Why One Should Also Secure RSA Public Key Elements

Eric Brier, Benoît Chevallier-Mames,
Mathieu Ciet and **Christophe Clavier**

Gemalto, Security Labs

CHES 2006, Yokohama - October 13, 2006

Outline

1. Introduction
   - Previous work
   - Our attack
   - The threat model
2. Description of the attack
   - Common Principle
   - The bias based variant
   - The collision based variant
   - The full consistency exploitation variant
3. Conclusion
   - Some interesting properties
   - Counter-measures
   - Open problems

Outline

1. Introduction
   - Previous work
   - Our attack
   - The threat model
2. Description of the attack
   - Common Principle
   - The bias based variant
   - The collision based variant
   - The full consistency exploitation variant
3. Conclusion
   - Some interesting properties
   - Counter-measures
   - Open problems

What is it about ?

Fault analysis on public key cryptosystems by corrupting the value of public parameters

What is it about ?

Fault analysis on public key cryptosystems by corrupting the value of public parameters

Motivation

It is usualy considered less important to secure public parameters than private ones

Previous work

- Elliptic Curve Cryptosystems

Previous work

- Elliptic Curve Cryptosystems
  - *Differential Fault Attacks on Elliptic Curve Cryptosystems* [BMV00], Crypto 2000

Previous work

- Elliptic Curve Cryptosystems
    - *Differential Fault Attacks on Elliptic Curve Cryptosystems* [BMV00], Crypto 2000
    - *Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults* [CJ05], Designs Codes and Cryptography, 2005

Previous work

- Elliptic Curve Cryptosystems
  - *Differential Fault Attacks on Elliptic Curve Cryptosystems* [BMV00], Crypto 2000
  - *Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults* [CJ05], Designs Codes and Cryptography, 2005

  Principle: alter public parameters of the curve to make the DL base point to be of small order.

Previous work

- Elliptic Curve Cryptosystems
  - *Differential Fault Attacks on Elliptic Curve Cryptosystems* [BMV00], Crypto 2000
  - *Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults* [CJ05], Designs Codes and Cryptography, 2005

  Principle: alter public parameters of the curve to make the DL base point to be of small order.

- RSA

Previous work

- Elliptic Curve Cryptosystems
  - *Differential Fault Attacks on Elliptic Curve Cryptosystems* [BMV00], Crypto 2000
  - *Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults* [CJ05], Designs Codes and Cryptography, 2005

    Principle: alter public parameters of the curve to make the DL base point to be of small order.

- RSA
  - *On authenticated computing and RSA-based authentication* [Sei05], ACM-CCS 2005

Previous work

- Elliptic Curve Cryptosystems
  - *Differential Fault Attacks on Elliptic Curve Cryptosystems* [BMV00], Crypto 2000
  - *Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults* [CJ05], Designs Codes and Cryptography, 2005

  Principle: alter public parameters of the curve to make the DL base point to be of small order.

- RSA
  - *On authenticated computing and RSA-based authentication* [Sei05], ACM-CCS 2005
  - *Is it wise to publish your Public RSA Keys?* [GS06], FDTC 2006

Previous work

- Elliptic Curve Cryptosystems
  - *Differential Fault Attacks on Elliptic Curve Cryptosystems* [BMV00], Crypto 2000
  - *Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults* [CJ05], Designs Codes and Cryptography, 2005

  Principle: alter public parameters of the curve to make the DL base point to be of small order.

- RSA
  - *On authenticated computing and RSA-based authentication* [Sei05], ACM-CCS 2005
  - *Is it wise to publish your Public RSA Keys?* [GS06], FDTC 2006

  These works allow a chosen message forged signature to be accepted (e.g. malicious applet), but . . .

Previous work

- Elliptic Curve Cryptosystems
  - *Differential Fault Attacks on Elliptic Curve Cryptosystems* [BMV00], Crypto 2000
  - *Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults* [CJ05], Designs Codes and Cryptography, 2005

  Principle: alter public parameters of the curve to make the DL base point to be of small order.

- RSA
  - *On authenticated computing and RSA-based authentication* [Sei05], ACM-CCS 2005
  - *Is it wise to publish your Public RSA Keys?* [GS06], FDTC 2006

  These works allow a chosen message forged signature to be accepted (e.g. malicious applet), but . . .

    - Do not reveal the signer's RSA key

Previous work

- Elliptic Curve Cryptosystems
  - *Differential Fault Attacks on Elliptic Curve Cryptosystems* [BMV00], Crypto 2000
  - *Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults* [CJ05], Designs Codes and Cryptography, 2005

  Principle: alter public parameters of the curve to make the DL base point to be of small order.

- RSA
  - *On authenticated computing and RSA-based authentication* [Sei05], ACM-CCS 2005
  - *Is it wise to publish your Public RSA Keys?* [GS06], FDTC 2006

  These works allow a chosen message forged signature to be accepted (e.g. malicious applet), but . . .

    - Do not reveal the signer's RSA key
    - Rely on some specific fault model

Our attack

Our attack

- Our attack also works by modifying only public elements, but . . .

