Fault Attacks Against EMV Signatures

Jean-Sébastien Coron[1] David Naccache[2]
Mehdi Tibouchi[1,2]

[1]Université du Luxembourg

[2]École normale supérieure

CT-RSA 2010

Our Results in a Nutshell

- Simplify a former fault attack [CJKNP09] on ISO 9796-2 signatures, obtaining vastly improved efficiency.
- Simulate this new fault attack on parameters of typical size, recovering secret keys with a small number of faulty signatures.
- Show how the attack applies to EMV signature formats that where far beyond the reach of former cryptanalytic techniques.

Our Results in a Nutshell

- Simplify a former fault attack [CJKNP09] on ISO 9796-2 signatures, obtaining vastly improved efficiency.
- Simulate this new fault attack on parameters of typical size, recovering secret keys with a small number of faulty signatures.
- Show how the attack applies to EMV signature formats that where far beyond the reach of former cryptanalytic techniques.

Our Results in a Nutshell

- Simplify a former fault attack [CJKNP09] on ISO 9796-2 signatures, obtaining vastly improved efficiency.
- Simulate this new fault attack on parameters of typical size, recovering secret keys with a small number of faulty signatures.
- Show how the attack applies to EMV signature formats that where far beyond the reach of former cryptanalytic techniques.

Outline

Outline

Signing with RSA-CRT

In RSA-based signature schemes, a signer with modulus $N = pq$ and key pair (e, d) signs a message m by computing:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma_q = \mu(m)^d \bmod q$
3. $\sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N$

where μ is the encoding function of the scheme.

The Chinese Remainder Theorem offers a welcome 4-fold speed-up in (often costly) signature generation.

Signing with RSA-CRT

In RSA-based signature schemes, a signer with modulus $N = pq$ and key pair (e, d) signs a message m by computing:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma_q = \mu(m)^d \bmod q$
3. $\sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N$

where μ is the encoding function of the scheme.

The Chinese Remainder Theorem offers a welcome 4-fold speed-up in (often costly) signature generation.

Signing with RSA-CRT

In RSA-based signature schemes, a signer with modulus $N = pq$ and key pair (e, d) signs a message m by computing:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma_q = \mu(m)^d \bmod q$
3. $\sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N$

where μ is the encoding function of the scheme.

The Chinese Remainder Theorem offers a welcome 4-fold speed-up in (often costly) signature generation.

Signing with RSA-CRT

In RSA-based signature schemes, a signer with modulus $N = pq$ and key pair (e, d) signs a message m by computing:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma_q = \mu(m)^d \bmod q$
3. $\sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N$

where μ is the encoding function of the scheme.

The Chinese Remainder Theorem offers a welcome 4-fold speed-up in (often costly) signature generation.

Signing with RSA-CRT

In RSA-based signature schemes, a signer with modulus $N = pq$ and key pair (e, d) signs a message m by computing:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma_q = \mu(m)^d \bmod q$
3. $\sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N$

where μ is the encoding function of the scheme.

The Chinese Remainder Theorem offers a welcome 4-fold speed-up in (often costly) signature generation.

The Bellcore Fault Attack

The problem with CRT: fault attacks. A fault in signature generation makes it possible to recover the secret key:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma'_q \neq \mu(m)^d \bmod q$
3. $\sigma' = \text{CRT}(\sigma_p, \sigma'_q) \bmod N$

Then σ'^e is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd(\sigma'^e - \mu(m), N)$$

This attack applies to:

The Bellcore Fault Attack

The problem with CRT: fault attacks. A fault in signature generation makes it possible to recover the secret key:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma'_q \neq \mu(m)^d \bmod q$
3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$

Then σ'^e is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd(\sigma'^e - \mu(m), N)$$

This attack applies to:

- any deterministic padding, e.g. FDH, $\sigma = H(m)^d \bmod N$
- any probabilistic padding with public randomizer, e.g. PFDH, $\sigma = (r, H(m\|r)^d \bmod N)$

The Bellcore Fault Attack

The problem with CRT: fault attacks. A fault in signature generation makes it possible to recover the secret key:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma'_q \neq \mu(m)^d \bmod q$
3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$

Then σ'^e is $\mu(m)$ mod p but not mod q, so the attacker can then factor N:

$$p = \gcd(\sigma'^e - \mu(m), N)$$

This attack applies to:

- any deterministic padding; e.g. FDH, $\sigma = H(m)^d \bmod N$
- any probabilistic padding with public randomizer; e.g. PFDH, $\sigma = (r, H(m\|r)^d \bmod N)$

The Bellcore Fault Attack

The problem with CRT: fault attacks. A fault in signature generation makes it possible to recover the secret key:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma'_q \neq \mu(m)^d \bmod q$
3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$

Then σ'^e is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd(\sigma'^e - \mu(m), N)$$

This attack applies to:

- any deterministic padding, e.g. FDH, $\sigma = H(m)^d \bmod N$
- any probabilistic padding with public randomizer, e.g. PFDH, $\sigma = (r, H(m\|r)^d \bmod N)$

The Bellcore Fault Attack

The problem with CRT: fault attacks. A fault in signature generation makes it possible to recover the secret key:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma'_q \neq \mu(m)^d \bmod q$
3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$

Then ${\sigma'}^e$ is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd({\sigma'}^e - \mu(m), N)$$

This attack applies to:

- any deterministic padding, e.g. FDH, $\sigma = H(m)^d \bmod N$
- any probabilistic padding with public randomizer, e.g. PFDH, $\sigma = (r, H(m\|r)^d \bmod N)$

The Bellcore Fault Attack

The problem with CRT: fault attacks. A fault in signature generation makes it possible to recover the secret key:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma'_q \neq \mu(m)^d \bmod q$
3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$

Then ${\sigma'}^e$ is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd({\sigma'}^e - \mu(m), N)$$

This attack applies to:

- any deterministic padding: e.g. FDH, $\sigma = H(m)^d \bmod N$
- any probabilistic padding with public randomizer: e.g. PFDH, $\sigma = (r, H(m\|r)^d \bmod N)$

