Towards Generic Countermeasures Against Fault Injection Attacks

Gilles Barthe[2], François Dupressoir[2], Sylvain Guilley[1], Pablo Rauzy[1], Pierre-Yves Strub[2]

[1] **Telecom** ParisTech
[2] IMDEA Software Institute

Crypto Seminar Day @ IMDEA

Itinerant Cryptography Seminars

January 22, 2015 @ Madrid, Spain

- 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.

$\rightarrow$ Many countermeasures have been developed.

- 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.

- 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.

- 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.

- Automated insertion of the *entanglement* countermeasure into arbitrary code.
- Short demo.

- Output executable code and benchmark the cost of the countermeasure.
- Proof of correctness of the transformation.
- Security proof.
- Generate protected implementation of currently unprotected algorithms (*e.g.*, ECC).

That was it. Questions?

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

rauzy@enst.fr