Our attack

- Our attack also works by modifying only public elements, but ...
- ...applies also to other RSA functions (in standard mode, no CRT):

Our attack

- Our attack also works by modifying only public elements, but ...
- ...applies also to other RSA functions (in standard mode, no CRT):
  - signature (with predictible padding, e.g. FDH or PFDH)

Our attack

- Our attack also works by modifying only public elements, but . . .
- . . . applies also to other RSA functions (in standard mode, no CRT):
  - signature (with predictible padding, e.g. FDH or PFDH)
  - decryption

Our attack

- Our attack also works by modifying only public elements, but . . .
- . . . applies also to other RSA functions (in standard mode, no CRT):
  - signature (with predictible padding, e.g. FDH or PFDH)
  - decryption
- Allows a full break of the secret key (private exponent d is revealed)

Our attack

- Our attack also works by modifying only public elements, but ...
- ...applies also to other RSA functions (in standard mode, no CRT):
  - signature (with predictible padding, e.g. FDH or PFDH)
  - decryption
- Allows a full break of the secret key (private exponent d is revealed)
- Comes in three flavours, one of which does not rely on any fault model

Our attack

- Our attack also works by modifying only public elements, but ...
- ...applies also to other RSA functions (in standard mode, no CRT):
  - signature (with predictible padding, e.g. FDH or PFDH)
  - decryption
- Allows a full break of the secret key (private exponent d is revealed)
- Comes in three flavours, one of which does not rely on any fault model
- Not realized in practice, but validated by extensive simulations

The threat model

The threat model

- Given a public RSA key (e, n), the attacker is able to obtain many faulty signatures for known varying inputs.

The threat model

- Given a public RSA key (e, n), the attacker is able to obtain many faulty signatures for known varying inputs.
- A faulty signature is one computed modulo a corrupted modulus value n':

$$s' = \mu^d \bmod n'$$

The threat model

- Given a public RSA key (e, n), the attacker is able to obtain many faulty signatures for known varying inputs.
- A faulty signature is one computed modulo a corrupted modulus value n':

$$s' = \mu^d \bmod n'$$

Example: On a smart card, the modulus value is altered during transfert from NVM to RAM.

Outline

1 Introduction
- Previous work
- Our attack
- The threat model

2 Description of the attack
- Common Principle
- The bias based variant
- The collision based variant
- The full consistency exploitation variant

3 Conclusion
- Some interesting properties
- Counter-measures
- Open problems

Common Principle

- Our attack comes with three variants:

Common Principle

- Our attack comes with three variants:
  1. The *bias based* variant
     Does not rely on any fault model

Common Principle

- Our attack comes with three variants:
  1. The *bias based* variant
     Does not rely on any fault model
  2. The *collision based* variant

Common Principle

- Our attack comes with three variants:
  1. The *bias based* variant
     Does not rely on any fault model
  2. The *collision based* variant
  3. The *full consistency exploitation* variant

Common Principle

- Our attack comes with three variants:
  1. The *bias based* variant
     Does not rely on any fault model
  2. The *collision based* variant
  3. The *full consistency exploitation* variant

- All variants aim at accumulating the knowledge of $d \bmod q_j$ for many small primes q_j.

Common Principle

- Our attack comes with three variants:
  1. The *bias based* variant
     Does not rely on any fault model
  2. The *collision based* variant
  3. The *full consistency exploitation* variant
- All variants aim at accumulating the knowledge of $d \bmod q_j$ for many small primes q_j.
- Whenever

$$\prod_j q_j \geqslant d$$

d may be retrieved using Chinese Remainder Theorem techniques.

Common Principle

- Our attack comes with three variants:
    1. The *bias based* variant
       Does not rely on any fault model
    2. The *collision based* variant
    3. The *full consistency exploitation* variant

- All variants aim at accumulating the knowledge of $d \bmod q_j$ for many small primes q_j.

- Whenever

$$\prod_j q_j \geqslant d$$

  d may be retrieved using Chinese Remainder Theorem techniques.

- Variants 2 and 3 rely on a fault model, but need much less fault injections than variant 1 (and than [Sei05]).

The *bias based* variant

The *bias based* variant

- The attacker obtains many signatures for the computation of which the modulus was corrupted:

$$s'_i = \mu_i^d \bmod n'_i \qquad i = 1, 2, \ldots$$

The *bias based* variant

- The attacker obtains many signatures for the computation of which the modulus was corrupted:

$$s'_i = \mu_i^d \bmod n'_i \qquad i = 1, 2, \ldots$$

  - $\mu_i = \mathtt{hash}(m_i)$

The *bias based* variant

- The attacker obtains many signatures for the computation of which the modulus was corrupted:

$$s'_i = \mu_i^d \bmod n'_i \qquad i = 1, 2, \ldots$$

  - $\mu_i = \mathtt{hash}(m_i)$
  - Inputs m_i may be arbitrarily chosen

The *bias based* variant

- The attacker obtains many signatures for the computation of which the modulus was corrupted:

$$s'_i = \mu_i^d \bmod n'_i \qquad i = 1, 2, \ldots$$

  - $\mu_i = \mathtt{hash}(m_i)$
  - Inputs m_i may be arbitrarily chosen

- He thus collects many couples (μ_i, s'_i, n'_i)

