SeseLab: une plateforme logicielle pour faciliter l'enseignement des attaques physiques

Pablo Rauzy
pablo.rauzy@univ-paris8.fr
pablo.rauzy.name

RESSI 2018

23 mai 2018 @ La Bresse

Contexte

- Cours de sécurité dans une formation non-centrée sur la sécurité :
  - histoire de la cryptologie,
  - cryptographie symétrique,
  - cryptographie asymétrique,
  - cryptanalyse classique,
  - attaques physiques.

Contexte

- Cours de sécurité dans une formation non-centrée sur la sécurité :
  - histoire de la cryptologie,
  - cryptographie symétrique,
  - cryptographie asymétrique,
  - cryptanalyse classique,
  - attaques physiques.

Problème

- Beaucoup d'étudiant·e·s (~50) en TP.
- Enseigner quand même la sécurité matérielle… sans matériel :
  - trop coûteux et encombrant pour 1 ou 2 séances de TP,
  - pas le temps de le faire prendre en main aux étudiant·e·s.
- Malgré tout, nécessité de sensibiliser les étudiant·e·s au sujet.

Proposition

- Développer une plateforme logicielle pour simuler des attaques :
  - utilisable dans une salle de TP classique,
  - facile à prendre en main et à utiliser le temps d'un TP.

SeseLab

- La plateforme est composée de deux parties :
  - du code Python qui simule du matériel,
  - une bibliothèque de manipulation de grands nombres.

Le “matériel”

- Simulation d’un microcontrôleur simple/idéalisé :
  - assembleur réduit, registres, RAM, pas de cache, code en ROM.
- Simulation de la consommation électrique du système :
  - distance de Hamming des mises à jour + poids de Hamming des valeurs.
- Simulation d’injection de fautes :
  - saut d’instructions,
  - faute sur les registres (aléatoire ou à zéro).

SeseASM

- Assembleur ressemblant au MIPS de SPIM (un peu simplifié).
- Conventions sur l'utilisation des registres et de la pile.

La bibliothèque `bignum.asm`

- Les étudiant·e·s doivent implémenter RSA (modexp) en SeseASM.
- Nécessité de manipuler des grands nombres, mais pas l'objet du TP.
- Du coup, SeseLab inclus une bibliothèque de grands nombres :
  - essentiellement affichage, comparaison, addition, multiplication, shift.

Représentation des nombres

- Nombres représentés en base 256 suivi de leur taille.
- Les fonctions manipules des pointeurs sur cette dernière case.
- Représentation du nombre `0x12ab34cd56ef` à l'adresse 100 :

Adresse :	94	95	96	97	98	99	100
Valeur :	0x12	0xab	0x34	0xcd	0x56	0xef	6

Exemple

```
1.  .include bignum.asm
2.  main:
3.    sub r30 r30 #1        ; decrement sp
4.    mov !r30 r31          ; push ra
5.    mov r20 #100          ; address
6.    mov r21 #6            ; size
7.    cal bignum_init
8.    mov !r20,#-6 #18      ; 0x12
9.    mov !r20,#-5 #171     ; 0xab
10.   mov !r20,#-4 #52      ; 0x34
11.   mov !r20,#-3 #205     ; 0xcd
10.   mov !r20,#-2 #86      ; 0x56
11.   mov !r20,#-1 #239     ; 0xef
12.   cal bignum_print
13.   mov r31 !r30          ; pop ra
14.   add r30 r30 #1        ; increment sp
15.   ret
```

Démo...

- Attaque par injection de faute sur CRT-RSA (morceau du second TP).

RSA

- Module du RSA $N = p \cdot q$, avec p et q deux grands nombres premiers.
- Soient e et d tels que $d \cdot e \equiv 1 \mod \varphi(N)$
  - (N, e) est la clef publique,
  - (N, d) est la clef privée.
  - Attaque : calcul d'inverse modulo $\varphi(N) = (p-1)(q-1)$, nécessite de connaître p et q, càd de factoriser N.
- Soit m un message :
  - signature : $s \equiv m^d \mod N$,
  - vérification : $m \equiv s^e \mod N$.

CRT-RSA

- Module du RSA $N = p \cdot q$, avec p et q deux grands nombres premiers.
- Soient e et d tels que $d \cdot e \equiv 1 \mod \varphi(N)$
  - $d_p \doteq d \bmod (p-1)$,
  - $d_q \doteq d \bmod (q-1)$,
  - $i_q \doteq q^{-1} \bmod p$,
  - (N, e) est la clef publique,
  - (p, q, d_p, d_q, i_q) est la clef privée.
- Soit m notre message :
  - $s_p = m^{d_p} \bmod p$,
  - $s_q = m^{d_q} \bmod q$,
  - signature : $s = s_q + q \cdot (i_q \cdot (s_p - s_q) \bmod p)$,
  - vérification : $m \equiv s^e \mod N$.

BellCoRe

- L'attaque BellCoRe consiste à fauter le calcul de s_p ou s_q.
- Par exemple si s_p est fautée comme $\widehat{s_p}$:
  - on récupère une signature fautée $\hat{s}$,
  - on obtient q en calculant $\mathrm{pgcd}(N, s - \hat{s})$,
  - du coup $p = N/q$,
  - ensuite on calcule $d = e^{-1} \bmod (p-1)(q-1)$.

Comment ça marche ?

- $\forall x \in \mathbb{N}$, $\mathrm{pgcd}(N, x)$ ne peut prendre que 4 valeurs :
  - 1, si N et x sont premiers entre eux,
  - p, si x est un multiple de p,
  - q, si x est un multiple de q,
  - N, si x est un multiple de p et de q, i.e., de N.

- Si s_p est fautée (i.e., remplacée par $\widehat{s_p} \neq s_p$) :
  - $s - \hat{s} = q \cdot \big((i_q \cdot (s_p - s_q) \mod p) - (i_q \cdot (\widehat{s_p} - s_q) \mod p)\big)$,

  $\Rightarrow \mathrm{pgcd}(N, s - \hat{s}) = q$.

Démo !

- May the luck be with us...

Perspectives

- Pédagogiques :
  - améliorer les TP existants,
  - créer des TP d'implémentation et d'étude de contremesures.
- SeseLab :
  - rendre SeseASM un peu plus réaliste,
  - développer une GUI pour permettre une meilleure utilisation (inspection mémoire, exécution pas à pas, etc.).