Making RSA-PSS Provably Secure Against Non-Random Faults

Gilles Barthe (IMDEA) François Dupressoir (IMDEA)
Pierre-Alain Fouque (Univ. Rennes 1) Benjamin Grégoire (Inria)
Mehdi Tibouchi (NTT) Jean-Christophe Zapalowicz (Inria)

CHES 2014
RSA signatures

RSA signatures still widely used (Software & Embedded devices)

- Chinese Remainder Theorem (CRT) technique for efficiency

- Bellcore attack to avoid
  - Fault attack
  - Inject a fault during one of the half exponentiations
  - Message + faulted signature $\Rightarrow$ Secret key
RSA signatures

RSA signatures still widely used (Software & Embedded devices)

- Chinese Remainder Theorem (CRT) technique for efficiency

- Bellcore attack to avoid
  - Fault attack
  - Inject a fault during one of the half exponentiations
  - Message + faulted signature $\Rightarrow$ Secret key
RSA signatures

RSA signatures still widely used (Software & Embedded devices)

- Chinese Remainder Theorem (CRT) technique for efficiency

- Bellcore attack to avoid
  - Fault attack
  - Inject a fault during one of the half exponentiations
  - Message + faulted signature ⇒ Secret key
RSA signatures and fault attacks

Bellcore attack (or some extensions) efficient against

- no encoding
- deterministic encodings
- some randomized padding schemes (ISO 9796-2, EMV, ...)

But RSA-PSS (randomized padding scheme proposed by Bellare and Rogaway, Eurocrypt 96) is secure against random faults (proven by Coron and Mandal, Asiacrypt 2009)
RSA signatures and fault attacks

Bellcore attack (or some extensions) efficient against

- no encoding
- deterministic encodings
- some randomized padding schemes (ISO 9796-2, EMV, ...)

But RSA-PSS (randomized padding scheme proposed by Bellare and Rogaway, Eurocrypt 96) is secure against random faults (proven by Coron and Mandal, Asiacrypt 2009)
RSA-PSS

$h$ outputs bitstrings of length $k_h$ ($\{0, 1\}^* \times \{0, 1\}^{k_0} \rightarrow \{0, 1\}^{k_h}$)

$g$ (mixing $g_1$ and $g_2$) outputs bitstrings of length $k_g$ ($\{0, 1\}^{k_h} \rightarrow \{0, 1\}^{k_g}$)

$k_h + k_g + 1 = n$ and $k_0 < k_g$
RSA signatures and fault attacks

Different non-random fault models proposed in 2012 (Fouque, Guillermín, Leresteux, Tibouchi, Zapalowicz, CHES 2012)
- Require Montgomery multiplication to be used
- Apply to any padding function
RSA signatures and fault attacks

Different non-random fault models proposed in 2012 (Fouque, Guillermín, Leresteux, Tibouchi, Zapalowicz, CHES 2012)

- Require Montgomery multiplication to be used
- Apply to any padding function

Our Goal:

- To propose a countermeasure for RSA-PSS against a large class of non-random faults
  - Extend Coron and Mandal’s result to a stronger model
- To prove this countermeasure
- To formally verify the proof with the tool EasyCrypt
Fault model

Coron and Mandal’s fault model:

- Correct value modulo $p$
- Random value modulo $q$

Our fault model:

- Correct value modulo $p$
- Precise value modulo $q$ fixed by the adversary
Countermeasure

Simplest protection against fault attacks: to verify the signature

- $y' = S^e \mod N$
- If $y' = y$ then Return $S$ Else Return Error
Countermeasure

Simplest protection against fault attacks: to verify the signature

- \( y' = S^e \mod N \)
- \( \text{If } y' = y \text{ then Return } S \text{ Else Return Error} \)

But a test can be bypassed ⇒ Infective countermeasure

- A result released all the time
- Gibberish when faulty computations occur
Protected signing algorithm

1: function Sign(sk, pk, m)
2: \[ r \leftarrow \{0, 1\}^{k_0} \quad \triangleright \text{Start of PSS padding} \]
3: \[ y \leftarrow PSS(m, r) \]
Protected signing algorithm

