Formal Analysis of CRT-RSA Vigilant’s Countermeasure Against the BellCoRe Attack

Pablo Rauzy  
rauzy@enst.fr  
pablo.rauzy.name

Sylvain Guilley  
sylvain.guilley@enst.fr  
perso.enst.fr/~guilley

Telecom ParisTech  
LTCI / COMELEC / SEN

January 25, 2014 — 16h30–17h  
PPREW 2014 @ San Diego

ACM Digital Library  
http://dx.doi.org/10.1145/2556464.2556466  
IACR ePrint 2013/810
RSA
CRT-RSA
The BellCoRe Attack
Countermeasures
Vigilant’s Countermeasure
Formal Analysis
Analysis
Results
Conclusions and Perspectives
RSA (Rivest, Shamir, Adleman)

RSA [RSA78] is an algorithm for public key cryptography. It can be used as both an encryption and a signature algorithm.

It works as follows (for simplicity we omit the padding operations):

- Let $M$ be the message, $(N, e)$ the public key, and $(N, d)$ the private key such that $d \cdot e \equiv 1 \pmod{\varphi(N)}$.
- The signature $S$ is computed by $S \equiv M^d \pmod{N}$.
- The signature can be verified by checking that $M \equiv S^e \pmod{N}$. 
RSA (Rivest, Shamir, Adleman)

RSA [RSA78] is an algorithm for public key cryptography. It can be used as both an encryption and a signature algorithm.

It works as follows (for simplicity we omit the padding operations):

- Let $M$ be the message, $(N, e)$ the public key, and $(N, d)$ the private key such that $d \cdot e \equiv 1 \mod \varphi(N)$.
- The signature $S$ is computed by $S \equiv M^d \mod N$.
- The signature can be verified by checking that $M \equiv S^e \mod N$. 

Pablo Rauzy (Telecom ParisTech)
Formal Analysis of Vigilant’s CRT-RSA
PPREW 2014
CRT (Chinese Remainder Theorem)

CRT-RSA [Koç94] is an optimization of the RSA computation which allows a fourfold speedup.

It works as follows:

- Let $p$ and $q$ be the primes from the key generation ($N = p \cdot q$).
- These values are pre-computed (considered part of the private key):
  - $d_p = d \mod (p - 1)$
  - $d_q = d \mod (q - 1)$
  - $i_q = q^{-1} \mod p$
- $S$ is then computed as follows:
  - $S_p = M^{d_p} \mod p$
  - $S_q = M^{d_q} \mod q$
  - $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$
  (recombination method of [Gar65]).
CRT (Chinese Remainder Theorem)

CRT-RSA [Koç94] is an optimization of the RSA computation which allows a fourfold speedup.

It works as follows:

- Let $p$ and $q$ be the primes from the key generation ($N = p \cdot q$).
- These values are pre-computed (considered part of the private key):
  - $d_p \equiv d \mod (p - 1)$
  - $d_q \equiv d \mod (q - 1)$
  - $i_q \equiv q^{-1} \mod p$
- $S$ is then computed as follows:
  - $S_p = M^{d_p} \mod p$
  - $S_q = M^{d_q} \mod q$
  - $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$
    (recombination method of [Gar65]).
CRT-RSA

CRT (Chinese Remainder Theorem)

CRT-RSA [Koç94] is an optimization of the RSA computation which allows a fourfold speedup.

It works as follows:

- Let $p$ and $q$ be the primes from the key generation ($N = p \cdot q$).
- These values are pre-computed (considered part of the private key):
  - $d_p \equiv d \mod (p - 1)$
  - $d_q \equiv d \mod (q - 1)$
  - $i_q \equiv q^{-1} \mod p$
- $S$ is then computed as follows:
  - $S_p = M^{d_p} \mod p$
  - $S_q = M^{d_q} \mod q$
  - $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$
    (recombination method of [Gar65]).
The BellCoRe attack [BDL97] consists in revealing the secret primes $p$ and $q$ by faulting the computation. It is very powerful as it works even with very random faulting.

It works as follows:

- The intermediate variable $S_p$ (resp. $S_q$) is faulted as $\hat{S}_p$ (resp. $\hat{S}_q$).
- The attacker thus gets an erroneous signature $\hat{S}$.
- The attacker can recover $p$ (resp. $q$) as $\gcd(N, S - \hat{S})$. 
The BellCoRe attack [BDL97] consists in revealing the secret primes $p$ and $q$ by faulting the computation. It is very powerful as it works even with very random faulting.