  Faulty moduli n'_i are unknown from the attacker who only knows (μ_i, s'_i)

The *bias based* variant

- The attacker obtains many signatures for the computation of which the modulus was corrupted:

$$s'_i = \mu_i^d \bmod n'_i \qquad i = 1, 2, \ldots$$

  - $\mu_i = \texttt{hash}(m_i)$
  - Inputs m_i may be arbitrarily chosen

- He thus collects many couples (μ_i, s'_i, n'_i)

  Faulty moduli n'_i are unknown from the attacker who only knows (μ_i, s'_i)

- For any given small prime q, let p be the smallest prime *s.t.* $q \mid p - 1$

  (Possible generalization : $q^e \mid \varphi(p^a)$)

The *bias based* variant

- The attacker obtains many signatures for the computation of which the modulus was corrupted:

$$s'_i = \mu_i^d \bmod n'_i \qquad i = 1, 2, \ldots$$

  - $\mu_i = \mathtt{hash}(m_i)$
  - Inputs m_i may be arbitrarily chosen

- He thus collects many couples (μ_i, s'_i, n'_i)

  Faulty moduli n'_i are unknown from the attacker who only knows (μ_i, s'_i)

- For any given small prime q, let p be the smallest prime *s.t.* $q \mid p - 1$

  (Possible generalization : $q^e \mid \varphi(p^a)$)

- Considering equation

$$s'_i = \mu_i^d \bmod p \ ,$$

  a statistical process on the collection $(\mu_i, s'_i)_i$ will reveal the value $d \bmod q$

The *bias based* variant

Notation

Let $q \mid p - 1$, and $\mu \in (\mathbb{Z}/p\mathbb{Z})^*$

We denote:

The *bias based* variant

Notation

Let $q \mid p-1$, and $\mu \in (\mathbb{Z}/p\mathbb{Z})^*$

We denote:

- $\mathsf{DL}(\mu, s', p)$ the discrete logarithm of s' to the base μ (provided $s' \in \langle\mu\rangle$)

The *bias based* variant

Notation

Let $q \mid p-1$, and $\mu \in (\mathbb{Z}/p\mathbb{Z})^*$

We denote:

- $\mathrm{DL}(\mu, s', p)$ the discrete logarithm of s' to the base μ (provided $s' \in \langle\mu\rangle$)
- $\mathrm{DL}(\mu, s', p, q) = \mathrm{DL}(\mu, s', p) \bmod q$ (provided $q \mid \mathrm{ord}_p(\mu)$)

The *bias based* variant

Notation

Let $q \mid p-1$, and $\mu \in (\mathbb{Z}/p\mathbb{Z})^*$

We denote:

- $\mathrm{DL}(\mu, s', p)$ the discrete logarithm of s' to the base μ (provided $s' \in \langle\mu\rangle$)
- $\mathrm{DL}(\mu, s', p, q) = \mathrm{DL}(\mu, s', p) \bmod q$ (provided $q \mid \mathrm{ord}_p(\mu)$)

Theorem

The *bias based* variant

Notation

Let $q \mid p-1$, and $\mu \in (\mathbb{Z}/p\mathbb{Z})^*$

We denote:

- $\mathrm{DL}(\mu, s', p)$ the discrete logarithm of s' to the base μ (provided $s' \in \langle\mu\rangle$)
- $\mathrm{DL}(\mu, s', p, q) = \mathrm{DL}(\mu, s', p) \bmod q$ (provided $q \mid \mathrm{ord}_p(\mu)$)

Theorem

- *If* $p \mid n'$ *then, whenever* $\mathrm{DL}(\mu, s', p, q)$ *exists, we have:*

$$\mathrm{DL}(\mu, s', p, q) = d \bmod q$$

The *bias based* variant

Notation

Let $q \mid p-1$, and $\mu \in (\mathbb{Z}/p\mathbb{Z})^*$

We denote:

- $\mathrm{DL}(\mu, s', p)$ the discrete logarithm of s' to the base μ (provided $s' \in \langle \mu \rangle$)
- $\mathrm{DL}(\mu, s', p, q) = \mathrm{DL}(\mu, s', p) \bmod q$ (provided $q \mid \mathrm{ord}_p(\mu)$)

Theorem

- *If* $p \mid n'$ *then, whenever* $\mathrm{DL}(\mu, s', p, q)$ *exists, we have:*

$$\mathrm{DL}(\mu, s', p, q) = d \bmod q$$

- *If* $p \nmid n'$ *then,* $\mathrm{DL}(\mu, s', p, q)$ *is supposed to be uniformly randomly distributed over the integers modulo* q

The *bias based* variant

- As the sum of two components, the statistical distribution of $\mathrm{DL}(\mu, s', p, q)$ shows a bias:

The *bias based* variant

- As the sum of two components, the statistical distribution of $\mathrm{DL}(\mu, s', p, q)$ shows a bias:
  - With probability $\frac{p-1}{p}$, $\mathrm{DL}(\mu, s', p, q)$ is drawn from a uniform distribution

