Optimization of the ROCA (CVE-2017-15361) Attack

Bruno Produit
Supervisor: Arnis Paršovs, MSc

Institute of Computer Science
University of Tartu

June 2, 2019

Table of contents

Context

- 2017 Czech researchers find flaw in Infineon's key generation algorithm[1]
- 750 000 Estonian ID-cards affected
- 140.8 CPU-years to factor an Estonian ID card (worst case)
- New result: 70.4 CPU-years to factor an Estonian ID card (worst case)
- New result: For 90 % of keys, average case is 4x better than ROCA paper

[1] https://crocs.fi.muni.cz/public/papers/rsa_ccs17

Facts about the ROCA attack

- Factorization of RSA keys
- Takes advantage of the polynomial form of primes
- Prime number construction: $p = k * M + (65537^a \bmod M)$
- Vulnerable variant of Joye and Paillier's secure prime generation algorithm[2]
- Key format:

$$N = \overbrace{(k * M + (65537^a \bmod M))}^{p} * \overbrace{(l * M + (65537^b \bmod M))}^{q}$$

- Fingerprintable: $N \equiv 65537^{a+b} \ mod \ M \equiv 65537^c \ mod \ M$

[2] http://joye.site88.net/papers/JPV00gen.pdf

Transfer entropy from a to k

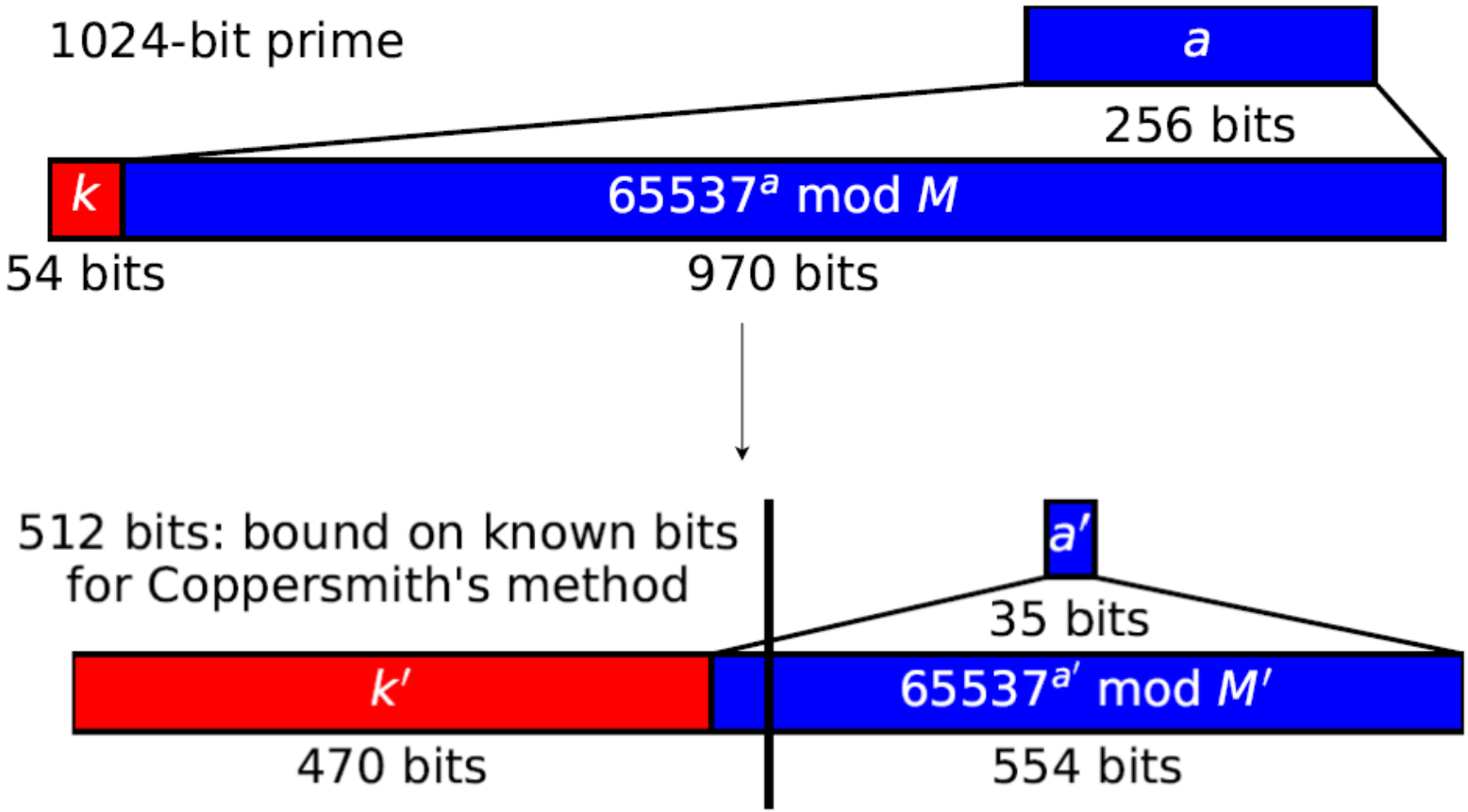

Figure: Prime form transformation[3]