It works as follows:

- The intermediate variable $S_p$ (resp. $S_q$) is faulted as $\widehat{S}_p$ (resp. $\widehat{S}_q$).
- The attacker thus gets an erroneous signature $\widehat{S}$.
- The attacker can recover $p$ (resp. $q$) as $\gcd(N, S - \widehat{S})$. 
For all integer $x$, $\gcd(N, x)$ can only take 4 values:

- $1$, if $N$ and $x$ are co-prime,
- $p$, if $x$ is a multiple of $p$,
- $q$, if $x$ is a multiple of $q$,
- $N$, if $x$ is a multiple of both $p$ and $q$, i.e., of $N$. 

If $S_p$ is faulted (i.e., replaced by $\hat{S}_p \neq S_p$):

- $S - \hat{S} = q \cdot \left( (i_q \cdot (S_p - S_q) \mod p) - (i_q \cdot (\hat{S}_p - S_q) \mod p) \right)$

$\Rightarrow \gcd(N, S - \hat{S}) = q$
If $S_q$ is faulted (i.e., replaced by $\hat{S}_q \neq S_q$):

$S - \hat{S} \equiv (S_q - \hat{S}_q) - (q \mod p) \cdot i_q \cdot (S_q - \hat{S}_q) \equiv 0 \mod p$

(because $(q \mod p) \cdot i_q \equiv 1 \mod p$)

$\Rightarrow \gcd(N, S - \hat{S}) = p$
If $S_q$ is faulted (i.e., replaced by $\hat{S}_q \neq S_q$):

\[ S - \hat{S} \equiv (S_q - \hat{S}_q) - (q \mod p) \cdot i_q \cdot (S_q - \hat{S}_q) \equiv 0 \mod p \]

(because $(q \mod p) \cdot i_q \equiv 1 \mod p$)

$\Rightarrow \gcd(N, S - \hat{S}) = p$
Several protections against the BellCoRe attacks have been proposed.

Some of them are given below:

- Obvious countermeasures: no CRT, or with signature verification;
- Shamir [Sha99];
- Aumüller et al. [ABF⁺02];
- Vigilant, original [Vig08] and with some corrections by Coron et al. [CGM⁺10];
- Rivain [Riv09];
- Blömer et al. [BOS03];
- Kim et al. [KKHH11].
Several protections against the BellCoRe attacks have been proposed.

Some of them are given below:

- Obvious countermeasures: no CRT, or with signature verification;
- Shamir [Sha99];
- Aumüller et al. [ABF+02];
- Vigilant, original [Vig08] and with some corrections by Coron et al. [CGM+10];
- Rivain [Riv09];
- Blömer et al. [BOS03];
- Kim et al. [KKHH11].

See our paper in the Journal of Cryptographic Engineering:
http://dx.doi.org/10.1007/s13389-013-0065-3
http://eprint.iacr.org/2013/506
Several protections against the BellCoRe attacks have been proposed.

Some of them are given below:

- Obvious countermeasures: no CRT, or with signature verification;
- Shamir [Sha99];
- Aumüller et al. [ABF⁺02];
- Vigilant, original [Vig08] and with some corrections by Coron et al. [CGM⁺10];
- Rivain [Riv09];
- Blömer et al. [BOS03];
- Kim et al. [KKHH11].
Vigilant’s Countermeasure

- All the CRT computations (even the recombination) is carried out in an overring $\mathbb{Z}_{N_r^2}$ of $\mathbb{Z}_N$, where $r$ is a small random number (coprime with $N$).

  - $M$ is transformed into $M^*$ such that
    - $M^* \equiv M \mod N$, and
    - $M^* \equiv 1 + r \mod r^2$.

  - Let $S^* = M^{*d} \mod Nr^2$, then
    - $S^* \equiv M^d \mod N$, and
    - $S^* \equiv 1 + dr \mod r^2$,
      using of the binomial theorem in the $\mathbb{Z}_{r^2}$ subring.

  - If the verification $S^* \overset{?}{=} 1 + dr \mod r^2$ succeeds, then the final result $S = S^* \mod N$ is returned.
Vigilant's Countermeasure

- All the CRT computations (even the recombination) is carried out in an overring $\mathbb{Z}_{N^2r}$ of $\mathbb{Z}_N$, where $r$ is a small random number (coprime with $N$).