1: function Sign(sk, pk, m)
2:     \[ r \leftarrow \{0, 1\}^{k_0} \] \hspace{1cm} \triangleright \text{Start of PSS padding}
3:     \[ y \leftarrow \text{PSS}(m, r) \] \hspace{1cm} \triangleright \text{Signature computation}
4:     \[ S_p \leftarrow y^{d_p} \mod p \]
5:     \[ S_q \leftarrow y^{d_q} \mod q \]
6:     \[ S \leftarrow (\alpha_p \cdot S_p + \alpha_q \cdot S_q) \mod N \] \hspace{1cm} \triangleright \alpha_p = q \cdot (q^{-1} \mod p)
Protected signing algorithm

1: function Sign(sk, pk, m)
2: \( r \leftarrow \{0, 1\}^{k_0} \) \( \triangleright \) Start of PSS padding
3: \( y \leftarrow PSS(m, r) \) \( \triangleright \) Signature computation
4: \( S_p \leftarrow y^{d_p} \mod p \)
5: \( S_q \leftarrow y^{d_q} \mod q \)
6: \( S \leftarrow (\alpha_p \cdot S_p + \alpha_q \cdot S_q) \mod N \) \( \triangleright \) \( \alpha_p = q \cdot (q^{-1} \mod p) \)
7: \( y' \leftarrow S^e \mod N \)
1: function Sign(sk, pk, m)
2: \( r \leftarrow \{0, 1\}^{k_0} \quad \triangleright \text{Start of PSS padding} \\
3: \ y \leftarrow \text{PSS}(m, r) \quad \triangleright \text{Signature computation} \\
4: \ S_p \leftarrow y^{d_p} \mod p \\
5: \ S_q \leftarrow y^{d_q} \mod q \\
6: \ S \leftarrow (\alpha_p \cdot S_p + \alpha_q \cdot S_q) \mod N \\
7: \ y' \leftarrow S^e \mod N \\
8: \ r' \leftarrow \{0, 1\}^\rho \\
9: \ S' \leftarrow S + r' \cdot (y - y') \mod N \\
10: \text{return } S'
Protected signing algorithm

1: function Sign(sk, pk, m)
2: \[ r \leftarrow \{0, 1\}^{k_0} \] \hspace{1cm} \triangleright \text{Start of PSS padding}
3: \[ y \leftarrow \text{PSS}(m, r) \] \hspace{1cm} \triangleright \text{Signature computation}
4: \[ S_p \leftarrow y^{d_p} \mod p \] \hspace{1cm} \triangleright \alpha_p = q \cdot (q^{-1} \mod p)
5: \[ S_q \leftarrow y^{d_q} \mod q \] \hspace{1cm} \triangleright \text{Infective countermeasure}
6: \[ S \leftarrow (\alpha_p \cdot S_p + \alpha_q \cdot S_q) \mod N \]
7: \[ y' \leftarrow S^e \mod N \]
8: \[ r' \leftarrow \{0, 1\}^{\rho} \]
9: \[ S' \leftarrow S + r' \cdot (y - y') \mod N \]
10: \text{return } S'

Size of \( r' \) (noted \( \rho \)) to define
UF-CMA Challenge
Challenger Adversary

$y \equiv x^e [N]$ 

OW-RSA Challenge
$y \equiv x^e[N]$

$e, N, y$

$x$

$m_i, \text{req}$

$S_i, h(m_i) \text{ or } g(m_i)$

$(m, S)$

**RSA-PSS reduction (Bellare and Rogaway)**
RSA-PSS reduction with random faults (Coron and Mandal)
$y \equiv x^e [N]$,

$y \equiv x^e [N]$,

$\text{Simulator}$,

$\text{Adversary}$,

$\text{RSA-PSS reduction with non-random faults}$,
Main Theorem

1: **game** *UF-CMA*
2: \((e, d, N) \leftarrow \mathcal{K}(\cdot)\)
3: \((m, s) \leftarrow \mathcal{A}^S,\mathcal{F},\mathcal{H},\mathcal{G}(e, N)\)
4: \(b \leftarrow \mathcal{V}(m, s)\)
5: \(\text{win} \leftarrow b \land (m, s) \notin Q^S\)
6: **return** \(\text{win}\)

1: **game** *OW-RSA*
2: \((e, d, N) \leftarrow \mathcal{K}(\cdot)\)
3: \(x^* \leftarrow [0..N]\)
4: \(y^* \leftarrow x^{e} \mod N\)
5: \(x \leftarrow \mathcal{I}(e, N, y^*)\)
6: **return** \(x = x^*\)