The Bellcore Fault Attack

The problem with CRT: fault attacks. A fault in signature generation makes it possible to recover the secret key:

1. $\sigma_p = \mu(m)^d \bmod p$
2. $\sigma'_q \neq \mu(m)^d \bmod q$
3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$

Then ${\sigma'}^e$ is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd({\sigma'}^e - \mu(m), N)$$

This attack applies to:

- any deterministic padding; e.g. FDH, $\sigma = H(m)^d \bmod N$
- any probabilistic padding with public randomizer; e.g. PFDH, $\sigma = \big(r, H(m\|r)^d \bmod N\big)$

Countermeasures and Extensions

The Bellcore attacks does not apply when only a part of the signed encoding is known to the attacker. Examples:

- $\sigma = (m \| r)^d \bmod N$, where r is a large enough random nonce unknown to the attacker.
- $\sigma = \left(\omega \| G_1(\omega) \oplus r \| G_2(\omega)\right)^d \bmod N$ where r is a random nonce and $\omega = H(m \| r)$. This is PSS.

The attacker doesn't know r, cannot compute $\sigma^e - \mu(m)$ to factor N: the Bellcore attack is thwarted.

In fact, PSS was shown to be secure against fault attacks [CM09].

However, variants of $(m \| r)^d$ actually used in practice, such as ISO 9796-2, are vulnerable to generalizations of the Bellcore attack.

Countermeasures and Extensions

The Bellcore attacks does not apply when only a part of the signed encoding is known to the attacker. Examples:

- $\sigma = (m\|r)^d \bmod N$, where r is a large enough random nonce unknown to the attacker.
- $\sigma = \big(\omega\|G_1(\omega) \oplus r\|G_2(\omega)\big)^d \bmod N$, where r is a random nonce and $\omega = H(m\|r)$. This is PSS.

The attacker doesn't know r, cannot compute $\sigma' - \mu(m)$ to factor N: the Bellcore attack is thwarted.

In fact, PSS was shown to be secure against fault attacks [CM09].

However, variants of $(m\|r)^d$ actually used in practice, such as ISO 9796-2, are vulnerable to generalizations of the Bellcore attack.

Countermeasures and Extensions

The Bellcore attacks does not apply when only a part of the signed encoding is known to the attacker. Examples:

- $\sigma = (m \| r)^d \bmod N$, where r is a large enough random nonce unknown to the attacker.
- $\sigma = \big(\omega \| G_1(\omega) \oplus r \| G_2(\omega)\big)^d \bmod N$, where r is a random nonce and $\omega = H(m\|r)$. This is PSS.

The attacker doesn't know r, cannot compute $\sigma' - \mu(m)$ to factor N: the Bellcore attack is thwarted.

In fact, PSS was shown to be secure against fault attacks [CM09].

However, variants of $(m\|r)^d$ actually used in practice, such as ISO 9796-2, are vulnerable to generalizations of the Bellcore attack.

Countermeasures and Extensions

The Bellcore attacks does not apply when only a part of the signed encoding is known to the attacker. Examples:

- $\sigma = (m \| r)^d \bmod N$, where r is a large enough random nonce unknown to the attacker.
- $\sigma = \big(\omega \| G_1(\omega) \oplus r \| G_2(\omega)\big)^d \bmod N$, where r is a random nonce and $\omega = H(m \| r)$. This is PSS.

The attacker doesn't know r, cannot compute $\sigma' - \mu(m)$ to factor N: the Bellcore attack is thwarted.

In fact, PSS was shown to be secure against fault attacks [CM09].

However, variants of $(m \| r)^d$ actually used in practice, such as ISO 9796-2, are vulnerable to generalizations of the Bellcore attack.

Countermeasures and Extensions

The Bellcore attacks does not apply when only a part of the signed encoding is known to the attacker. Examples:

- $\sigma = (m \| r)^d \bmod N$, where r is a large enough random nonce unknown to the attacker.
- $\sigma = \big(\omega \| G_1(\omega) \oplus r \| G_2(\omega)\big)^d \bmod N$, where r is a random nonce and $\omega = H(m \| r)$. This is PSS.

The attacker doesn't know r, cannot compute $\sigma' - \mu(m)$ to factor N: the Bellcore attack is thwarted.

In fact, PSS was shown to be secure against fault attacks [CM09]. However, variants of $(m \| r)^d$ actually used in practice, such as ISO 9796-2, are vulnerable to generalizations of the Bellcore attack.

Countermeasures and Extensions

The Bellcore attacks does not apply when only a part of the signed encoding is known to the attacker. Examples:

- $\sigma = (m\|r)^d \bmod N$, where r is a large enough random nonce unknown to the attacker.
- $\sigma = \big(\omega\|G_1(\omega) \oplus r\|G_2(\omega)\big)^d \bmod N$, where r is a random nonce and $\omega = H(m\|r)$. This is PSS.

The attacker doesn't know r, cannot compute $\sigma' - \mu(m)$ to factor N: the Bellcore attack is thwarted.
In fact, PSS was shown to be secure against fault attacks [CM09].
However, variants of $(m\|r)^d$ actually used in practice, such as ISO 9796-2, are vulnerable to generalizations of the Bellcore attack.

Outline

ISO 9796-2

- ISO/IEC 9796-2 defines an encoding with partial recovery: messages m are divided as $m[1]\|m[2]$, and only $m[2]$ is transmitted; $m[1]$ is recovered during signature verification. More precisely:

$$\mu(m) = \texttt{6A}_{16}\|m[1]\|H(m)\|\texttt{BC}_{16}$$

- In cases of interest (*e.g.* EMV signatures), we can write:

$$m[1] = \alpha\|r\|\alpha' \qquad m[2] = \text{DATA}$$

where α, α' are known bit patterns, and r is unknown.

- The encoded message is thus:

$$\mu(m) = \texttt{6A}_{16}\|\alpha\|r\|\alpha'\|H(m)\|\texttt{BC}_{16}$$

where the highlighted parts are unknown.

