Attacking right-to-left modular exponentiation with timely random faults

Michele Boreale
Dipartimento di Sistemi e Informatica
Università di Firenze – Italy

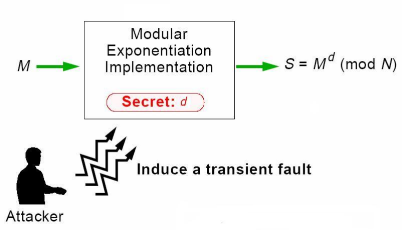
FDTC 2006, Yokohama - October 10, 2006

Smartcards applications and security

- **Applications:** digital signature, pay-TV, credit cards,...
- **Security:**
  1. user authentication: PIN, biometric techniques
  2. internal data access control
     - protection of internal data (tamper-resistance)
     - some data may not even accessible to the card owner (e.g. private keys in public key cryptography)
- **Fault attacks:** directed against tamper-resistance

Fault-based cryptanalysis

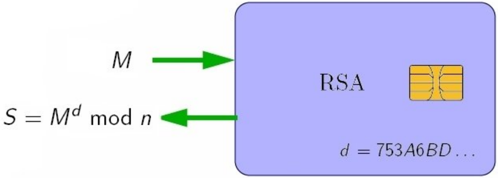
- Fault analysis: induce errors during the computation and observe the effect on the result of the computation
- We shall focus on faults attacks against smartcards implementing digital signature schemes based on traditional public key cryptography (e.g. RSA)
- Goal: extract the private signing key



Fault Analysis: classification of faults

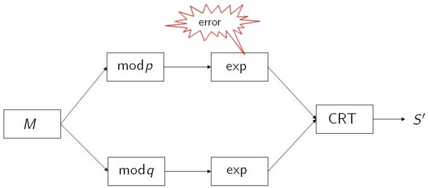
- Permanent
  - e.g. non-reversible modification of memory content
  - may damage smartcard
  - may require sophisticated equipment and expertise
- Transient
  - Affect a single computation (e.g.: a single operation yields an incorrect result)
  - difficult to detect
  - (relatively) easy to induce using a glitch: an instant variation of voltage and/or clock frequency (see e.g. H. Bar-El at al. in FDTC 2004)
- We shall focus on transient faults

RSA signature



- Public parameters: modulus $n = p \cdot q$, public exponent e
- Secret parameters: private exponent d, primes p,q (protected inside the smartcard)
- The user transmits the card a document M to be signed (or its digest)
- The smartcard computes the signature as $S = M^d \bmod n$
- The signature can be verified by checking if $M = S^e \bmod n$

The Bellcore attack (D. Boneh at al. 1996)



- Works against RSA implementation based on the CRT
- Exploits a single random fault
- Exponentiation is computed separately mod p and mod q and recombined using CRT
  - $S_p = M^d$ mod p
  - $S_q = M^d$ mod q
  - $S =$CRT(S_p, S_q)
- $\mathrm{GCD}(S'^e - M, n) = q$

Beyond CRT

Differential Fault Analysis (Feng Bao et al., 1997)

Fault model: a transient fault during computation of S flips a few individual bits of d.

Safe Errors (Yen and Joye, 2000)

Fault model: a transient fault during a modular multiplication alters a selected portion of a data register

Beyond CRT

Differential Fault Analysis (Feng Bao et al., 1997)

Fault model: a transient fault during computation of S flips a few individual bits of d.

Safe Errors (Yen and Joye, 2000)

Fault model: a transient fault during a modular multiplication alters a selected portion of a data register

Unlike Bellcore's, these fault models presuppose the ability to modify the content of data registers in a selective manner. This can be regarded as being a bit too idealized.

Q: Is it possible to achieve similar results using truly random, hence practical faults?
A: Yes, under certain assumptions.