The *bias based* variant

- As the sum of two components, the statistical distribution of $\mathrm{DL}(\mu, s', p, q)$ shows a bias:
  - With probability $\frac{p-1}{p}$, $\mathrm{DL}(\mu, s', p, q)$ is drawn from a uniform distribution
  - With probability $\frac{1}{p}$, $\mathrm{DL}(\mu, s', p, q)$ is drawn from a Dirac distribution centered on $d \bmod q$

The *bias based* variant

- As the sum of two components, the statistical distribution of $\mathrm{DL}(\mu, s', p, q)$ shows a bias:
  - With probability $\frac{p-1}{p}$, $\mathrm{DL}(\mu, s', p, q)$ is drawn from a uniform distribution
  - With probability $\frac{1}{p}$, $\mathrm{DL}(\mu, s', p, q)$ is drawn from a Dirac distribution centered on $d \bmod q$

With enough faulty samples, the statistical bias in the distribution of $\mathrm{DL}(\mu, s', p, q)$ will make the correct value $d \bmod q$ emerge

The *bias based* variant

- As the sum of two components, the statistical distribution of $\mathrm{DL}(\mu, s', p, q)$ shows a bias:
  - With probability $\frac{p-1}{p}$, $\mathrm{DL}(\mu, s', p, q)$ is drawn from a uniform distribution
  - With probability $\frac{1}{p}$, $\mathrm{DL}(\mu, s', p, q)$ is drawn from a Dirac distribution centered on $d \bmod q$

With enough faulty samples, the statistical bias in the distribution of $\mathrm{DL}(\mu, s', p, q)$ will make the correct value $d \bmod q$ emerge

The private exponent of a 1024-bit key is fully retrieved within 20,000 faults

Dictionary of faulty moduli

Dictionary of faulty moduli

- Let S be the set of all reachable values for a faulty modulus

Dictionary of faulty moduli

- Let S be the set of all reachable values for a faulty modulus
- This dictionary depends on:

Dictionary of faulty moduli

- Let S be the set of all reachable values for a faulty modulus
- This dictionary depends on:
  - the correct value n of the modulus,

Dictionary of faulty moduli

- Let S be the set of all reachable values for a faulty modulus
- This dictionary depends on:
  - the correct value n of the modulus,
  - a given fault model,

Dictionary of faulty moduli

- Let S be the set of all reachable values for a faulty modulus
- This dictionary depends on:
  - the correct value n of the modulus,
  - a given fault model,
  - assumptions on fault injection precision, chip architecture, counter-measures...

Dictionary of faulty moduli

- Let S be the set of all reachable values for a faulty modulus
- This dictionary depends on:
  - the correct value n of the modulus,
  - a given fault model,
  - assumptions on fault injection precision, chip architecture, counter-measures...

Example

Model: random register value   Architecture: 8 bits   Injection: precise (no CM)

n	92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB
$\|S\| = 256$	92DC**230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB

Dictionary of faulty moduli

- Let S be the set of all reachable values for a faulty modulus
- This dictionary depends on:
  - the correct value n of the modulus,
  - a given fault model,
  - assumptions on fault injection precision, chip architecture, counter-measures. . .

Example

Model: random register value   Architecture: 8 bits   Injection: precise (no CM)

n	92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB
$\|S\| = 256$	92DC**230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB

Example

Model: random register value   Architecture: 32 bits   Injection: precise (no CM)

n	92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB
$\|S\| = 2^{32}$	92DC1423********FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB

Example

Model: random register value   Arch.: 8 bits   Injection: unprecise (random order or delay)

n	92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB
$\|S\| = 2^{15}$ (1024 bits)	**DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB 92**14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB 92DC**230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB ... 92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256**FB 92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9**

Example

Model: random register value Arch.: 8 bits Injection: unprecise (random order or delay)

n	92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB
$\lvert S\rvert = 2^{15}$ (1024 bits)	**DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB 92**14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB 92DC**230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB ... 92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256**FB 92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9**

Example

Model: fixed register value (0) Arch.: 32 bits Injection: unprecise (random order or delay)

n	92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB
$\lvert S\rvert = 32$ (1024 bits)	000000000A32B821FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB 92DC142300000000FF23ED094B18A0C83729420C928CD020A0EE29023256F9FB ... 92DC14230A32B821FF23ED094B18A0C83729420C928CD020A0EE290200000000

The *collision based* variant

- The *collision based* variant needs a dictionary S of possible faulty moduli.

The *collision based* variant

- The *collision based* variant needs a dictionary S of possible faulty moduli.
- It aims at identifying, for some (μ_i, s_i'), which faulty modulus value $n_i' \in S$ actually occured.

The *collision based* variant

- The *collision based* variant needs a dictionary S of possible faulty moduli.
- It aims at identifying, for some (μ_i, s'_i), which faulty modulus value $n'_i \in S$ actually occured.

Definition

Let $\nu \in S$, a hit for ν is the identification of some (μ_i, s'_i) for which $n'_i = \nu$

The *collision based* variant

- The *collision based* variant needs a dictionary S of possible faulty moduli.
- It aims at identifying, for some (μ_i, s'_i), which faulty modulus value $n'_i \in S$ actually occured.