ISO 9796-2

- ISO/IEC 9796-2 defines an encoding with partial recovery: messages m are divided as $m[1] \| m[2]$, and only $m[2]$ is transmitted; $m[1]$ is recovered during signature verification. More precisely:

$$\mu(m) = \texttt{6A}_{16} \| m[1] \| H(m) \| \texttt{BC}_{16}$$

- In cases of interest (*e.g.* EMV signatures), we can write:

$$m[1] = \alpha \| r \| \alpha' \qquad m[2] = \text{DATA}$$

where α, α' are known bit patterns, and r is unknown.

- The encoded message is thus:

$$\mu(m) = \texttt{6A}_{16} \| \alpha \| r \| \alpha' \| H(m) \| \texttt{BC}_{16}$$

where the highlighted parts are unknown.

ISO 9796-2

- ISO/IEC 9796-2 defines an encoding with partial recovery: messages m are divided as $m[1]\|m[2]$, and only $m[2]$ is transmitted; $m[1]$ is recovered during signature verification. More precisely:

$$\mu(m) = \texttt{6A}_{16}\|m[1]\|H(m)\|\texttt{BC}_{16}$$

- In cases of interest (*e.g.* EMV signatures), we can write:

$$m[1] = \alpha\|r\|\alpha' \qquad m[2] = \textsc{data}$$

  where α, α' are known bit patterns, and r is unknown.
- The encoded message is thus:

$$\mu(m) = \texttt{6A}_{16}\|\alpha\|r\|\alpha'\|H(m)\|\texttt{BC}_{16}$$

  where the highlighted parts are unknown.

The CJKNP Attack

Due to unknown message parts, the Bellcore attack does not apply to ISO 9796-2 signatures. However, Coron *et al.* [CJKNP09] propose the following fault attack.

1. Write the encoded message as:

$$\mu(m) = t + r \cdot 2^{n_r} + H(m) \cdot 2^8$$

2. A faulty signature σ' yields an equation of the form

$$A + B \cdot r + C \cdot H(m) = 0 \pmod{p}$$

with $A = t - \sigma'^e$, $B = 2^{n_r}$, $C = 2^8$.

3. $(x_0, y_0) = (r, H(m))$ is a small root mod p of the bivariate polynomial $A + Bx + Cy$.
4. If (x_0, y_0) is small enough, Coppersmith-like techniques by Herrmann and May [HM08] can recover it.
5. Then, $\mu(m)$ can be computed to find $p = \gcd(\sigma'^e - \mu(m), N)$.

The CJKNP Attack

Due to unknown message parts, the Bellcore attack does not apply to ISO 9796-2 signatures. However, Coron *et al.* [CJKNP09] propose the following fault attack.

1. Write the encoded message as:

$$\mu(m) = t + r \cdot 2^{n_r} + H(m) \cdot 2^8$$

2. A faulty signature σ' yields an equation of the form:

$$A + B \cdot r + C \cdot H(m) \equiv 0 \pmod{p}$$

   with $A = t - {\sigma'}^e$, $B = 2^{n_r}$, $C = 2^8$.

3. $(x_0, y_0) = (r, H(m))$ is a small root mod p of the bivariate polynomial $A + Bx + Cy$.
4. If (x_0, y_0) is small enough, Coppersmith-like techniques by Hermann and May [HM08] can recover it.
5. Then, $\mu(m)$ can be computed to find $p = \gcd(\sigma'^e - \mu(m), N)$.

The CJKNP Attack

Due to unknown message parts, the Bellcore attack does not apply to ISO 9796-2 signatures. However, Coron *et al.* [CJKNP09] propose the following fault attack.

1. Write the encoded message as:

$$\mu(m) = t + r \cdot 2^{n_r} + H(m) \cdot 2^8$$

2. A faulty signature σ' yields an equation of the form:

$$A + B \cdot r + C \cdot H(m) \equiv 0 \pmod{p}$$

   with $A = t - \sigma'^e$, $B = 2^{n_r}$, $C = 2^8$.
3. $(x_0, y_0) = (r, H(m))$ is a small root mod p of the bivariate polynomial $A + Bx + Cy$.
4. If (x_0, y_0) is small enough, Coppersmith-like techniques by Hermann and May [HM08] can recover it.
5. Then, $\mu(m)$ can be computed to find $p = \gcd(\sigma'^e - \mu(m), N)$.

The CJKNP Attack

Due to unknown message parts, the Bellcore attack does not apply to ISO 9796-2 signatures. However, Coron *et al.* [CJKNP09] propose the following fault attack.

1. Write the encoded message as:

$$\mu(m) = t + r \cdot 2^{n_r} + H(m) \cdot 2^8$$

2. A faulty signature σ' yields an equation of the form:

$$A + B \cdot r + C \cdot H(m) \equiv 0 \pmod{p}$$

with $A = t - \sigma'^e$, $B = 2^{n_r}$, $C = 2^8$.

3. $(x_0, y_0) = (r, H(m))$ is a small root mod p of the bivariate polynomial $A + Bx + Cy$.
4. If (x_0, y_0) is small enough, Coppersmith-like techniques by Hermann and May [HM08] can recover it.
5. Then, $\mu(m)$ can be computed to find $p = \gcd(\sigma'^e - \mu(m), N)$.

The CJKNP Attack

Due to unknown message parts, the Bellcore attack does not apply to ISO 9796-2 signatures. However, Coron *et al.* [CJKNP09] propose the following fault attack.

1. Write the encoded message as:

$$\mu(m) = t + r \cdot 2^{n_r} + H(m) \cdot 2^8$$

2. A faulty signature σ' yields an equation of the form:

$$A + B \cdot r + C \cdot H(m) \equiv 0 \pmod p$$

   with $A = t - \sigma'^e$, $B = 2^{n_r}$, $C = 2^8$.

3. $(x_0, y_0) = (r, H(m))$ is a small root mod p of the bivariate polynomial $A + Bx + Cy$.
4. If (x_0, y_0) is small enough, Coppersmith-like techniques by Hermann and May [HM08] can recover it.
5. Then, $\mu(m)$ can be computed to find $p = \gcd(\sigma'^e - \mu(m), N)$.

The CJKNP Attack

Due to unknown message parts, the Bellcore attack does not apply to ISO 9796-2 signatures. However, Coron *et al.* [CJKNP09] propose the following fault attack.

