Towards Generic Countermeasures Against Fault Injection Attacks

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

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

TELECOM ParisTech
CNRS LTCI / COMELEC / SEN

TRUDEVICE 2015

March 13, 2015 @ Grenoble, France

- Allows to recover the secret primes p and q used in the secret keys of the CRT-RSA cryptosystem.
- Only requires a single fault injection and a gcd computation.

→ Many countermeasures have been developed.

State-of-the-Art Countermeasures

- Mostly resulting from engineering efforts.
- Development by trial-and-error leading to overkill protections.
- Many different countermeasures (NIH, patents), not all of them work.

- Formal studies of these countermeasures allowed to understand their working factor.

→ We were able to fix the broken ones and to simplify many of them (e.g., original Vigilant's countermeasure: broken, 9 tests, 5 random numbers; our fixed and simplified version: working, 3 tests, 1 random number).

- More importantly, the working factor is actually not tied to the BellCoRe attack, nor to the CRT-RSA algorithm.

→ It is possible to abstract it and get a recipe for cost-effectively verifying the integrity of any arithmetic computation.

Integrity Verification

- Idea: verify the integrity of the computation by introducing redundancy.
- Simply repeating the computation and comparing results is bad:
  (*a*) it is too expensive, and
  (*b*) nothing stops the attacker from injecting the same fault twice.
- Thus, existing countermeasures optimize this idea in different ways.

Entanglement

- The *entanglement* protection scheme solves both issues, by:
    - lifting the computation to an over-structure (a direct product) allowing
      (*a*) to project the result back onto the original structure, and
      (*b*) to project a checksum onto a smaller structure (e.g., `int32`-sized);
    - performing in parallel the same computation is the smaller structure;
    - both the checksum and the smaller result should be equal.
- The redundant part of the computation is almost free (arithmetic with 32-bit vs. 2,048-bit numbers).
- It is very hard to precisely fault the small computation to produce a consistent value modification.
- Limitation: possible collisions in the small structure.
  Mitigated by the possibility to use several different small structures.

- At IMDEA Software Institute (Madrid, Spain), I developed a compiler called enredo, while supervised by Gilles Barthe, François Dupressoir and Pierre-Yves Strub.
- Automated insertion of the *entanglement* countermeasure into arbitrary code.
- ~~Short demo.~~

Input

```
Field Zp ;

Zp a, b, c, d ;

a := b + (c * d) ;
return a ;
```

Output: `enredo -n 1`

```
Field Zp, Zr0 ;
Ring Zpr0 = Zp * Zr0 ;

Zp _ret_a ;
Zr0 _ar0, _br0, _cr0, _dr0 ;
Zpr0 a, b, c, d ;

_dr0 := d [Zr0] ;
_cr0 := c [Zr0] ;
_br0 := b [Zr0] ;
_ar0 := _br0 + (_cr0 * _dr0) ;
a := b + (c * d) ;
if a [Zr0] = _ar0
then
  _ret_a := a [Zp] ;
  return _ret_a ;
else
  return ERROR ;
end
```

Output: `enredo -n 2`

```
Field Zp, Zr0, Zr1 ;
Ring Zpr0r1 = Zp * Zr0 * Zr1 ;

Zp _ret_a ;
Zr0 _ar0, _br0, _cr0, _dr0 ;
Zr1 _ar1, _br1, _cr1, _dr1 ;
Zpr0r1 a, b, c, d ;

_dr1 := d [Zr1] ;
_cr1 := c [Zr1] ;
_br1 := b [Zr1] ;
_dr0 := d [Zr0] ;
_cr0 := c [Zr0] ;
_br0 := b [Zr0] ;
_ar0 := _br0 + (_cr0 * _dr0) ;
_ar1 := _br1 + (_cr1 * _dr1) ;
a := b + (c * d) ;
if a [Zr1] = _ar1 /\ a [Zr0] = _ar0
then
  _ret_a := a [Zp] ;
  return _ret_a ;
else
  return ERROR ;
end
```

Perspectives

- We already have:
  - an executable code output (Python),
  - a correctness proof of the code transformation.

- Benchmark of the cost of the countermeasure.
- Security proof.
- Protected implementations of currently unprotected algorithms.
- Practical lab tests.

- This work will appear in a chapter of *Handbook of Pairing Based Cryptography* (Editors: Nadia El Mrabet and Marc Joye).

That was it. Questions?

The BellCoRe Attack
State-of-the-Art Countermeasures
Formal Study of Countermeasures
Integrity Verification
Entanglement
enredo
Perspectives

`rauzy @ enst · fr`
`http://pablo.rauzy.name/`