- $M$ is transformed into $M^*$ such that
  - $M^* \equiv M \mod N$, and
  - $M^* \equiv 1 + r \mod r^2$.

- Let $S^* = M^d \mod Nr^2$, then
  - $S^* \equiv M^d \mod N$, and
  - $S^* \equiv 1 + dr \mod r^2$, using of the binomial theorem in the $\mathbb{Z}_{r^2}$ subring.

- If the verification $S^* \equiv 1 + dr \mod r^2$ succeeds, then the final result $S = S^* \mod N$ is returned.
All the CRT computations (even the recombination) is carried out in an overring $\mathbb{Z}_{N^r^2}$ of $\mathbb{Z}_N$, where $r$ is a small random number (coprime with $N$).

- $M$ is transformed into $M^*$ such that
  - $M^* \equiv M \mod N$, and
  - $M^* \equiv 1 + r \mod r^2$.

- Let $S^* = M^{*d} \mod Nr^2$, then
  - $S^* \equiv M^d \mod N$, and
  - $S^* \equiv 1 + dr \mod r^2$, using of the binomial theorem in the $\mathbb{Z}_{r^2}$ subring.

- If the verification $S^* \equiv 1 + dr \mod r^2$ succeeds, then the final result $S = S^* \mod N$ is returned.
All the CRT computations (even the recombination) is carried out in an overring $\mathbb{Z}_{Nr^2}$ of $\mathbb{Z}_N$, where $r$ is a small random number (coprime with $N$).

- $M$ is transformed into $M^*$ such that
  - $M^* \equiv M \pmod{N}$, and
  - $M^* \equiv 1 + r \pmod{r^2}$.

- Let $S^* = M^{*d} \pmod{Nr^2}$, then
  - $S^* \equiv M^d \pmod{N}$, and
  - $S^* \equiv 1 + dr \pmod{r^2}$,
    using of the binomial theorem in the $\mathbb{Z}_{r^2}$ subring.

If the verification $S^* \overset{?}{=} 1 + dr \pmod{r^2}$ succeeds, then the final result $S = S^* \pmod{N}$ is returned.
Three small modifications are proposed by the authors.

After that, a safety-claim is made, however

"Formal proof of the FA-resistance of Vigilant’s scheme including our countermeasures is still an open (and challenging) issue."
Input: Message $M$, key $(p, q, d_p, d_q, i_q)$.

Output: Signature $M^d \mod N$.

1. Choose random numbers $r, R_1, R_2, R_3, \text{ and } R_4$.
2. $p' = pr^2$
3. $M_p = M \mod p'$
4. $i_{pr} = p^{-1} \mod r^2$
5. $B_p = p \cdot i_{pr}$
6. $A_p = 1 - B_p \mod p'$
7. $M_p' = A_p M_p + B_p \cdot (1 + r) \mod p'$
8. if $M_p' \not\equiv M \mod p$ then
   return error
9. end
10. $d'_p = d_p + R_1 \cdot (p - 1)$
11. $S_{pr} = M_p'^{d'_p} \mod p'$
12. if $d'_p \not\equiv d_p \mod p - 1$ then
   return error
13. end
14. $B_p S_{pr} \not\equiv B_p \cdot (1 + d'_p r) \mod p'$ then
   return error
15. end
16. $S'_p = S_{pr} - B_p \cdot (1 + d'_p r - R_3)$
17. $q' = q r^2$
18. $M_q = M \mod q'$
19. $i_{qr} = q^{-1} \mod r^2$
20. $B_q = q \cdot i_{qr}$
21. $A_q = 1 - B_q \mod q'$
22. $M_q' = A_q M_q + B_q \cdot (1 + r) \mod q'$
23. if $M_q' \not\equiv M \mod q$ then
   return error
24. end
25. if $M_p \not\equiv M_q \mod r^2$ then
   return error
26. end
27. $d'_q = dq + R_2 \cdot (q - 1)$
28. $S_{qr} = M_q'^{d'_q} \mod q'$
29. if $d'_q \not\equiv dq \mod q - 1$ then
   return error
30. end
31. if $B_q S_{qr} \not\equiv B_q \cdot (1 + d'_q r) \mod q'$ then
   return error
32. end
33. $S'_q = S_{qr} - B_q \cdot (1 + d'_q r - R_4)$
34. $S = S'_q + q \cdot (i_q \cdot (S'_p - S'_q) \mod p')$
35. $N = pq$
36. if $N \cdot (S - R_4 - q \cdot i_q \cdot (R_3 - R_4)) \not\equiv 0 \mod Nr^2$ then
   return error
37. end
38. if $q \cdot i_q \not\equiv 1 \mod p$ then
   return error
39. end
40. return $S \mod N$
Vigilant’s Countermeasure
Algorithm