1. Write the encoded message as:

$$\mu(m) = t + r \cdot 2^{n_r} + H(m) \cdot 2^8$$

2. A faulty signature σ' yields an equation of the form:

$$A + B \cdot r + C \cdot H(m) \equiv 0 \pmod{p}$$

with $A = t - \sigma'^e$, $B = 2^{n_r}$, $C = 2^8$.

3. $(x_0, y_0) = (r, H(m))$ is a small root mod p of the bivariate polynomial $A + Bx + Cy$.
4. If (x_0, y_0) is small enough, Coppersmith-like techniques by Hermann and May [HM08] can recover it.
5. Then, $\mu(m)$ can be computed to find $p = \gcd(\sigma'^e - \mu(m), N)$.

The CJKNP Attack

Due to unknown message parts, the Bellcore attack does not apply to ISO 9796-2 signatures. However, Coron *et al.* [CJKNP09] propose the following fault attack.

1. Write the encoded message as:

$$\mu(m) = t + r \cdot 2^{n_r} + H(m) \cdot 2^8$$

2. A faulty signature σ' yields an equation of the form:

$$A + B \cdot r + C \cdot H(m) \equiv 0 \pmod{p}$$

   with $A = t - {\sigma'}^e$, $B = 2^{n_r}$, $C = 2^8$.
3. $(x_0, y_0) = (r, H(m))$ is a small root mod p of the bivariate polynomial $A + Bx + Cy$.
4. If (x_0, y_0) is small enough, Coppersmith-like techniques by Hermann and May [HM08] can recover it.
5. Then, $\mu(m)$ can be computed to find $p = \gcd({\sigma'}^e - \mu(m), N)$.

Limitations of the CJKNP Attack

- Severe size constraint on $r, H(m)$: the combined bit length of unknown message parts (UMP) must be $< 0.207 \cdot n$. For a 160-bit digest and 1024-bit modulus, r can be at most 52 bits.
- As usual with multivariate Coppersmith techniques, the Hermann-May algorithm is only heuristic, performs poorly or fails when UMP size gets close to the limit.
- To handle larger UMPs, up to $0.5 \cdot n$ in theory, one can take advantage of multiple faults.
- However, complexity grows exponentially with the number of faulty signatures. Going beyond about $0.23 \cdot n$ is totally unfeasible.

Limitations of the CJKNP Attack

- Severe size constraint on $r, H(m)$: the combined bit length of unknown message parts (UMP) must be $< 0.207 \cdot n$. For a 160-bit digest and 1024-bit modulus, r can be at most 52 bits.
- As usual with multivariate Coppersmith techniques, the Hermann-May algorithm is only heuristic, performs poorly or fails when UMP size gets close to the limit.
- To handle larger UMPs, up to $0.5 \cdot n$ in theory, one can take advantage of multiple faults.
- However, complexity grows exponentially with the number of faulty signatures. Going beyond about $0.23 \cdot n$ is totally unfeasible.

Limitations of the CJKNP Attack

- Severe size constraint on $r, H(m)$: the combined bit length of unknown message parts (UMP) must be $< 0.207 \cdot n$. For a 160-bit digest and 1024-bit modulus, r can be at most 52 bits.
- As usual with multivariate Coppersmith techniques, the Hermann-May algorithm is only heuristic, performs poorly or fails when UMP size gets close to the limit.
- To handle larger UMPs, up to $0.5 \cdot n$ in theory, one can take advantage of multiple faults.
- However, complexity grows exponentially with the number of faulty signatures. Going beyond about $0.23 \cdot n$ is totally unfeasible.

Limitations of the CJKNP Attack

- Severe size constraint on $r, H(m)$: the combined bit length of unknown message parts (UMP) must be $< 0.207 \cdot n$. For a 160-bit digest and 1024-bit modulus, r can be at most 52 bits.
- As usual with multivariate Coppersmith techniques, the Hermann-May algorithm is only heuristic, performs poorly or fails when UMP size gets close to the limit.
- To handle larger UMPs, up to $0.5 \cdot n$ in theory, one can take advantage of multiple faults.
- However, complexity grows exponentially with the number of faulty signatures. Going beyond about $0.23 \cdot n$ is totally unfeasible.

Outline

Pushing Beyond CJKNP

Our paper introduces a new multiple fault attack on ISO 9796-2 lifting most limitations of the CJKNP attack.

- Simpler and purely linear: doesn't suffer from algebraic independence problems of multivariate Coppersmith techniques.
- Scales well with the number of faults: easy to handle UMPs almost as large as the theoretical maximum of $0.5 \cdot n$.
- Applicable to many EMV signature formats well beyond the reach of CJKNP attacks.

Pushing Beyond CJKNP

Our paper introduces a new multiple fault attack on ISO 9796-2 lifting most limitations of the CJKNP attack.

- Simpler and purely linear: doesn't suffer from algebraic independence problems of multivariate Coppersmith techniques.
- Scales well with the number of faults: easy to handle UMPs almost as large as the theoretical maximum of $0.5 \cdot n$.
- Applicable to many EMV signature formats well beyond the reach of CJKNP attacks.

Pushing Beyond CJKNP

Our paper introduces a new multiple fault attack on ISO 9796-2 lifting most limitations of the CJKNP attack.

- Simpler and purely linear: doesn't suffer from algebraic independence problems of multivariate Coppersmith techniques.
- Scales well with the number of faults: easy to handle UMPs almost as large as the theoretical maximum of $0.5 \cdot n$.
- Applicable to many EMV signature formats well beyond the reach of CJKNP attacks.

Pushing Beyond CJKNP

Our paper introduces a new multiple fault attack on ISO 9796-2 lifting most limitations of the CJKNP attack.

- Simpler and purely linear: doesn't suffer from algebraic independence problems of multivariate Coppersmith techniques.
- Scales well with the number of faults: easy to handle UMPs almost as large as the theoretical maximum of $0.5 \cdot n$.
- Applicable to many EMV signature formats well beyond the reach of CJKNP attacks.

