain en précisant l'usage et le public visés.

Facteur et raison 1 :

Facteur et raison 2 :

Question 3.
Grâce à la technologie NFC, intégrée aujourd'hui dans certaines cartes à puce, il est désormais possible de réaliser des paiements, sans contact, avec un téléphone mobile.

Quels peuvent être les principaux freins (en indiquer au maximum trois) à l'adoption de cette technologie ?

Frein 1 :
Frein 2 :
Frein 3 :

Examen – Attaques sur carte à puce

Durée : 54mn

Christophe Giraud
c.giraud@oberthur.com

Exercice 1. Un attaquant mesure la consommation d'une carte lors d'une signature RSA et obtient la courbe représentée en Figure 1.

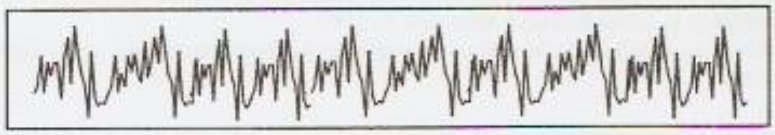


FIGURE 1 – Courbe de consommation observée

1. En observant la courbe, combien de motifs différents distinguez-vous ?
2. Quel type d'algorithme est très probablement utilisé par la carte pour implémenter l'exponentiation modulaire ?
3. Comment l'attaquant en déduit-il la valeur de l'exposant ?

Exercice 2. Analyse différentielle de consommation de courant sur l'AES.

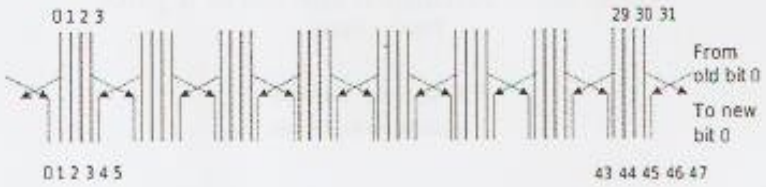
Quelles variables temporaires de l'AES peuvent être attaquées par analyse différentielle en faisant une hypothèse sur 16 bits de clé si l'attaquant dispose uniquement de la valeur du texte chiffré ? La fin de l'AES est schématisée dans le transparent 84 du cours.

Exercice 3. Attaque par injection de fautes sur le DES.

En cours nous avons vu la description détaillée du DES (transparents 65 à 67). En particulier, la permutation expensive E est représentée par la table suivante :

```
E={31, 0, 1, 2, 3, 4, 3, 4, 5, 6, 7, 8, 7, 8, 9,10,11,12,11,12,13,14,15,16,
   15,16,17,18,19,20,19,20,21,22,23,24,23,24,25,26,27,28,27,28,29,30,31, 0};
```

qui se schématise par :



Nous supposons qu'un attaquant peut perturber un seul bit de la partie droite de l'entrée du dernier tour du DES. Cependant la position du bit perturbé est toujours la même à chaque exécution du DES.
Combien de bits de la clé pourra-t-il retrouver au maximum grâce à une analyse DFA ?

Exercice 4.

1. Une implémentation du RSA sur une carte utilise l'algorithme *Square-and-Multiply Always* pour effectuer l'exponentiation modulaire de la signature RSA. Cet algorithme est présenté au transparent 33 du cours.
   (a) Quel est le surcoût de cet algorithme en moyenne par rapport à un *Square-and-Multiply* classique si nous supposons qu'une élévation au carré prend le même temps qu'une multiplication ?
   (b) Quel est son intérêt ?
   (c) Donner l'équivalent de l'algorithme *Square-and-Multiply Always* pour la multiplication scalaire des courbes elliptiques.
2. Le développeur remarque le surcoût de cette méthode. Il décide donc d'utiliser la méthode suivante dite *Atomique* :

Algorithme 1 Signature RSA utilisant une exponentiation atomique

ENTRÉES: Le message m, l'exposant privé $d = (d_{n-1}, \ldots, d_0)_2$ et le modulus N
SORTIE: La signature $S = m^d \bmod N$

1. $R_0 \leftarrow 1$
2. $R_1 \leftarrow m$
3. $i \leftarrow n - 1$
4. $k \leftarrow 0$
5. **while** $(i \geq 0)$ **do**
6. $\quad R_0 \leftarrow R_0 \cdot R_k \bmod N$
7. $\quad k \leftarrow k \oplus d_i$
8. $\quad i \leftarrow i - (k \oplus 1)$
9. **return** R_0

(a) Expliquer pourquoi l'algorithme 1 retourne $m^d \bmod N$.

(b) Est-ce que l'algorithme 1 est plus efficace que la méthode du *Square-and-Multiply Always* si le coût d'une multiplication est le même que celui d'une élévation au carré ? Donner l'éventuel gain/perte moyen comparé à cet algorithme.