**Input**: Message $M$, key $(p, q, d_p, d_q, i_q)$.

**Output**: Signature $M^d \mod N$.

1. Choose random numbers $r, R_1, R_2, R_3,$ and $R_4$.
2. $p' = pr^2$
3. $M_p = M \mod p'$
4. $i_{pr} = p^{-1} \mod r^2$
5. $B_p = p \cdot i_{pr}$
6. $A_p = 1 - B_p \mod p'$
7. $M'_p = A_p M_p + B_p \cdot (1 + r) \mod p'$
8. if $M'_p \not\equiv M \mod p$ then
   9.   return error
10. end
11. $d'_p = d_p + R_1 \cdot (p - 1)$
12. $S_{pr} = M'_p d'_p \mod p'$
13. if $d'_p \not\equiv d_p \mod p - 1$ then
   14.   return error
15. end
16. if $B_p S_{pr} \not\equiv B_p \cdot (1 + d'_p r) \mod p'$ then
   17.   return error
18. end
19. $S'_p = S_{pr} - B_p \cdot (1 + d'_p r - R_3)$
20. $q' = qr^2$
21. $M_q = M \mod q'$
22. $i_{qr} = q^{-1} \mod r^2$
23. $B_q = q \cdot i_{qr}$
24. $A_q = 1 - B_q \mod q'$
25. $M'_q = A_q M_q + B_q \cdot (1 + r) \mod q'$
26. if $M'_q \not\equiv M \mod q$ then
   27.   return error
   28. end
29. if $M_p \not\equiv M_q \mod r^2$ then
   30.   return error
31. end
32. $d'_q = d_q + R_2 \cdot (q - 1)$
33. $S_{qr} = M'_q d'_q \mod q'$
34. if $d'_q \not\equiv d_q \mod q - 1$ then
   35.   return error
36. end
37. if $B_q S_{qr} \not\equiv B_q \cdot (1 + d'_q r) \mod q'$ then
   38.   return error
39. end
40. $S'_q = S_{qr} - B_q \cdot (1 + d'_q r - R_4)$
41. $S = S'_q + q \cdot (i_q \cdot (S'_p - S'_q) \mod p')$
42. $N = pq$
43. if $N \cdot (S - R_4 - q \cdot i_q \cdot (R_3 - R_4)) \not\equiv 0 \mod Nr^2$ then
   44.   return error
45. end
46. if $q \cdot i_q \not\equiv 1 \mod p$ then
   47.   return error
48. end
49. return $S \mod N$
Vigilant’s Countermeasure

Algorithm

Input: Message $M$, key $(p, q, d_p, d_q, i_q)$.
Output: Signature $M^d \mod N$.

