Sécurité et systèmes embarqués

Pablo Rauzy
pr@up8.edu
pablo.rauzy.name/teaching/sese

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS
UFR MITSIC / M1 informatique

Séance 7
Les injections de fautes

Les injections de fautes

Attaques physiques (rappel)

- Un algorithme cryptographique peut être vu de deux façons :
  - d'un côté, c'est un objet mathématique abstrait,
  - de l'autre, c'est un code qui va finir par être exécuté sur du matériel.
- Le premier point de vue correspond à celui de la cryptanalyse classique.
- Le second correspond à celui de la sécurité physique.
- Les attaques physiques tirent partie des caractéristiques spécifiques des implémentations pour retrouver les paramètres secrets utilisés pendant le calcul.
- Ces attaques sont donc moins générales que celles de la cryptanalyse classique, mais elles sont à la fois beaucoup plus puissantes.

- Il existe de nombreux types d'attaques physiques.
- À haut niveau, on peut déjà les classer selon deux axes :
  - *invasives* ou *non-invasives* : faut-il ouvrir ou casser en partie l'implémentation, ou au contraire n'exploiter que des informations naturellement (bien que non-intentionnellement) émises ?
  - *active* ou *passive* : l'attaque agit-elle sur l'implémentation ou se contente-t-elle de l'observer ?
- Aujourd'hui, on va s'intéresser aux attaques non-invasives *actives*.

Attaques par injection de fautes

- Le principe d'*une attaque par injection de faute* est d'induire une erreur dans le calcul pendant celui-ci, par un moyen physique.
- Ce moyen peut être
  - un pulse électromagnétique / un laser visant le système,
  - une variation courte dans l'alimentation électrique du système,
  - ...
- Cela peut provoquer
  - un saut d'instructions,
  - la mise à une valeur aléatoire d'une variable intermédiaire du calcul,
  - la mise à zéro d'une variable intermédiaire du calcul.
- Les techniques de visées en temps et en localisation sont de plus en plus puissantes et précises.

- Les fautes peuvent être utilisées de différentes façons.
- Si on est très précis, on peut contourner un test de contrôle d'accès en sautant des instructions ou en changeant une valeur interprétée comme un booléen.

- Les fautes peuvent être utilisées de différentes façons.
- Si on est très précis, on peut contourner un test de contrôle d'accès en sautant des instructions ou en changeant une valeur interprétée comme un booléen.
- Dans le cadre d'une attaque cryptographique, où l'on cherche à retrouver la clef secrète, c'est différent : l'attaquant récupère le résultat fauté du calcul par la sortie normale du système, et essaye d'exploiter ce résultat.

RSA (rappel)

- RSA est un algorithme de cryptographie asymétrique dont la sécurité repose sur la difficulté du calcul des facteurs premiers de grands nombres
- RSA peut servir au chiffrement où à la signature de message.
- Les deux opérations sont similaires donc on va se concentrer aujourd'hui sur la signature.

- Soit $N = p \cdot q$ le module de notre RSA, 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.
  - retrouver d à partir de la clef publique est compliqué car il faut calculer son inverse modulo $\varphi(N) = (p-1)(q-1)$ ce qui suppose de connaître p et q.

- Soit $N = p \cdot q$ le module de notre RSA, 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.
  - retrouver d à partir de la clef publique est compliqué car il faut calculer son inverse modulo $\varphi(N) = (p-1)(q-1)$ ce qui suppose de connaître p et q.
- Soit m notre message.
- Alors $s \equiv m^d \mod N$ est la signature du message m.
- Et $m \equiv s^e \mod N$ permet la vérification de la signature.

Exponentiation modulaire (rappel)

- On avait déjà vu l'algorithme *square-and-multiply* pour effectuer une exponentiation modulaire rapide :

```
1 s := 1
2 m := m % N
3 tant que d ≠ 0:
4     si d & 1 = 1 alors:
5         s := (s * m) % N
6     fin
7     m := (m * m) % N
8     d := d >> 1
9 fin
```

Exponentiation modulaire (rappel)

- On avait déjà vu l'algorithme *square-and-multiply* pour effectuer une exponentiation modulaire rapide :

```
1 s := 1
2 m := m % N
3 tant que d ≠ 0:
4     si d & 1 = 1 alors:
5         s := (s * m) % N
6     fin
7     m := (m * m) % N
8     d := d >> 1
9 fin
```

- Pour se protéger des attaques par canaux auxiliaires qu'on a vu ensemble, on pourrait proposer ceci :

```
1 s := 1
2 m := m % N
3 tant que d ≠ 0:
4     t := (s * m) % N
5     s := s * (1 - (d & 1)) + t * (d & 1)
6     m := (m * m) % N
7     d := d >> 1
8 fin
```

Exponentiation modulaire (rappel)

- On avait déjà vu l'algorithme *square-and-multiply* pour effectuer une exponentiation modulaire rapide :

```
1 s := 1
2 m := m % N
3 tant que d ≠ 0:
4     si d & 1 = 1 alors:
5         s := (s * m) % N
6     fin
7     m := (m * m) % N
8     d := d >> 1
9 fin
```