Our fault model

Assumptions

1. *Right-to-left* Exponentiation
2. Multiplication and squaring take constant time δ
3. Attacker has tight control on timing (supplies the clock signal!)
4. A glitch-perturbed squaring $z \leftarrow z^2$ has same effect as $z \leftarrow r$, for a random $r \in \mathbb{Z}_n$

$S = M^d \bmod n$, computed by:

$w \leftarrow 1$

$z \leftarrow M$

phase 0 $\left[\begin{array}{l} \textbf{if } d_0 = 1 \textbf{ then } w \leftarrow w \cdot z \bmod n \\ z \leftarrow z^2 \bmod n \end{array}\right.$

phase 1 $\left[\begin{array}{l} \textbf{if } d_1 = 1 \textbf{ then } w \leftarrow w \cdot z \bmod n \\ z \leftarrow z^2 \bmod n \end{array}\right.$

$\vdots$

$\vdots$

phase l-1 $\left[\begin{array}{l} \textbf{if } d_{l-1} = 1 \textbf{ then } w \leftarrow w \cdot z \bmod n \\ z \leftarrow z^2 \bmod n \end{array}\right.$

return w

Correct signature:

$$S \quad =_n \quad M^{d_0} \cdot M^{2d_1} \quad \cdots \quad M^{2^{i-1}d_{i-1}} \cdot M^{2^i d_i} \cdot M^{2^{i+1}d_{i+1}} \cdots M^{2^{l-1}d_{l-1}}$$

Let $d = (d_{l-1}, \ldots, d_1, d_0)_2$ and assume the least significant $i-1$ bits $d_0, \ldots, d_{i-1}$ have been determined.

The attacker targets bit d_i:

1. determines the time T at which phase i starts
2. applies a glitch at some time in $(T-\delta, T)$: at phase $i-1$ $z \leftarrow z^2$ is replaced by $z \leftarrow r$, with r random
3. obtains a faulty signature S'
4. analyzes S'

glitch at phase $i-1$

$w \leftarrow 1$

$z \leftarrow M$

phase 0: if $d_0 = 1$ then $w \leftarrow w \cdot z \bmod n$; $z \leftarrow z^2 \bmod n$

$\vdots$

phase i-1: $z \leftarrow z^2 \bmod n$ — glitch

phase i (time T): if $d_i = 1$ then $w \leftarrow w \cdot z \bmod n$; $z \leftarrow z^2 \bmod n$

$\vdots$

The attack

Let $d = (d_{l-1}, ..., d_1, d_0)_2$ and assume the least significant $i-1$ bits $d_0, ..., d_{i-1}$ have been determined.

The attacker targets bit d_i:

1. determines the time T at which phase i starts
2. applies a glitch at some time in $(T-\delta, T)$: at phase $i-1$ $z \leftarrow z^2$ is replaced by $z \leftarrow r$, with r random
3. obtains a faulty signature S'
4. analyzes S'

glitch at phase $i-1$

$w \leftarrow 1$

$z \leftarrow M$

phase 0: $\left[\begin{array}{l} \textbf{if } d_0 = 1 \textbf{ then } w \leftarrow w \cdot z \bmod n \\ z \leftarrow z^2 \bmod n \end{array} \right.$

$\vdots$

phase i-1: $\left[\; z \leftarrow z^2 \bmod n \right.$ glitch

phase i (time T): $\left[\begin{array}{l} \textbf{if } d_i = 1 \textbf{ then } w \leftarrow w \cdot z \bmod n \\ z \leftarrow z^2 \bmod n \end{array} \right.$

$\vdots$

Correct vs. **faulty** signature:

$$\begin{array}{llllllll} S & =_n & M^{d_0} \cdot M^{2d_1} & \cdots & M^{2^{i-1}d_{i-1}} & \cdot\, M^{2^i d_i} & \cdot\, M^{2^{i+1}d_{i+1}} & \cdots\, M^{2^{l-1}d_{l-1}} \\ S' & =_n & M^{d_0} \cdot M^{2d_1} & \cdots & M^{2^{i-1}d_{i-1}} & \cdot\, r^{d_i} & \cdot\, r^{2d_{i+1}} & \cdots\, r^{2^{l-i-1}d_{l-1}} \end{array}$$

Analysis of S'

The attacker computes the *Jacobi symbol* of the faulty signature.