1. Choose random numbers $r, R_1, R_2, R_3,$ and $R_4$.
2. $p' = pr^2$
3. $M_p = M \mod p'$
4. $i_{pr} = p^{-1} \mod r^2$
5. $B_p = p \cdot i_{pr}$
6. $A_p = 1 - B_p \mod p'$
7. $M_p' = A_p M_p + B_p \cdot (1 + r) \mod p'$
8. if $M_p' \not\equiv M \mod p$ then
   --- return error
9. end
10. $d_p' = d_p + R_1 \cdot (p - 1)$
11. $S_{pr} = M_p'^{d_p'} \mod p'$
12. if $d_p' \not\equiv d_p \mod p - 1$ then
   --- return error
13. end
14. if $B_p S_{pr} \not\equiv B_p \cdot (1 + d_p' r) \mod p'$ then
   --- return error
15. end
16. $S'_p = S_{pr} - B_p \cdot (1 + d_p' r - R_3)$
17. $q' = qr^2$
18. $M_q = M \mod q'$
19. $i_{qr} = q^{-1} \mod r^2$
20. $B_q = q \cdot i_{qr}$
21. $A_q = 1 - B_q \mod q'$
22. $M_q' = A_q M_q + B_q \cdot (1 + r) \mod q'$
23. if $M_q' \not\equiv M \mod q$ then
   --- return error
24. end
25. if $M_q \not\equiv M_p \mod r^2$ then
   --- return error
26. end
27. $d_q' = dq + R_2 \cdot (q - 1)$
28. $S_{qr} = M_q'^{d_q'} \mod q'$
29. if $d_q' \not\equiv dq \mod q - 1$ then
   --- return error
30. end
31. if $B_q S_{qr} \not\equiv B_q \cdot (1 + d_q' r) \mod q'$ then
   --- return error
32. end
33. $S'_q = S_{qr} - B_q \cdot (1 + d_q' r - R_4)$
34. $S = S'_q + q \cdot (i_q \cdot (S'_p - S'_q) \mod p')$
35. $N = pq$
36. if $pq \cdot (S - R_4 - q \cdot i_q \cdot (R_3 - R_4)) \not\equiv 0 \mod N r^2$ then
   --- return error
37. end
38. if $q \cdot i_q \not\equiv 1 \mod p$ then
   --- return error
39. end
40. return $S \mod N$
Vigilant’s Countermeasure
Algorithm

Input: Message \( M \), key \((p, q, d_p, d_q, i_q)\).

Output: Signature \( M^d \mod N \).

1. Choose random numbers \( r, R_1, R_2, R_3, \) and \( R_4 \).
2. \( p' = pr^2 \)
3. \( M_p = M \mod p' \)
4. \( i_{pr} = p^{-1} \mod r^2 \)
5. \( B_p = p \cdot i_{pr} \)
6. \( A_p = 1 - B_p \mod p' \)
7. \( M'_p = A_p M_p + B_p \cdot (1 + r) \mod p' \)
8. if \( M'_p \not\equiv M \mod p \) then
   return error
9. end
10. \( d'_p = d_p + R_1 \cdot (p - 1) \)
11. \( S_{pr} = M'_p d'_p \mod p' \)
12. if \( d'_p \not\equiv d_p \mod p - 1 \) then
   return error
13. end
14. \( B_p S_{pr} \not\equiv B_p \cdot (1 + d'_p r) \mod p' \) then
   return error
15. end
16. \( S'_p = S_{pr} - B_p \cdot (1 + d'_p r - R_3) \)
17. \( q' = q r^2 \)
18. \( M_q = M \mod q' \)
19. \( i_{qr} = q^{-1} \mod r^2 \)
20. \( B_q = q \cdot i_{qr} \)
21. \( A_q = 1 - B_q \mod q' \)
22. \( M'_q = A_q M_q + B_q \cdot (1 + r) \mod q' \)
23. if \( M'_q \not\equiv M \mod q \) then
   return error
24. end
25. \( d'_q = dq + R_2 \cdot (q - 1) \)
26. \( S_{qr} = M'_q d'_q \mod q' \)
27. if \( d'_q \not\equiv dq \mod q - 1 \) then
   return error
28. end
29. \( B_q S_{qr} \not\equiv B_q \cdot (1 + d'_q r) \mod q' \) then
   return error
30. end
31. \( S'_q = S_{qr} - B_q \cdot (1 + d'_q r - R_4) \)
32. \( S = S'_q + q \cdot (i_q \cdot (S'_p - S'_q) \mod p') \)
33. \( N = pq \)
34. if \( pq \cdot (S - R_4 - q \cdot i_q \cdot (R_3 - R_4)) \not\equiv 0 \mod N r^2 \) then
   return error
35. end
36. if \( q \cdot i_q \not\equiv 1 \mod p \) then
   return error
37. end
38. return \( S \mod N \)
The goal is making sure countermeasures are trustworthy.

We want to cover a very general attacker model.

We want our proof to apply to any implementation that is a refinement of the abstract algorithm.

We want our tool to offer a full fault coverage of CRT-RSA algorithm, thereby keeping the proof valid even if the code is transformed (e.g., optimized, compiled, partitioned in software/hardware, or equipped with dedicated countermeasures).
The goal is making sure countermeasures are trustworthy.

We want to cover a very general attacker model.

We want our proof to apply to any implementation that is a refinement of the abstract algorithm.