Definition

Let $\nu \in S$, a hit for ν is the identification of some (μ_i, s'_i) for which $n'_i = \nu$

- Once a hit for n'_i is obtained, it is possible to derive $d \bmod q$ for (almost) all primes q verifying $q \mid p - 1$ where p is a known prime factor of n'_i:

$$d \bmod q = \mathrm{DL}(\mu_i, s'_i, p, q)$$

The *collision based* variant

- The *collision based* variant needs a dictionary S of possible faulty moduli.
- It aims at identifying, for some (μ_i, s_i'), which faulty modulus value $n_i' \in S$ actually occured.

Definition

Let $\nu \in S$, a hit for ν is the identification of some (μ_i, s_i') for which $n_i' = \nu$

- Once a hit for n_i' is obtained, it is possible to derive $d \bmod q$ for (almost) all primes q verifying $q \mid p - 1$ where p is a known prime factor of n_i':

$$d \bmod q = \mathrm{DL}(\mu_i, s_i', p, q)$$

- Each hit yields more than 50 bits of modular information about d on average.

The *collision based* variant

- The *collision based* variant needs a dictionary S of possible faulty moduli.
- It aims at identifying, for some (μ_i, s'_i), which faulty modulus value $n'_i \in S$ actually occured.

Definition

Let $\nu \in S$, a hit for ν is the identification of some (μ_i, s'_i) for which $n'_i = \nu$

- Once a hit for n'_i is obtained, it is possible to derive $d \bmod q$ for (almost) all primes q verifying $q \mid p - 1$ where p is a known prime factor of n'_i:

$$d \bmod q = \mathrm{DL}(\mu_i, s'_i, p, q)$$

- Each hit yields more than 50 bits of modular information about d on average.

$\rightarrow$ Only about 10 to 20 hits suffice to recover the private exponent.

How to identify hits ?

How to identify hits ?

- For as much $\nu \in S$ as possible, find some *marker* (p_ν, q_ν) verifying:

How to identify hits ?

- For as much $\nu \in S$ as possible, find some *marker* (p_ν, q_ν) verifying:
    - q_ν is a *not too small* prime (say 10^6 to 10^9)

How to identify hits ?

- For as much $\nu \in S$ as possible, find some *marker* (p_ν, q_ν) verifying:
  - q_ν is a *not too small* prime (say 10^6 to 10^9)
  - $q_\nu \mid p_\nu - 1$ and $p_\nu \mid \nu$

How to identify hits ?

- For as much $\nu \in S$ as possible, find some *marker* (p_ν, q_ν) verifying:
  - q_ν is a *not too small* prime (say 10^6 to 10^9)
  - $q_\nu \mid p_\nu - 1$ and $p_\nu \mid \nu$
- For each $i = 1, 2, \ldots$, compute $\mathrm{DL}(\mu_i, s'_i, p_\nu, q_\nu)$ for all markers (p_ν, q_ν).

How to identify hits ?

- For as much $\nu \in S$ as possible, find some *marker* (p_ν, q_ν) verifying:
  - q_ν is a *not too small* prime (say 10^6 to 10^9)
  - $q_\nu \mid p_\nu - 1$ and $p_\nu \mid \nu$
- For each $i = 1, 2, \ldots$, compute $\mathrm{DL}(\mu_i, s'_i, p_\nu, q_\nu)$ for all markers (p_ν, q_ν).
- Each $\mathrm{DL}(\mu_i, s'_i, p_\nu, q_\nu)$ gives an hypothesis for $d \bmod q_\nu$ which is ...

How to identify hits ?

- For as much $\nu \in S$ as possible, find some *marker* (p_ν, q_ν) verifying:
  - q_ν is a *not too small* prime (say 10^6 to 10^9)
  - $q_\nu \mid p_\nu - 1$ and $p_\nu \mid \nu$
- For each $i = 1, 2, \ldots$, compute $\mathrm{DL}(\mu_i, s'_i, p_\nu, q_\nu)$ for all markers (p_ν, q_ν).
- Each $\mathrm{DL}(\mu_i, s'_i, p_\nu, q_\nu)$ gives an hypothesis for $d \bmod q_\nu$ which is . . .
  - correct if $n'_i = \nu$

How to identify hits ?

- For as much $\nu \in S$ as possible, find some *marker* (p_ν, q_ν) verifying:
  - q_ν is a *not too small* prime (say 10^6 to 10^9)
  - $q_\nu \mid p_\nu - 1$ and $p_\nu \mid \nu$
- For each $i = 1, 2, \ldots$, compute $\mathrm{DL}(\mu_i, s'_i, p_\nu, q_\nu)$ for all markers (p_ν, q_ν).
- Each $\mathrm{DL}(\mu_i, s'_i, p_\nu, q_\nu)$ gives an hypothesis for $d \bmod q_\nu$ which is . . .
  - correct if $n'_i = \nu$
  - random in $\{0, \ldots, q_\nu - 1\}$ with high probability if $n'_i \neq \nu$

How to identify hits ?