Rundown of Our Attack

Recall that each faulty ISO 9796-2 signature σ'_i gives an equation $A_i + Bx_i + Cy_i \equiv 0 \pmod{p}$, with $(x_i, y_i) = (r_i, H(m_i))$. Dividing by B, we get affine relations:

$$a_i + x_i + cy_i \equiv 0 \pmod{p} \qquad (*)$$

Given ℓ faulty signatures, our attack proceeds as follows:

1. Linearize: find vectors $\mathbf{u_j} = (u_{1j}, \dots, u_{\ell j})$ such that $\mathbf{u_j} \cdot \mathbf{a} \equiv 0 \pmod{N}$. Use them to cancel constant terms between the relations $(*)$.
2. Orthogonalize: if the vectors as small enough, each $\mathbf{u_j}$ is orthogonal to $\mathbf{x}$ and $\mathbf{y}$. Deduce a $\mathbb{Z}$-lattice containing $\mathbf{x}$ and $\mathbf{y}$.
3. Factor: find a vector $\mathbf{v}$ orthogonal to both $\mathbf{x}$ and $\mathbf{y}$ mod N, but not to $\mathbf{a}$. Then $p = \gcd(\mathbf{v} \cdot \mathbf{a}, N)$.

Rundown of Our Attack

Recall that each faulty ISO 9796-2 signature σ_i' gives an equation $A_i + Bx_i + Cy_i \equiv 0 \pmod p$, with $(x_i, y_i) = (r_i, H(m_i))$. Dividing by B, we get affine relations:

$$a_i + x_i + cy_i \equiv 0 \pmod p \qquad (*)$$

Given ℓ faulty signatures, our attack proceeds as follows:

1. Linearize: find vectors $\mathbf{u_j} = (u_{1j}, \ldots, u_{\ell j})$ such that $\mathbf{u_j} \cdot \mathbf{a} \equiv 0 \pmod N$. Use them to cancel constant terms between the relations $(*)$.
2. Orthogonalize: if the vectors as small enough, each $\mathbf{u_j}$ is orthogonal to $\mathbf{x}$ and $\mathbf{y}$. Deduce a $\mathbb{Z}$-lattice containing $\mathbf{x}$ and $\mathbf{y}$.
3. Factor: find a vector $\mathbf{v}$ orthogonal to both $\mathbf{x}$ and $\mathbf{y}$ mod N, but not to $\mathbf{a}$. Then $p = \gcd(\mathbf{v} \cdot \mathbf{a}, N)$.

Rundown of Our Attack

Recall that each faulty ISO 9796-2 signature σ'_i gives an equation $A_i + Bx_i + Cy_i \equiv 0 \pmod{p}$, with $(x_i, y_i) = (r_i, H(m_i))$. Dividing by B, we get affine relations:

$$a_i + x_i + cy_i \equiv 0 \pmod{p} \qquad (*)$$

Given ℓ faulty signatures, our attack proceeds as follows:

1. Linearize: find vectors $\mathbf{u_j} = (u_{1j}, \dots, u_{\ell j})$ such that $\mathbf{u_j} \cdot \mathbf{a} \equiv 0 \pmod{N}$. Use them to cancel constant terms between the relations $(*)$.
2. Orthogonalize: if the vectors as small enough, each $\mathbf{u_j}$ is orthogonal to $\mathbf{x}$ and $\mathbf{y}$. Deduce a $\mathbb{Z}$-lattice containing $\mathbf{x}$ and $\mathbf{y}$.
3. Factor: find a vector $\mathbf{v}$ orthogonal to both $\mathbf{x}$ and $\mathbf{y}$ mod N, but not to $\mathbf{a}$. Then $p = \gcd(\mathbf{v} \cdot \mathbf{a}, N)$.

Rundown of Our Attack

Recall that each faulty ISO 9796-2 signature σ'_i gives an equation $A_i + Bx_i + Cy_i \equiv 0 \pmod{p}$, with $(x_i, y_i) = (r_i, H(m_i))$. Dividing by B, we get affine relations:

$$a_i + x_i + cy_i \equiv 0 \pmod{p} \qquad (*)$$

Given ℓ faulty signatures, our attack proceeds as follows:

1. Linearize: find vectors $\mathbf{u_j} = (u_{1j}, \ldots, u_{\ell j})$ such that $\mathbf{u_j} \cdot \mathbf{a} \equiv 0 \pmod{N}$. Use them to cancel constant terms between the relations $(*)$.
2. Orthogonalize: if the vectors as small enough, each $\mathbf{u_j}$ is orthogonal to $\mathbf{x}$ and $\mathbf{y}$. Deduce a $\mathbb{Z}$-lattice containing $\mathbf{x}$ and $\mathbf{y}$.
3. Factor: find a vector $\mathbf{v}$ orthogonal to both $\mathbf{x}$ and $\mathbf{y}$ mod N, but not to $\mathbf{a}$. Then $p = \gcd(\mathbf{v} \cdot \mathbf{a}, N)$.

More details

All three steps involve standard orthogonal lattice techniques, as used by Nguyen and Stern in the late 90's.

1. Linearization: to find short vectors $\mathbf{u_j}$ such that $\mathbf{u_j} \cdot \mathbf{a} \equiv 0 \pmod{N}$, apply LLL-reduction. Then, letting $\alpha_j = \mathbf{u_j} \cdot \mathbf{x}$, $\beta_j = \mathbf{u_j} \cdot \mathbf{y}$, we get $\alpha_j + c\beta_j \equiv 0 \pmod{p}$.
2. Orthogonalization: (α_j, β_j) is a short vector in a lattice $L(c, p) \subset \mathbb{Z}^2$. If it is short enough, it must be $(0, 0)$, hence the $\mathbf{u_j}$ are all orthogonal to $\mathbf{x}$, $\mathbf{y}$ over $\mathbb{Z}$.
   Say we have $\ell - 2$ of those $\mathbf{u_j}$: then their orthogonal lattice in $\mathbb{Z}^\ell$ is 2-dimensional and contains $\mathbf{x}$, $\mathbf{y}$. Find a basis of this lattice using LLL.
3. Factoring: finding a vector $\mathbf{v}$ orthogonal to both $\mathbf{x}$ and $\mathbf{y}$ mod N is then a simple matter. It will not be orthogonal to $\mathbf{a}$ with overwhelming probability.