- Pour se protéger des attaques par canaux auxiliaires qu'on a vu ensemble, on pourrait proposer ceci :

```
1 s := 1
2 m := m % N
3 tant que d ≠ 0:
4     t := (s * m) % N # faute ici
5     s := s * (1 - (d & 1)) + t * (d & 1)
6     m := (m * m) % N
7     d := d >> 1
8 fin
```

Exponentiation modulaire (rappel)

- On avait déjà vu l'algorithme *square-and-multiply* pour effectuer une exponentiation modulaire rapide :

```
1 s := 1
2 m := m % N
3 tant que d ≠ 0:
4     si d & 1 = 1 alors:
5         s := (s * m) % N
6     fin
7     m := (m * m) % N
8     d := d >> 1
9 fin
```

- Pour se protéger des attaques par canaux auxiliaires qu'on a vu ensemble, on pourrait proposer ceci :

```
1 s := 1
2 m := m % N
3 tant que d ≠ 0:
4     t := (s * m) % N # faute ici = safe-error attack
5     s := s * (1 - (d & 1)) + t * (d & 1)
6     m := (m * m) % N
7     d := d >> 1
8 fin
```

CRT-RSA

- En pratique, dans les systèmes embarqués du type smartcard, les contraintes de ressources sont telles qu'on utilisent une variante optimisée de RSA : *CRT-RSA*.
- CRT-RSA permet de gagner un facteur 4 en vitesse :
  - il remplace l'exponentiation modulaire par deux exponentiations modulaires avec des exposants deux fois plus petits,
  - ces calculs vont 8 fois plus vites,
  - après il ne reste qu'un recombinaison rapide à effectuer.

Définition formelle

- Soit $N = p \cdot q$ le module de notre RSA, avec p et q deux grands nombres premiers.
- Soient e et d tels que $d \cdot e \equiv 1 \mod \varphi(N)$ et
  - $d_p \doteq d \bmod (p-1)$,
  - $d_q \doteq d \bmod (q-1)$,
  - $i_q \doteq q^{-1} \bmod p$,
  - (N, e) est la clef publique,
  - (p, q, d_p, d_q, i_q) est la clef privée.
- Soit m notre message.
- Alors sa signature se calcule comme suit :
  - $s_p = m^{d_p} \bmod p$,
  - $s_q = m^{d_q} \bmod q$,
  - $s = s_q + q \cdot (i_q \cdot (s_p - s_q) \bmod p)$.
- Et $m \equiv s^e \mod N$ permet toujours la vérification de la signature.

BellCoRe

- L'attaque *BellCoRe* (de Bell Communication Research) consiste à retrouver p et q en injectant une faute à peu prêt n'importe où dans le calcul.
- C'est la première attaque par injection de faute (1997).
- Si s_p (resp. s_q) est fauté comme $\widehat{s_p}$ (resp. $\widehat{s_q}$), l'attaquant
  - récupère une signature fautée $\hat{s}$,
  - peut retrouver q (resp. p) en calculant $\mathrm{pgcd}(N, s - \hat{s})$.

Comment ça marche ?

- Pour tout entier x, $\text{pgcd}(N, x)$ 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 fauté (i.e., remplacé par $\widehat{s_p} \neq s_p$) :
  - $s - \hat{s} = q \cdot \big((i_q \cdot (s_p - s_q) \mod p) - (i_q \cdot (\widehat{s_p} - s_q) \mod p)\big)$,

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

- Si s_q est fauté (i.e., remplacé par $\widehat{s_q} \neq s_q$):
  - $s - \hat{s} \equiv (s_q - \widehat{s_q}) - (q \mod p) \cdot i_q \cdot (s_q - \widehat{s_q}) \equiv 0 \mod p$,

  $\Rightarrow \text{pgcd}(N, s - \hat{s}) = p$.

Comment ça marche ?

- Pour tout entier x, $\text{pgcd}(N, x)$ 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 fauté (i.e., remplacé par $\widehat{s_p} \neq s_p$) :
  - $s - \hat{s} = q \cdot \big((i_q \cdot (s_p - s_q) \mod p) - (i_q \cdot (\widehat{s_p} - s_q) \mod p)\big)$,

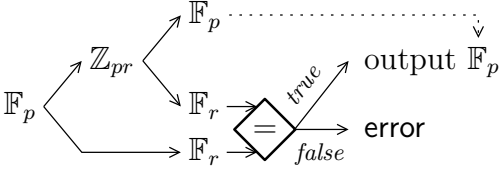
  $\Rightarrow \text{pgcd}(N, s - \hat{s}) = q$.

- Si s_q est fauté (i.e., remplacé par $\widehat{s_q} \neq s_q$):
  - $s - \hat{s} \equiv (s_q - \widehat{s_q}) - (q \mod p) \cdot i_q \cdot (s_q - \widehat{s_q}) \equiv 0 \mod p$,

  $\Rightarrow \text{pgcd}(N, s - \hat{s}) = p$.

Contre-mesures : l'extension modulaire

- Les contre-mesures contre l'attaque BellCoRe sont légions :
  - ∼20 articles depuis 1999,
  - aussi bien du côté académique qu'industriel.
- La plupart sont basées sur une même idée : l'*extension modulaire*.



→ Regardons certaines de ces contremesures avec finja [RG14].