- For as much $\nu \in S$ as possible, find some *marker* (p_ν, q_ν) verifying:
  - q_ν is a *not too small* prime (say 10^6 to 10^9)
  - $q_\nu \mid p_\nu - 1$ and $p_\nu \mid \nu$
- For each $i = 1, 2, \ldots$, compute $\mathrm{DL}(\mu_i, s'_i, p_\nu, q_\nu)$ for all markers (p_ν, q_ν).
- Each $\mathrm{DL}(\mu_i, s'_i, p_\nu, q_\nu)$ gives an hypothesis for $d \bmod q_\nu$ which is ...
  - correct if $n'_i = \nu$
  - random in $\{0, \ldots, q_\nu - 1\}$ with high probability if $n'_i \neq \nu$
- A hit will be identified as soon as a collision of DL will occur for some q_ν:

$$\mathrm{DL}(\mu_i, s'_i, p_\nu, q_\nu) = \mathrm{DL}(\mu_j, s'_j, p_\nu, q_\nu) \implies n'_i = n'_j = \nu$$

(see the paper for a discussion on false positive occurence probability)

How to identify hits ?

- For as much $\nu \in S$ as possible, find some *marker* (p_ν, q_ν) verifying:
  - q_ν is a *not too small* prime (say 10^6 to 10^9)
  - $q_\nu \mid p_\nu - 1 \quad \text{and} \quad p_\nu \mid \nu$
- For each $i = 1, 2, \ldots$, compute $\mathrm{DL}(\mu_i, s_i', p_\nu, q_\nu)$ for all markers (p_ν, q_ν).
- Each $\mathrm{DL}(\mu_i, s_i', p_\nu, q_\nu)$ gives an hypothesis for $d \bmod q_\nu$ which is ...
  - correct if $n_i' = \nu$
  - random in $\{0, \ldots, q_\nu - 1\}$ with high probability if $n_i' \neq \nu$
- A hit will be identified as soon as a collision of DL will occur for some q_ν:

$$\mathrm{DL}(\mu_i, s_i', p_\nu, q_\nu) = \mathrm{DL}(\mu_j, s_j', p_\nu, q_\nu) \implies n_i' = n_j' = \nu$$

(see the paper for a discussion on false positive occurence probability)

- The number of required fault is $\mathcal{O}(\sqrt{\frac{t}{\alpha}|S|})$.

($t = \#$ of hits $\quad$ and $\quad \alpha \cdot |S| = \#$ of markers)

The *full consistency exploitation* variant

The *full consistency exploitation* variant

- The *full consistency exploitation* variant needs a dictionary S of possible faulty moduli.

The *full consistency exploitation* variant

- The *full consistency exploitation* variant needs a dictionary S of possible faulty moduli.
- The principle is to check some *intra*-signature and *inter*-signature consistencies.

The *full consistency exploitation* variant

- The *full consistency exploitation* variant needs a dictionary S of possible faulty moduli.
- The principle is to check some *intra*-signature and *inter*-signature consistencies.

Definition

For any $\nu \in S$ and any prime q, let $\Psi(\nu, q) = \{p \,:\, p \mid \nu \text{ and } q \mid p-1\}$

The *full consistency exploitation* variant

- The *full consistency exploitation* variant needs a dictionary S of possible faulty moduli.
- The principle is to check some *intra*-signature and *inter*-signature consistencies.

Definition

For any $\nu \in S$ and any prime q, let $\Psi(\nu, q) = \{p \,:\, p \mid \nu \text{ and } q \mid p - 1\}$

Intra-signature consistency

For any faulty signature (μ_i, s_i', n_i'), and for any prime q:

$$\left| \left\{ \mathrm{DL}(\mu_i, s_i', p, q) \,:\, p \in \Psi(n_i', q) \right\} \right| \leqslant 1$$

The *full consistency exploitation* variant

- The *full consistency exploitation* variant needs a dictionary S of possible faulty moduli.
- The principle is to check some *intra*-signature and *inter*-signature consistencies.

Definition

For any $\nu \in S$ and any prime q, let $\Psi(\nu, q) = \{p \,:\, p \mid \nu \text{ and } q \mid p-1\}$

Intra-signature consistency

For any faulty signature (μ_i, s_i', n_i'), and for any prime q:

$$\left|\left\{\,\mathrm{DL}(\mu_i, s_i', p, q) \;:\; p \in \Psi(n_i', q)\right\}\right| \leqslant 1$$

- Any candidate modulus ν for the signature (μ_i, s_i') must be excluded as soon as

$$\left|\left\{\,\mathrm{DL}(\mu_i, s_i', p, q) \;:\; p \in \Psi(\nu, q)\right\}\right| \geqslant 2 \quad \text{for some } q$$

The full consistency exploitation variant

Inter-signature consistency

For any faulty signatures $(\mu_{i_1}, s'_{i_1}, n'_{i_1})$ and $(\mu_{i_2}, s'_{i_2}, n'_{i_2})$, and any prime q:

$$\left| \{\, \mathrm{DL}(\mu_{i_1}, s'_{i_1}, p, q) : p \in \Psi(n'_{i_1}, q) \} \cup \{\, \mathrm{DL}(\mu_{i_2}, s'_{i_2}, p, q) : p \in \Psi(n'_{i_2}, q) \} \right| \leqslant 1$$