More details

All three steps involve standard orthogonal lattice techniques, as used by Nguyen and Stern in the late 90's.

1. Linearization: to find short vectors $\mathbf{u_j}$ such that $\mathbf{u_j} \cdot \mathbf{a} \equiv 0 \pmod{N}$, apply LLL-reduction to the lattice:

$$\begin{pmatrix} \kappa a_1 & \cdots & \kappa a_\ell & N \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{pmatrix}$$

   for some large enough constant κ. Then, letting $\alpha_j = \mathbf{u_j} \cdot \mathbf{x}$, $\beta_j = \mathbf{u_j} \cdot \mathbf{y}$, we get $\alpha_j + c\beta_j \equiv 0 \pmod{p}$.
2. Orthogonalization: (α_j, β_j) is a short vector in a lattice $L(c,p) \subset \mathbb{Z}^2$. If it is short enough, it must be $(0,0)$, hence the $\mathbf{u_j}$ are all orthogonal to $\mathbf{x}$, $\mathbf{y}$ over $\mathbb{Z}$.

   Say we have $\ell - 2$ of those $\mathbf{u_j}$, then their orthogonal lattice in $\mathbb{Z}^\ell$ is 2-dimensional and contains $\mathbf{x}, \mathbf{y}$. Find a basis of this

More details

All three steps involve standard orthogonal lattice techniques, as used by Nguyen and Stern in the late 90's.

1. Linearization: to find short vectors $\mathbf{u_j}$ such that $\mathbf{u_j} \cdot \mathbf{a} \equiv 0 \pmod{N}$, apply LLL-reduction. Then, letting $\alpha_j = \mathbf{u_j} \cdot \mathbf{x}$, $\beta_j = \mathbf{u_j} \cdot \mathbf{y}$, we get $\alpha_j + c\beta_j \equiv 0 \pmod{p}$.
2. Orthogonalization: (α_j, β_j) is a short vector in a lattice $L(c, p) \subset \mathbb{Z}^2$. If it is short enough, it must be $(0, 0)$, hence the $\mathbf{u_j}$ are all orthogonal to $\mathbf{x}$, $\mathbf{y}$ over $\mathbb{Z}$.

   Say we have $\ell - 2$ of those $\mathbf{u_j}$: then their orthogonal lattice in $\mathbb{Z}^\ell$ is 2-dimensional and contains $\mathbf{x}$, $\mathbf{y}$. Find a basis of this lattice using LLL.
3. Factoring: finding a vector $\mathbf{v}$ orthogonal to both $\mathbf{x}$ and $\mathbf{y}$ mod N is then a simple matter. It will not be orthogonal to $\mathbf{a}$ with overwhelming probability.

More details

All three steps involve standard orthogonal lattice techniques, as used by Nguyen and Stern in the late 90's.

1. Linearization: to find short vectors $\mathbf{u_j}$ such that $\mathbf{u_j} \cdot \mathbf{a} \equiv 0 \pmod{N}$, apply LLL-reduction. Then, letting $\alpha_j = \mathbf{u_j} \cdot \mathbf{x}$, $\beta_j = \mathbf{u_j} \cdot \mathbf{y}$, we get $\alpha_j + c\beta_j \equiv 0 \pmod{p}$.
2. Orthogonalization: (α_j, β_j) is a short vector in a lattice $L(c, p) \subset \mathbb{Z}^2$. If it is short enough, it must be $(0, 0)$, hence the $\mathbf{u_j}$ are all orthogonal to $\mathbf{x}$, $\mathbf{y}$ over $\mathbb{Z}$.

   Say we have $\ell - 2$ of those $\mathbf{u_j}$: then their orthogonal lattice in $\mathbb{Z}^\ell$ is 2-dimensional and contains $\mathbf{x}, \mathbf{y}$. Find a basis of this lattice using LLL.
3. Factoring: finding a vector $\mathbf{v}$ orthogonal to both $\mathbf{x}$ and $\mathbf{y}$ mod N is then a simple matter. It will not be orthogonal to $\mathbf{a}$ with overwhelming probability.

More details

All three steps involve standard orthogonal lattice techniques, as used by Nguyen and Stern in the late 90's.

1. Linearization: to find short vectors $\mathbf{u_j}$ such that $\mathbf{u_j} \cdot \mathbf{a} \equiv 0 \pmod{N}$, apply LLL-reduction. Then, letting $\alpha_j = \mathbf{u_j} \cdot \mathbf{x}$, $\beta_j = \mathbf{u_j} \cdot \mathbf{y}$, we get $\alpha_j + c\beta_j \equiv 0 \pmod{p}$.
2. Orthogonalization: (α_j, β_j) is a short vector in a lattice $L(c, p) \subset \mathbb{Z}^2$. If it is short enough, it must be $(0, 0)$, hence the $\mathbf{u_j}$ are all orthogonal to $\mathbf{x}$, $\mathbf{y}$ over $\mathbb{Z}$.
   Say we have $\ell - 2$ of those $\mathbf{u_j}$: then their orthogonal lattice in $\mathbb{Z}^\ell$ is 2-dimensional and contains $\mathbf{x}, \mathbf{y}$. Find a basis of this lattice using LLL.
3. Factoring: finding a vector $\mathbf{v}$ orthogonal to both $\mathbf{x}$ and $\mathbf{y}$ mod N is then a simple matter. It will not be orthogonal to $\mathbf{a}$ with overwhelming probability.

More details

All three steps involve standard orthogonal lattice techniques, as used by Nguyen and Stern in the late 90's.