7 games later

Result

Given a CMA adversary \(\mathcal{A}\) against the faulty signature scheme \((\mathcal{K}, S, F, V)\), we build a one-way inverter \(\mathcal{I}\) such that

\[
\Pr[\text{UF-CMA} : \text{win}] \leq \Pr[\text{OW-RSA} : x = x^*] + \varepsilon
\]
Behind this theorem

Our proof follows the same path as Coron and Mandal’s proof

\[
S' = y^d \cdot \alpha^p + (a + r' \cdot (y - a^e)) \cdot \alpha^q \mod N |
\]

\[
S' = y^d + r' y \mod N |
\]

\[
\Rightarrow \quad \text{For large enough } N, \text{ it suffices to take } \rho \text{ slightly larger than a given } \varepsilon \text{ to obtain a statistical distance of } 2^{-\varepsilon}
\]
Behind this theorem

Our proof follows the same path as Coron and Mandal’s proof

1 lemma for the distribution of the faulted signatures

- Prove that the faulted signatures provide no information
- We want to remove the computation of $S'$ (which uses the secret key)
Behind this theorem

Our proof follows the same path as Coron and Mandal’s proof

1 lemma for the distribution of the faulted signatures

- Prove that the faulted signatures provide no information
- We want to remove the computation of $S'$ (which uses the secret key)

First lemma

\[
\begin{align*}
S' &= y^d \cdot \alpha_p + (a + r'(y - a^e)) \cdot \alpha_q \mod N \mid (y, r') \in [0, 2^{n-1}) \times [1, 2^\rho) \\
S' &= y^d + r'y \mod N \mid (y, r') \in [0, 2^{n-1}) \times [1, 2^\rho) \\
&\approx_s U_N
\end{align*}
\]

⇒ For large enough $N$, it suffices to take $\rho$ slightly larger than a given $\varepsilon$ to obtain a statistical distance of $2^{-\varepsilon}$
Behind this theorem

1 lemma for the probability of guessing $\omega$ given $S'$

⇒ represents a bad event in the initial proof of RSA-PSS

Second lemma

$$\Pr [\omega = \omega'|S'] \leq \frac{3}{2^k_h}$$

⇒ Requires $\rho$ larger than $k_h$
Behind this theorem

1 lemma for the probability of guessing $\omega$ given $S'$

$\Rightarrow$ represents a bad event in the initial proof of RSA-PSS

**Second lemma**

$$\Pr[\omega = \omega'|S'] \leq 3/2^{k_h}$$

$\Rightarrow$ Requires $\rho$ larger than $k_h$

However, counts are more difficult to perform because of our stronger fault model

$\Rightarrow$ Use of more complex mathematic tools

- Dirichlet characters sum
- Generalized Riemann Hypothesis for best result on $\rho$
  (can be replaced by Polya-Vinogradov inequality or Burgess bound, but $\rho$ increases)
Formal verification

Our proof = Game-based proof
⇒ can be handled by a computer-aided tool
⇒ EasyCrypt

Game = Program & Games transition = Approximate program equivalence

**EasyCrypt**: SMT based interactive proof engine

Already done:
- proof of standard crypto schemes (FDH, OAEP, ...)
- proof of a OAEP implementation with side channel
- Framework for building higher level tools
  - ZooCrypt: Automatic synthesis of padding based encryption schemes
  - Automatic verification of batch signature algorithms
  - Automatic verification of assumptions in the generic group model
  - Automatic synthesis of fault attacks
Without computer-aided tool:

- Lot of work for writing a proof
- Lot of work for trusting it
Without computer-aided tool:
  - Lot of work for writing a proof
  - Lot of work for trusting it

With computer-aided tool:
  - Lot of work for writing a proof
  - Easy to trust it
Without computer-aided tool:
- Lot of work for writing a proof
- Lot of work for trusting it

With computer-aided tool:
- Lot of work for writing a proof
- Easy to trust it

As an example, Coron and Mandal’s proof
- Correct result
- **But** glitch in the proof
Conclusion

Proven infective countermeasure for protecting RSA-PSS against a stronger fault model

Proof verified with EasyCrypt

Another step towards the provable security in the context of fault attacks
Thank you for your attention