The full consistency exploitation variant

Inter-signature consistency

For any faulty signatures $(\mu_{i_1}, s'_{i_1}, n'_{i_1})$ and $(\mu_{i_2}, s'_{i_2}, n'_{i_2})$, and any prime q:

$$\left| \{\, \mathrm{DL}(\mu_{i_1}, s'_{i_1}, p, q) : p \in \Psi(n'_{i_1}, q)\} \cup \{\, \mathrm{DL}(\mu_{i_2}, s'_{i_2}, p, q) : p \in \Psi(n'_{i_2}, q)\} \right| \leqslant 1$$

- Any couple (ν_1, ν_2) of candidate moduli for the signatures (μ_{i_1}, s'_{i_1}) and (μ_{i_2}, s'_{i_2}) must be excluded (as not being simultaneously valid) if the consistency is not verified for some q.

The full consistency exploitation variant

Inter-signature consistency

For any faulty signatures $(\mu_{i_1}, s'_{i_1}, n'_{i_1})$ and $(\mu_{i_2}, s'_{i_2}, n'_{i_2})$, and any prime q:

$$\left|\{\, \mathrm{DL}(\mu_{i_1}, s'_{i_1}, p, q) : p \in \Psi(n'_{i_1}, q)\} \cup \{\, \mathrm{DL}(\mu_{i_2}, s'_{i_2}, p, q) : p \in \Psi(n'_{i_2}, q)\}\right| \leqslant 1$$

- Any couple (ν_1, ν_2) of candidate moduli for the signatures (μ_{i_1}, s'_{i_1}) and (μ_{i_2}, s'_{i_2}) must be excluded (as not being simultaneously valid) if the consistency is not verified for some q.
- The consistency check may be generalized to sets of candidate moduli with respect to sets of signatures.

The full consistency exploitation variant

- The paper describes an algorithm which, for a set of t signatures:

The full consistency exploitation variant

- The paper describes an algorithm which, for a set of t signatures:
  - Exhibits the list of all sets of t candidate moduli which are fully consistent with the signatures

The full consistency exploitation variant

- The paper describes an algorithm which, for a set of t signatures:
  - Exhibits the list of all sets of t candidate moduli which are fully consistent with the signatures
  - Assign a confidence index to each such set of candidate moduli

The full consistency exploitation variant

- The paper describes an algorithm which, for a set of t signatures:
  - Exhibits the list of all sets of t candidate moduli which are fully consistent with the signatures
  - Assign a confidence index to each such set of candidate moduli
- Combinatorial explosion prevents to use this algorithm when S is too large. (typically $|S| > 10,000$)

The full consistency exploitation variant

- The paper describes an algorithm which, for a set of t signatures:
  - Exhibits the list of all sets of t candidate moduli which are fully consistent with the signatures
  - Assign a confidence index to each such set of candidate moduli
- Combinatorial explosion prevents to use this algorithm when S is too large. (typically $|S| > 10,000$)
- This full consistency exploitation method allows to identify nearly t hits when considering t signatures.

The full consistency exploitation variant

- The paper describes an algorithm which, for a set of t signatures:
  - Exhibits the list of all sets of t candidate moduli which are fully consistent with the signatures
  - Assign a confidence index to each such set of candidate moduli
- Combinatorial explosion prevents to use this algorithm when S is too large. (typically $|S| > 10,000$)
- This full consistency exploitation method allows to identify nearly t hits when considering t signatures.

This method recovers the private exponent within only 10 to 20 faults

Outline

1. Introduction
   - Previous work
   - Our attack
   - The threat model
2. Description of the attack
   - Common Principle
   - The bias based variant
   - The collision based variant
   - The full consistency exploitation variant
3. Conclusion
   - Some interesting properties
   - Counter-measures
   - Open problems

Some interesting properties

- Our new attack present some notable properties:

Some interesting properties

- Our new attack present some notable properties:

The first fault attack on RSA ever published, which reveals the private exponent by only corrupting public elements of the key. 1 2 3

Some interesting properties

- Our new attack present some notable properties:

The first fault attack on RSA ever published, which reveals the private exponent by only corrupting public elements of the key. 1 2 3

The first fault attack on standard RSA ever published, which does not rely on any fault model, nor any implementation assumption. 1

Some interesting properties

- Our new attack present some notable properties:

The first fault attack on RSA ever published, which reveals the private exponent by only corrupting public elements of the key. 1 2 3

The first fault attack on standard RSA ever published, which does not rely on any fault model, nor any implementation assumption. 1

The fault attack on standard RSA, which reveals the private exponent with the smallest number of required faults. 3

Counter-measures

- The previously described fault attack on standard RSA is very efficient on non-protected implementations, but . . .

Counter-measures

- The previously described fault attack on standard RSA is very efficient on non-protected implementations, but ...
- ... many counter-measures exist that may prevent this attack:

Counter-measures

- The previously described fault attack on standard RSA is very efficient on non-protected implementations, but ...
- ... many counter-measures exist that may prevent this attack:
  - The integrity of the modulus may be ensured by a consistency check (checksum, ...)