(c) Est-ce que l'algorithme 1 est résistant à l'analyse simple observant les événements qui dépendent de la valeur de la clé ?

Master 2 CSI
Université Bordeaux 1

2010-2011

Cartes à Puces
Méthodes Formelles

Exercice 1.

On considère le code source Java suivant, annoté par une spécification Esc/Java. Deux passages de code sont cachés: il ont été remplacés par les expressions `**EXPR1**` et `**EXPR2**`.

```
//@ requires t1 != null && t2 != null;
//@ ensures \result == true <==> (\forall int j; j>=0 & j < t1.length ==> **EXPR1**);
public static boolean compare(int[] t1, int[] t2){
      int i = 0;

      //@ loop_invariant i>=0;
      //@ loop_invariant **EXPR2**
      while (i < t1.length){
            if (t2[i] > t1[i])
                  return false;
            i++;
      }
      return true;
}
```

1. Que fait la fonction `compare(int[]t1,int[]t2)`?

2. La vérification par Esc/Java donne l'avertissement suivant:

```
------------------------------------------------------------------------
Tableaux.java: Warning: Array index possibly too large (IndexTooBig)
if (t2[i] > t1[i])
```

Que proposez vous pour remédier à cela ?

3. Donnez les expressions `**EXPR1**` et `**EXPR2**`.

4. Les deux fonctions suivantes sont décrites de façon informelle. Pour chacune d'entre elles, rédigez une spécification JML ainsi que le code java adéquat également annoté en JML. de telle sorte qu'il puisse être vérifié par Esc/Java. Vous n'utiliserez pas de fonction de l'API java.

(a) `public static int maximum(int[] t);`
dont la fonction consiste à calculer le maximum d'un tableau d'entiers.

(b) `public static void somme(int[] t1, int[] t2, int[] t3);`
dont la fonction consiste à calculer la somme terme-à-terme des deux tableaux d'entiers `t1` et `t2`, et de placer le résultat dans `t3`.

Exercice 2.

On rappelle la spécification EscJava du tri à bulle :

```
public class TriBulle {

    /*****************************************************/
    /*@ requires t != null;                              */
    /*@ requires 0<=i && i<t.length;                     */
    /*@ requires 0<=j && j<t.length;                     */
    /*@ modifies t[i],t[j];                              */
    /*@ ensures t[i] == \old(t[j]) && t[j] == \old(t[i]); */
    /*****************************************************/
    static void echange(int[] t, int i, int j) {
        int c;
        c = t[i];
        t[i] = t[j];
        t[j] = c;
    }

    /*************************************************************************/
    /*@ requires t != null                                                   */
    /*@ requires t.length > 2                                                */
    /*@ modifies t[*]                                                        */
    /*@ ensures (\forall int i,j; 0<=j && j<i && i<t.length; t[j] <= t[i])   */
    /*************************************************************************/
    static void tri_bulle(int[] t) {
        int k,l;

        //@ loop_invariant 0 <= k && k <= t.length-1
        //@ loop_invariant (\forall int i,j; 0<=j && j<=i && i<=k; t[j] <= t[i])
        //@ loop_invariant (\forall int i; k!=0 && k<=i && i<t.length ==> t[k-1] <= t[i])
        for(k=0; k<t.length-1; k++) {
                //@ loop_invariant l>=k && l<t.length
                //@ loop_invariant (\forall int i; l<i && i<t.length ==> t[i] <= t[i])
                //@ loop_invariant (\forall int i,j; 0<=j && j<=i && i<=k; t[j] <= t[i])
                //@ loop_invariant (\forall int i,j; 0<=j && j<k && i>=k && i<t.length ==> t[j] <= t[i])
                for(l=t.length-1; l > k; l--)
                        if (t[l-1] > t[l])
                                echange(t,l-1,l);
        }
    }
}
```

1. Dans la spécification de la fonction echange, expliquez la ligne suivante :

```
/*@ requires 0<=i && i<t.length;   */
```

Est-ce une pré condition ou bien une post-condition ? A quoi sert-elle ?

2. En fait, le comportement de la fonction tri_bulle n'est pas totalement spécifié. Indiquez pourquoi.

Exercice 3.