1. Linearization: to find short vectors $\mathbf{u_j}$ such that $\mathbf{u_j} \cdot \mathbf{a} \equiv 0 \pmod{N}$, apply LLL-reduction. Then, letting $\alpha_j = \mathbf{u_j} \cdot \mathbf{x}$, $\beta_j = \mathbf{u_j} \cdot \mathbf{y}$, we get $\alpha_j + c\beta_j \equiv 0 \pmod{p}$.
2. Orthogonalization: (α_j, β_j) is a short vector in a lattice $L(c, p) \subset \mathbb{Z}^2$. If it is short enough, it must be $(0, 0)$, hence the $\mathbf{u_j}$ are all orthogonal to $\mathbf{x}$, $\mathbf{y}$ over $\mathbb{Z}$.
   Say we have $\ell - 2$ of those $\mathbf{u_j}$: then their orthogonal lattice in $\mathbb{Z}^\ell$ is 2-dimensional and contains $\mathbf{x}, \mathbf{y}$. Find a basis of this lattice using LLL.
3. Factoring: finding a vector $\mathbf{v}$ orthogonal to both $\mathbf{x}$ and $\mathbf{y}$ mod N is then a simple matter. It will not be orthogonal to $\mathbf{a}$ with overwhelming probability.

Outline

Size Constraints

For the attack to work, we need the (α_j, β_j) from the previous slide to be "short enough." How short is short enough?

Heuristically, the shortest vector in the lattice $L(c, p) \subset \mathbb{Z}^2$ is of length $\approx \sqrt{p}$. Thus, if $|\alpha_j| \cdot |\beta_j| < p \approx N^{1/2}$, we expect the attack to work.

Let N^γ and N^δ be the bounds on x_i and y_i. The LLL-reduced vectors $\mathbf{u_j}$ have components smaller than about $N^{1/\ell}$, so:

$$|\alpha_j| = |\mathbf{u_j} \cdot \mathbf{x}| \lesssim N^{1/\ell+\gamma} \quad |\beta_j| = |\mathbf{u_j} \cdot \mathbf{y}| \lesssim N^{1/\ell+\delta}$$

Hence the heuristic size constraint:

$$\frac{2}{\ell} + \gamma + \delta < \frac{1}{2}$$

Size Constraints

For the attack to work, we need the (α_j, β_j) from the previous slide to be "short enough." How short is short enough?

Heuristically, the shortest vector in the lattice $L(c, p) \subset \mathbb{Z}^2$ is of length $\approx \sqrt{p}$. Thus, if $|\alpha_j| \cdot |\beta_j| < p \approx N^{1/2}$, we expect the attack to work.

Let N^γ and N^δ be the bounds on x_i and y_i. The LLL-reduced vectors $\mathbf{u_j}$ have components smaller than about $N^{1/\ell}$, so:

$$|\alpha_j| = |\mathbf{u_j} \cdot \mathbf{x}| \lesssim N^{1/\ell + \gamma} \quad |\beta_j| = |\mathbf{u_j} \cdot \mathbf{y}| \lesssim N^{1/\ell + \delta}$$

Hence the heuristic size constraint:

$$\frac{2}{\ell} + \gamma + \delta < \frac{1}{2}$$

Size Constraints

For the attack to work, we need the (α_j, β_j) from the previous slide to be "short enough." How short is short enough?

Heuristically, the shortest vector in the lattice $L(c, p) \subset \mathbb{Z}^2$ is of length $\approx \sqrt{p}$. Thus, if $|\alpha_j| \cdot |\beta_j| < p \approx N^{1/2}$, we expect the attack to work.

Let N^γ and N^δ be the bounds on x_i and y_i. The LLL-reduced vectors $\mathbf{u_j}$ have components smaller than about $N^{1/\ell}$, so:

$$|\alpha_j| = |\mathbf{u_j} \cdot \mathbf{x}| \lesssim N^{1/\ell+\gamma} \quad |\beta_j| = |\mathbf{u_j} \cdot \mathbf{y}| \lesssim N^{1/\ell+\delta}$$

Hence the heuristic size constraint:

$$\frac{2}{\ell} + \gamma + \delta < \frac{1}{2}$$

Size Constraints

For the attack to work, we need the (α_j, β_j) from the previous slide to be "short enough." How short is short enough?

Heuristically, the shortest vector in the lattice $L(c, p) \subset \mathbb{Z}^2$ is of length $\approx \sqrt{p}$. Thus, if $|\alpha_j| \cdot |\beta_j| < p \approx N^{1/2}$, we expect the attack to work.

Let N^γ and N^δ be the bounds on x_i and y_i. The LLL-reduced vectors $\mathbf{u_j}$ have components smaller than about $N^{1/\ell}$, so:

$$|\alpha_j| = |\mathbf{u_j} \cdot \mathbf{x}| \lesssim N^{1/\ell+\gamma} \quad |\beta_j| = |\mathbf{u_j} \cdot \mathbf{y}| \lesssim N^{1/\ell+\delta}$$

Hence the heuristic size constraint:

$$\frac{2}{\ell} + \gamma + \delta < \frac{1}{2}$$

Implementation

We implemented the attack in SAGE, and simulated its application to random faults on ISO 9796-2 signatures:

1. Generate correct mod-p parts $(\sigma_p)_i \equiv \mu(m_i)^d \pmod{p}$.
2. Pick random mod-q parts $(\sigma'_q)_i \in \mathbb{Z}_q$.
3. Compute the corresponding faulty σ'_i with the CRT, and carry out the attack.

We used random 1024-bit moduli, and tested various parameters γ, δ, first to verify the heuristic size constraint, and then to compare our attack to CJKNP. Experiments where conducted on a single 2.5 GHz Intel CPU core.

Implementation

We implemented the attack in SAGE, and simulated its application to random faults on ISO 9796-2 signatures:

1. Generate correct mod-p parts $(\sigma_p)_i \equiv \mu(m_i)^d \pmod{p}$.
2. Pick random mod-q parts $(\sigma'_q)_i \in \mathbb{Z}_q$.
3. Compute the corresponding faulty σ'_i with the CRT, and carry out the attack.

We used random 1024-bit moduli, and tested various parameters γ, δ, first to verify the heuristic size constraint, and then to compare our attack to CJKNP. Experiments where conducted on a single 2.5 GHz Intel CPU core.

Implementation

We implemented the attack in SAGE, and simulated its application to random faults on ISO 9796-2 signatures:

1. Generate correct mod-p parts $(\sigma_p)_i \equiv \mu(m_i)^d \pmod{p}$.
2. Pick random mod-q parts $(\sigma'_q)_i \in \mathbb{Z}_q$.
3. Compute the corresponding faulty σ'_i with the CRT, and carry out the attack.