Counter-measures

- The previously described fault attack on standard RSA is very efficient on non-protected implementations, but ...
- ... many counter-measures exist that may prevent this attack:
  - The integrity of the modulus may be ensured by a consistency check (checksum, ...)
  - The private exponent may be randomized

Counter-measures

- The previously described fault attack on standard RSA is very efficient on non-protected implementations, but ...
- ... many counter-measures exist that may prevent this attack:
  - The integrity of the modulus may be ensured by a consistency check (checksum, ...)
  - The private exponent may be randomized
  - The signature computation may be verified, and no signature is returned if the verification fails

Counter-measures

- The previously described fault attack on standard RSA is very efficient on non-protected implementations, but . . .
- . . . many counter-measures exist that may prevent this attack:
  - The integrity of the modulus may be ensured by a consistency check (checksum, . . .)
  - The private exponent may be randomized
  - The signature computation may be verified, and no signature is returned if the verification fails
  - . . .

Open problems

The fault attack presented here raises some open questions:

In standard mode, is it possible to recover the RSA private key by only corrupting the modulus when the private exponent is randomized ?

Open problems

The fault attack presented here raises some open questions:

In standard mode, is it possible to recover the RSA private key by only corrupting the modulus when the private exponent is randomized ?

Is it possible to adapt the attack in the case of a probabilistic padding with randomness recovery (e.g. RSA-PSS) ?

ERRATUM

In this paper:

ERRATUM

In this paper:

Abstract. It is well known that a malicious adversary can try to retrieve secret information by inducing a fault during cryptographic operations. Following the work of Seifert on fault inductions during RSA signature verification, we consider in this paper the signature counterpart. Our article introduces the first fault attack applied on RSA in standard mode. By only corrupting one public key element, one can recover the private exponent. Indeed, similarly to Seifert's attack, our attack is done by modifying the modulus.

ERRATUM

In this paper:

Abstract. It is well known that a malicious adversary can try to retrieve secret information by inducing a fault during cryptographic operations. Following the work of Seifert on fault inductions during RSA signature verification, we consider in this paper the signature counterpart. Our article introduces the first fault attack applied on RSA in standard mode. By only corrupting one public key element, one can recover the private exponent. Indeed, similarly to Seifert's attack, our attack is done by modifying the modulus.

Our paper DID NOT introduce the first fault attack on standard RSA !

ERRATUM

In this paper:

Abstract. It is well known that a malicious adversary can try to retrieve secret information by inducing a fault during cryptographic operations. Following the work of Seifert on fault inductions during RSA signature verification, we consider in this paper the signature counterpart. Our article introduces the first fault attack applied on RSA in standard mode. By only corrupting one public key element, one can recover the private exponent. Indeed, similarly to Seifert's attack, our attack is done by modifying the modulus.

Our paper DID NOT introduce the first fault attack on standard RSA !

ERRATUM

In this paper:

> **Abstract.** It is well known that a malicious adversary can try to retrieve secret information by inducing a fault during cryptographic operations. Following the work of Seifert on fault inductions during RSA signature verification, we consider in this paper the signature counterpart. Our article introduces the first fault attack applied on RSA in standard mode. By only corrupting one public key element, one can recover the private exponent. Indeed, similarly to Seifert's attack, our attack is done by modifying the modulus.

Our paper DID NOT introduce the first fault attack on standard RSA !

In the submission:

ERRATUM

In this paper:

> **Abstract.** It is well known that a malicious adversary can try to retrieve secret information by inducing a fault during cryptographic operations. Following the work of Seifert on fault inductions during RSA signature verification, we consider in this paper the signature counterpart. Our article introduces the first fault attack applied on RSA in standard mode. By only corrupting one public key element, one can recover the private exponent. Indeed, similarly to Seifert's attack, our attack is done by modifying the modulus.

Our paper DID NOT introduce the first fault attack on standard RSA !

In the submission:

> **Abstract.** Nowadays, it is well known that a malicious adversary can try to retrieve secret information by inducing a fault during cryptographic operations. Following the work of Seifert on fault induction during RSA signature verification, we consider in this paper the signature counterpart. Our article introduces the first fault attack that can be applied on RSA in standard mode in order to recover the private exponent by only corrupting one public key element. Indeed, similarly to Seifert's attack, our attack is managed by modifying the modulus by fault.

ERRATUM

In this paper:

Abstract. It is well known that a malicious adversary can try to retrieve secret information by inducing a fault during cryptographic operations. Following the work of Seifert on fault inductions during RSA signature verification, we consider in this paper the signature counterpart. Our article introduces the first fault attack applied on RSA in standard mode. By only corrupting one public key element, one can recover the private exponent. Indeed, similarly to Seifert's attack, our attack is done by modifying the modulus.

Our paper DID NOT introduce the first fault attack on standard RSA !

In the submission:

Abstract. Nowadays, it is well known that a malicious adversary can try to retrieve secret information by inducing a fault during cryptographic operations. Following the work of Seifert on fault induction during RSA signature verification, we consider in this paper the signature counterpart. Our article introduces the first fault attack that can be applied on RSA in standard mode in order to recover the private exponent by only corrupting one public key element. Indeed, similarly to Seifert's attack, our attack is managed by modifying the modulus by fault.

APOLOGIES !

Thank you for your attention !

Questions ?