On considère un système modélisé sous smv de la façon suivante :

```
-----------------------------------------------------------------------
MODULE thermostat
VAR
       mesure : { froid, normal};
ASSIGN
       init(mesure) := { froid, normal };
       next(mesure) := { froid, normal };
-----------------------------------------------------------------------
MODULE systeme(thermostat)
VAR
  etat : { attente, initialisation_chaudiere, marche };
ASSIGN
  init(etat) := attente;
  next(etat) :=
    case
     (etat = attente):
        case
         (thermostat.mesure = froid) : initialisation_chaudiere;
         (thermostat.mesure = normal) : attente;
        esac;
     (etat = initialisation_chaudiere) : marche;
     (etat = marche):
        case
         (thermostat.mesure = froid) : marche;
         (thermostat.mesure = normal) : attente;
    esac;
    esac;
-----------------------------------------------------------------------
MODULE main
VAR
t : thermostat;
s : systeme(t);
-----------------------------------------------------------------------
```

1. Expliquez la ligne de code suivante :

```
...
init(mesure) := { froid, normal };
...
```

2. Rédigez une spécification indiquant que l'état `marche` du système succède toujours à l'état `initialisation_chaudiere`.

3. Expliquez la signification de la spécification suivante; puis indiquez en justifiant votre réponse si le système la satisfait.

```
EG EX EX EX s.etat = marche
```

février 2011

Cartes à puce

Écrit

Exercice 1 : PIN (X points – Damien Sauveron)

Documents autorisés : Le cours et les TDs

1. À quelles attaques l'algorithme suivant de vérification de PIN est-il sensible et pourquoi ?

```
for ( i = 0 ; i <= 7; i++)
    if ( pinCarte [ i ] != pinPresente [ i ] )
        return false ;
    return true;
```

2. Implémenter une version sécurisée de cet algorithme.

Exercice 2 : Java Card (X points – Damien Sauveron) – Jouons un peu !

Documents autorisés : Le cours et les TDs

Le but de cet exercice est d'implanter une applet Java Card qui permet de jouer à un petit jeu entre deux amis. L'un d'eux (ami 1) va utiliser la commande APDU Choisir_Secret pour entrer un nombre secret et l'autre (ami 2) utilisera la commande APDU Deviner_Secret pour essayer de découvrir ce nombre secret. Il aura un maximum de 5 essais pour découvrir le fameux nombre secret. L'applet gérera le compteur d'essais et pour chaque proposition lui indiquera, le nombre d'essais restants et si le nombre secret est supérieur ou inférieur au nombre qu'il a proposé au travers d'un code de retour. Si l'ami 2 échoue dans sa quête (i.e. compteur d'essais à 0), l'applet le lui indiquera au travers d'un code de retour particulier. Les spécifications des commandes et réponses sont données ci-dessous. Le nombre secret sera compris entre 0 et 127.

- Elle acceptera les commandes suivantes :

Commande «Choisir_Secret»						
Commande APDU						
CLA	INS	P1	P2	Lc	Data field	Le
0x80	0x10	secret	N/A	N/A	N/A	0x00
With secret in [0,127]						
Réponse APDU						
Optional data	Status word	Meaning of status word				
	0x9000	Successful processing				
	0x9010	One argument is invalid (i.e. out of the given range)				

Damien SAUVERON damien.sauveron@unilim.fr

février 2011

Commande «Deviner_Secret»						
Commande APDU						
CLA	INS	P1	P2	Lc	Data field	Le
0x80	0x20	v_tried	N/1	N/A	N/A	0x01
With v_tried in [0,127]						
Réponse APDU						
Optional data	Status word	Meaning of status word				
Remaining tries	0x9000	Successful processing (YOU WIN)				
	0x9010	One argument is invalid (i.e. out of the given range)				
Remaining tries	0x9022	secret > v_tried and number of tries ≠ 0				
Remaining tries	0x9024	secret < v_tried and number of tries ≠ 0				
	0x9030	YOU LOSE (number of tries = 0 and secret ≠ v_tried)				

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

```
package fr.unilim.msi.calc;
import javacard.framework.*;
public class CestPlusCestMoins extends Applet
{
    public static void install(byte[] bArray, short bOffset, byte bLength) throws ISOException
    {
        new CestPlusCestMoins().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.
4. Modifier le code de l'applet que vous venez d'écrire pour qu'elle enregistre les différentes propositions faites par l'ami 2 lorsqu'il utilise la commande APDU Deviner_Secret (il y en a donc au maximum 5).

Ajouter également la réinitialisation dans cette « zone » mémoire des propositions avec des « -1 » lorsque l'ami 1 appelle la commande APDU Choisir_Secret pour fixer un nouveau secret.
Enfin ajouter le support de la commande APDU Voir_Propositions_Saisies qui permet de récupérer à tout moment les différentes valeurs présentées par l'ami 2 (i.e. les valeurs qui dans la « zone » sont différentes de « -1 »).

Commande «Voir_Propositions_Saisies»						
Commande APDU						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0x80	0x30	N/A	N/A	N/A	N/A	0x05
Réponse APDU						
Optional data		**Status word**		**Meaning of status word**		
The different values tried (between 0 and 5)		0x9000		Successful processing		

 Damien SAUVERON damien.sauveron@unilim.fr