We want our tool to offer a full fault coverage of CRT-RSA algorithm, thereby keeping the proof valid even if the code is transformed (e.g., optimized, compiled, partitioned in software/hardware, or equipped with dedicated countermeasures).
An attacker can request a CRT-RSA computation. During the computation, the attacker can fault any intermediate value. A faulted value can be zero or random. The attacker can read the final result of the computation.

Faulting can occur in the global memory (*permanent fault*) or in a local register or bus (*transient fault*). The control flow graph is untouched (however, our fault model covers some types of CFG modifications).
An attacker can request a CRT-RSA computation.
During the computation, the attacker can fault any intermediate value.
A faulted value can be zero or random.
The attacker can read the final result of the computation.

Faulting can occur in the global memory (permanent fault) or in a local register or bus (transient fault).
The control flow graph is untouched (however, our fault model covers some types of CFG modifications).
▶ Low level enough for the attack to work if protections are not implemented.

▶ Intermediate variable that would appear during refinement could be the target of an attack, but such a fault would propagate to an intermediate variable of the high level description.
Input:
- A high level description of the computation, and
- an attack success condition.

Output:
- Either the list of possible attacks, or
- a proof that the computation is resistant to fault injections.

⇒ http://pablo.rauzy.name/sensi/finja.html
How does it Works?

▶ The description of the computation is transformed into a *term*.

▶ The term is a tree which encodes:
  ▶ dependencies between the intermediate values, and
  ▶ properties of the intermediate values (such as being null, being null modulo another term, or being a multiple of another term).

▶ Each intermediate value (subterms of the tree) can be faulted, in such case its properties become:
  ▶ nothing, in the case of a randomizing fault, or
  ▶ being null, in the case of a zeroing fault.

▶ Symbolic computation by term rewriting is used to simplify the term and the attack success condition.
The description of the computation is transformed into a *term*.

The term is a tree which encodes:
- dependencies between the intermediate values, and
- properties of the intermediate values (such as being null, being null modulo another term, or being a multiple of another term).

Each intermediate value (subterms of the tree) can be faulted, in such case its properties become:
- nothing, in the case of a randomizing fault, or
- being null, in the case of a zeroing fault.

Symbolic computation by term rewriting is used to simplify the term and the attack success condition.
How does it Works?

- The description of the computation is transformed into a term.
- The term is a tree which encodes:
  - dependencies between the intermediate values, and
  - properties of the intermediate values (such as being null, being null modulo another term, or being a multiple of another term).
- Each intermediate value (subterms of the tree) can be faulted, in such case its properties become:
  - nothing, in the case of a randomizing fault, or
  - being null, in the case of a zeroing fault.
- Symbolic computation by term rewriting is used to simplify the term and the attack success condition.
Most of the $\mathbb{Z}$ ring axioms,
$\mathbb{Z}_N$ subrings,
And a few theorems.
Most of the $\mathbb{Z}$ ring axioms:
- neutral elements (0 for sums, 1 for products);
- absorbing element (0, for products);
- inverses and opposites;
- associativity and commutativity;
- but no distributivity (not confluent).

$\mathbb{Z}_N$ subrings,

And a few theorems.
Most of the \( \mathbb{Z} \) ring axioms,

\( \mathbb{Z}_N \) subrings:

- identity:
  \[
  (a \mod N) \mod N = a \mod N,
  \]
  \[
  N^k \mod N = 0;
  \]

- inverse:
  \[
  (a \mod N) \times (a^{-1} \mod N) \mod N = 1,
  \]
  \[
  (a \mod N) + (-a \mod N) \mod N = 0;
  \]

- associativity and commutativity:
  \[
  (b \mod N) + (a \mod N) \mod N = a + b \mod N,
  \]
  \[
  (a \mod N) \times (b \mod N) \mod N = a \times b \mod N;
  \]
  
  subrings: \( (a \mod N \times m) \mod N = a \mod N \).