Assume for simplicity $\left(\frac{M}{n}\right) = 1$ and $r \in \mathbb{Z}_n^*$. Then:

$$S' =_n M^{d_0} \cdot M^{2d_1} \cdots M^{2^{i-1}d_{i-1}} \cdot r^{d_i} \cdot r^{2d_{i+1}} \cdots r^{2^{l-i-1}d_{l-1}}$$

$$\left(\frac{S'}{n}\right) = \left(\frac{M^{d_0}}{n}\right) \cdot \left(\frac{M^{2d_1}}{n}\right) \cdots \left(\frac{M^{2^{i-1}d_{i-1}}}{n}\right) \cdot \left(\frac{r^{d_i}}{n}\right) \cdot \left(\frac{r^{2d_{i+1}}}{n}\right) \cdots \left(\frac{r^{2^{l-i-1}d_{l-1}}}{n}\right)$$
$$= \left(\frac{r}{n}\right)^{d_i}$$

Thus:

- $\left(\frac{S'}{n}\right) = -1 \Rightarrow d_i = 1$
- $d_i = 0 \Rightarrow \left(\frac{S'}{n}\right) \neq -1$ with "high probability"

A probabilistic decision test for d_i

In other words, the attacker applies the following probabilistic decision algorithm for d_i:

- $J = \left(\frac{S'}{n}\right)$;
- $J = -1$: conclude $d_i = 1$ with certainty;
- $J \neq -1$: conclude $d_i = 0$ with an error probability $\leq \epsilon$.

A probabilistic decision test for d_i

In other words, the attacker applies the following probabilistic decision algorithm for d_i:

- $J = \left(\frac{S'}{n}\right)$;
- $J = -1$: conclude $d_i = 1$ with certainty;
- $J \neq -1$: conclude $d_i = 0$ with an error probability $\leq \epsilon$.

Error probability ϵ. Under reasonable assumptions, for an RSA modulus n, we have $\epsilon \leq \frac{3}{7}$. By repeating the test m times independently, the error probability can be made lower than $(\frac{3}{7})^m$

Software simulations

Several trials have been performed using randomly generated keys of different size

Average numbers of signatures per trial

RSA-256	RSA-384	RSA-512	RSA-768
1389	2110	2816	4227
1532	2285	3065	4206
1670	2502	3321	4970

Rate of unsuccessful trials (%)

RSA-256	RSA-384	RSA-512	RSA-768
19	34	38	57
11	15	25	50
3	6	11	33

For RSA-768, about 5000 faulty signatures would in theory be sufficient to recover a key in about 70% of cases. Estimated time: 25 minutes, assuming a time of 300 μ-s per signature.

Extensions and SW countermeasures

- **Extensions**. The attack works well also in the following cases:
  - prime moduli (hence El Gamal decryption and Diffie-Hellman KE can be targeted)
  - multiplications taking non-constant, normally distributed execution times – with moderate variance
  - RSA with message blinding.

- **SW countermeasures**. The following c.m. appear to thwart our attack:
  - Checking before output ($S^e = M$?), efficient if e is small
  - Blinding of the exponent (add $k \cdot \phi(n)$ to d, for k random, before exponentiation)
  - Shamir's method (Shamir, Eurocrypt 1997), in the case of prime moduli

Conclusions and further work

- We have presented a fault analysis technique on non-CRT public key schemes that combines timing control and truly random computational faults
- Albeit not implemented on real devices, the attack points to more subtleties and dangers arising from faulty behaviour
- Directions for **further work**:
  - Left-to-right exponentiation, no obvious modification works!
  - Double-and-add algorithms used in ECC deserve further investigation