We used random 1024-bit moduli, and tested various parameters γ, δ, first to verify the heuristic size constraint, and then to compare our attack to CJKNP. Experiments where conducted on a single 2.5 GHz Intel CPU core.

Implementation

We implemented the attack in SAGE, and simulated its application to random faults on ISO 9796-2 signatures:

1. Generate correct mod-p parts $(\sigma_p)_i \equiv \mu(m_i)^d \pmod{p}$.
2. Pick random mod-q parts $(\sigma'_q)_i \in \mathbb{Z}_q$.
3. Compute the corresponding faulty σ'_i with the CRT, and carry out the attack.

We used random 1024-bit moduli, and tested various parameters γ, δ, first to verify the heuristic size constraint, and then to compare our attack to CJKNP. Experiments where conducted on a single 2.5 GHz Intel CPU core.

Implementation

We implemented the attack in SAGE, and simulated its application to random faults on ISO 9796-2 signatures:

1. Generate correct mod-p parts $(\sigma_p)_i \equiv \mu(m_i)^d \pmod{p}$.
2. Pick random mod-q parts $(\sigma'_q)_i \in \mathbb{Z}_q$.
3. Compute the corresponding faulty σ'_i with the CRT, and carry out the attack.

We used random 1024-bit moduli, and tested various parameters γ, δ, first to verify the heuristic size constraint, and then to compare our attack to CJKNP. Experiments where conducted on a single 2.5 GHz Intel CPU core.

Verifying the Size Constraint

For $\gamma + \delta = 1/3$, our heuristic argument predicts that 13 faults are needed to factor N. Very well verified in practice, both for balanced and unbalanced γ, δ.

Number of faults ℓ	12	13	14
Success rate with $\gamma = \delta = \frac{1}{6}$	13%	100%	100%
Success rate with $\gamma = \frac{1}{4}$, $\delta = \frac{1}{12}$	0%	100%	100%
Average CPU time (seconds)	0.19	0.14	0.17

Comparison to CJKNP

Number of required faults, lattice dimension and CPU time for various UMP sizes, in our new attack (left) and the CJKNP attack (right).

$\gamma + \delta$	ℓ_{new}	ω_{new}	CPU time	ℓ_{old}	ω_{old}	CPU time
0.204	7	8	0.03 s	3	84	49 s
0.214	8	9	0.04 s	2	126	22 min
0.230	8	9	0.04 s	2	462	centuries?
0.280	10	11	0.07 s	6	6188	—
0.330	14	15	0.17 s	8	2^{21}	—
0.400	25	26	1.44 s	—	—	—
0.450	70	71	36.94 s	—	—	—

Fast with parameters well beyond the reach of CJKNP. However, more faults needed for any given UMP size: the CJKNP attack is preferable for very small sizes.

Outline

Further Work

- Application to EMV: the EMV specification defines a number of ISO 9796-2-based signature formats for all sorts of data, and most of them are vulnerable to this attack.
  We give an explicit example (EMV Test 2CC.086.1 Case 07) in which $\gamma + \delta = 0.28$: broken with 10 faulty signatures with our attack, but impossible to attack using CJKNP.
- Recovering unknown moduli: we show how similar techniques make it possible to recover the modulus N from a set of sufficiently many *valid* ISO 9796-2 signatures.
  This makes sense when attacking a proprietary protocol where the public parameters are not available to the attacker.

Further Work

- Application to EMV: the EMV specification defines a number of ISO 9796-2-based signature formats for all sorts of data, and most of them are vulnerable to this attack.
  We give an explicit example (EMV Test 2CC.086.1 Case 07) in which $\gamma + \delta = 0.28$: broken with 10 faulty signatures with our attack, but impossible to attack using CJKNP.
- Recovering unknown moduli: we show how similar techniques make it possible to recover the modulus *N* from a set of sufficiently many *valid* ISO 9796-2 signatures.
  This makes sense when attacking a proprietary protocol, where the public parameters are not available to the attacker.

Further Work

- Application to EMV: the EMV specification defines a number of ISO 9796-2-based signature formats for all sorts of data, and most of them are vulnerable to this attack.
  We give an explicit example (EMV Test 2CC.086.1 Case 07) in which $\gamma + \delta = 0.28$: broken with 10 faulty signatures with our attack, but impossible to attack using CJKNP.
- Recovering unknown moduli: we show how similar techniques make it possible to recover the modulus N from a set of sufficiently many *valid* ISO 9796-2 signatures.
  This makes sense when attacking a proprietary protocol, where the public parameters are not available to the attacker.

Further Work

- Application to EMV: the EMV specification defines a number of ISO 9796-2-based signature formats for all sorts of data, and most of them are vulnerable to this attack.
  We give an explicit example (EMV Test 2CC.086.1 Case 07) in which $\gamma + \delta = 0.28$: broken with 10 faulty signatures with our attack, but impossible to attack using CJKNP.
- Recovering unknown moduli: we show how similar techniques make it possible to recover the modulus N from a set of sufficiently many *valid* ISO 9796-2 signatures.
  This makes sense when attacking a proprietary protocol, where the public parameters are not available to the attacker.

Conclusion

- Given a few faulty ISO 9796-2 signatures, it is fast and easy to factor the public modulus.
- Signature formats based on this standard, such as EMV, are vulnerable.
- In situations where fault attacks are a concern, provably secure encodings, such as PSS, should be prefered.

Conclusion

- Given a few faulty ISO 9796-2 signatures, it is fast and easy to factor the public modulus.
- Signature formats based on this standard, such as EMV, are vulnerable.
- In situations where fault attacks are a concern, provably secure encodings, such as PSS, should be prefered.

Conclusion

- Given a few faulty ISO 9796-2 signatures, it is fast and easy to factor the public modulus.
- Signature formats based on this standard, such as EMV, are vulnerable.
- In situations where fault attacks are a concern, provably secure encodings, such as PSS, should be prefered.

Thank you!