And a few theorems.
Most of the $\mathbb{Z}$ ring axioms,

$\mathbb{Z}_N$ subrings,

And a few theorems:

- Fermat’s little theorem;
- its generalization, Euler’s theorem;
- Chinese remainder theorem;
- Binomial theorem in $\mathbb{Z}_{r^2}$ rings
  \[(1 + r)^d \equiv 1 + dr \mod r^2.\]
For each possible fault attack:

- the faulted term is simplified to propagate to modified properties;
- simplified terms (faulted and original) are then fed into the attack success condition;
- the attack success condition itself is then simplified to either true (the attack works) or false (it doesn’t).
Minimal Example of Usage

- Computation: \( t = a + b \times c \).
- Let’s say the “attack” works if \( t \not\equiv a \mod b \).

```plaintext
minimal-example.fia

noprop a, b, c;

<<
t := a + b * c;

return t;

%%
@ !=[b] a
```

- `finja minimal-example.fia -r`
- `finja minimal-example.fia -z`
▶ finja crt-rsa_vigilant.fia -t -r
▶ finja crt-rsa_vigilant.fia -t -z
▶ finja crt-rsa_vigilant-fixed.fia -t -r
▶ finja crt-rsa_vigilant-fixed.fia -t -z
▶ finja crt-rsa_vigilant-fixed.fia -s -t -n 2 -r -r
▶ finja crt-rsa_vigilant-fixed.fia -s -t -n 2 -r -z
▶ finja crt-rsa_vigilant-fixed.fia -s -t -n 2 -z -r
▶ finja crt-rsa_vigilant-fixed.fia -s -t -n 2 -z -z
▶ finja crt-rsa_vigilant-fixed_pc.fia -s -t -n 2 -r -r
▶ finja crt-rsa_vigilant-fixed_pc.fia -s -t -n 2 -r -z
▶ finja crt-rsa_vigilant-fixed_pc.fia -s -t -n 2 -z -r
▶ finja crt-rsa_vigilant-fixed_pc.fia -s -t -n 2 -z -z
“Formal proof of the FA-resistance of Vigilant’s scheme including our countermeasures is still an open (and challenging) issue.”

Jean-Sébastien Coron, Christophe Giraud, Nicolas Morin, Gilles Piret, and David Vigilant
Results

- We have formally proven the resistance of a fixed version of Vigilant’s CRT-RSA countermeasure against the BellCoRe fault injection attack.

- Our research allowed us to safely remove two out of nine verifications, thereby simplifying the protected computation of CRT-RSA while keeping it formally proved.

⇒ We have shown the importance of formal analysis in the field of implementation security. Not only for the development of trustable devices, but also as an optimization enabler, both for speed and security.
We have formally proven the resistance of a fixed version of Vigilant’s CRT-RSA countermeasure against the BellCoRe fault injection attack.

Our research allowed us to safely remove two out of nine verifications, thereby simplifying the protected computation of CRT-RSA while keeping it formally proved.

We have shown the importance of formal analysis in the field of implementation security. Not only for the development of trustable devices, but also as an optimization enabler, both for speed and security.
We have formally proven the resistance of a fixed version of Vigilant’s CRT-RSA countermeasure against the BellCoRe fault injection attack.

Our research allowed us to safely remove two out of nine verifications, thereby simplifying the protected computation of CRT-RSA while keeping it formally proved.

⇒ We have shown the importance of formal analysis in the field of implementation security. Not only for the development of trustable devices, but also as an optimization enabler, both for speed and security.
Conclusions and Perspectives

- We would like to continue improving finja:
  - take into account fault injection in the control flow as studied by Heydemann et al. [HMER13];
  - automatic variable properties refinement (for attacks by chosen message);
  - parallelizing computations...

- It would also be interesting to see if general purpose tool such as EasyCrypt [BGZB09] could be a good fit for this kind of work.
Conclusions and Perspectives

We would like to continue improving finja:
- take into account fault injection in the control flow as studied by Heydemann et al. [HMER13];
- automatic variable properties refinement (for attacks by chosen message);
- parallelizing computations...

It would also be interesting to see if general purpose tool such as EasyCrypt [BGZB09] could be a good fit for this kind of work.
References


Matthieu Rivain.
Securing rsa against fault analysis by double addition chain exponentiation.
http://eprint.iacr.org/.

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.

Adi Shamir.
Method and apparatus for protecting public key schemes from timing and fault attacks, November 1999.
Patent Number 5,991,415; also presented at the rump session of EUROCRYPT ’97.

David Vigilant.
RSA with CRT: A New Cost-Effective Solution to Thwart Fault Attacks.
RSA
CRT-RSA
The BellCoRe Attack
  Why does it Works?
Countermeasures
Vigilant’s Countermeasure
  Corrections by Coron et al.
  Algorithm
Formal Analysis
  Attacker Model
  Algorithm Description
  finja
Analysis
Results
Conclusions and Perspectives

rauzy@enst.fr