[3] https://crocs.fi.muni.cz/_media/public/papers/nemec_roca_csaw_poster.pdf

Overview of the ROCA attack

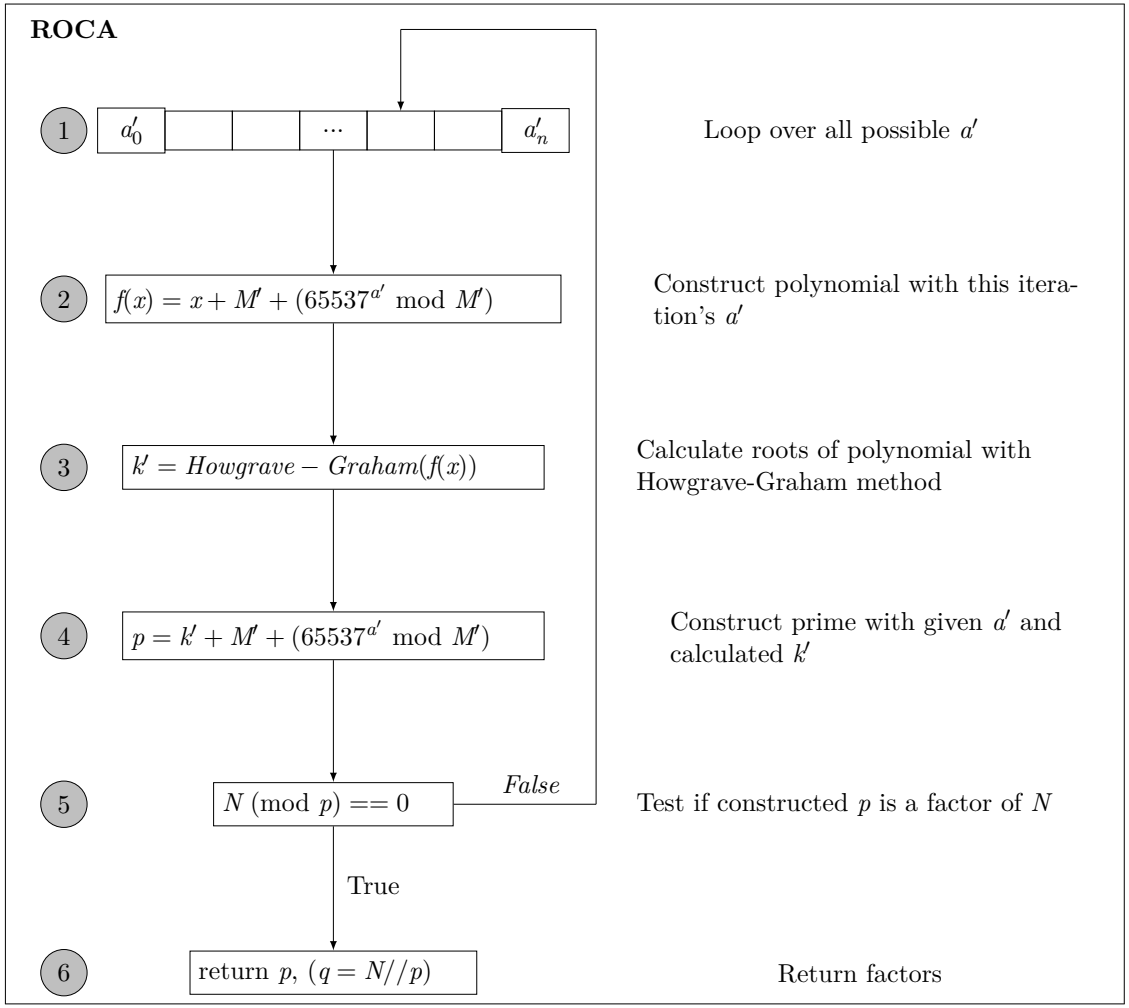

Figure: Overview of the ROCA attack

Entropy of a and k

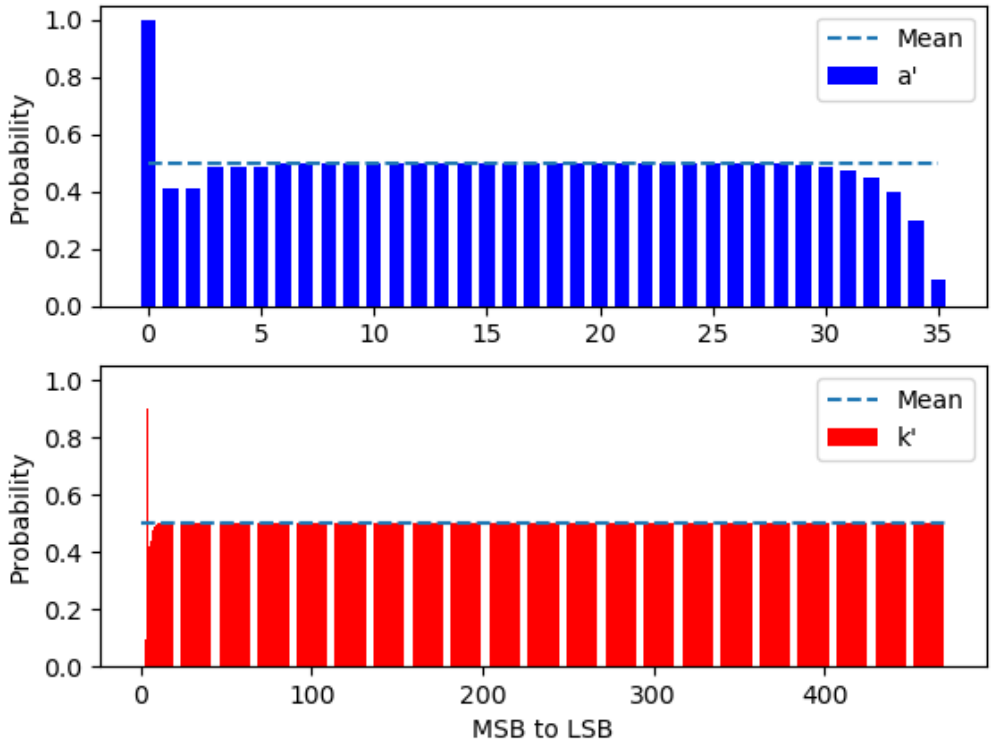

Figure: Entropy of each bit in a' and k', MSB to LSB (2048-bit keys)

Impact of the Optimizations

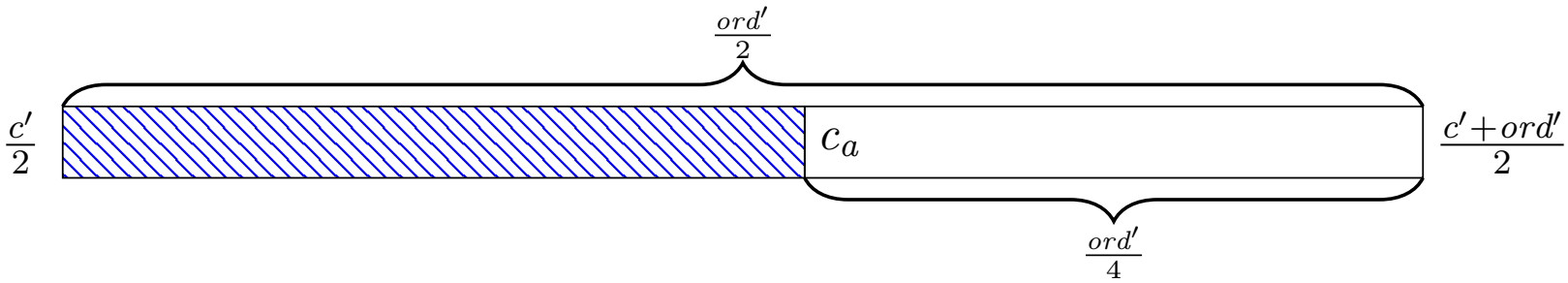

Figure: Comparison of the original and the new bruteforce range

$$p = \overbrace{(c_k + r)}^{k'} * M' + (65537^{a'} \bmod M') \tag{1}$$

Efficiency

Key size	*Non-optimized*	*Optimized*	*Optimized Random key*	*Optimized Cherry-picked*
512-bit	2.0333 CPU-hours*	2.2 CPU-hours	0.73 CPU-hours*	0.51 CPU-hours*
1024-bit	102.4 CPU-days	51.2 CPU-days	36.5 CPU-days	25.6 CPU-days
2048-bit	161.2 CPU-years	80.6 CPU-years	57.5 CPU-years	40.3 CPU-years (336$)

Table: Efficiency of the ROCA attack using HPC

Implementation

- https://blog.cr.yp.to/20171105-infineon3.txt
  - Attack with known a
  - Not using M' transformation (not needed when known a)
- https://github.com/brunoproduit/roca
  - First publicly available full attack
  - Based on SageMath

```
$ python2 roca.py data/512.pem
[+] Importing key
[+] Key is vulnerable!
[+] RSA-512 key
[+] N = 80474497870208039394761476993787283293147334261
64267535316072793294233587337682475529099270039635820
02260707371017160997944821548814875889400167842361138 9
[+] c' = 588970
[+] Time for 1 coppersmith iteration: 0.04 seconds
[+] Estimated (worst case) time needed for the attack:
4 hours, 30 minutes and 3.46 seconds
[+] Found factors of N:
[+] p = 89316585341239200103164798629168230185983346548 5
58222489515734210664382833579
[+] q = 90100284916570139638439028489735281413386074204 4
03670730318637933879173958391
[+] Took 7742.3 s
[+] Exporting key to priv.pem
```

Listing 1: Execution of the attack against vulnerable key

Parrallelization

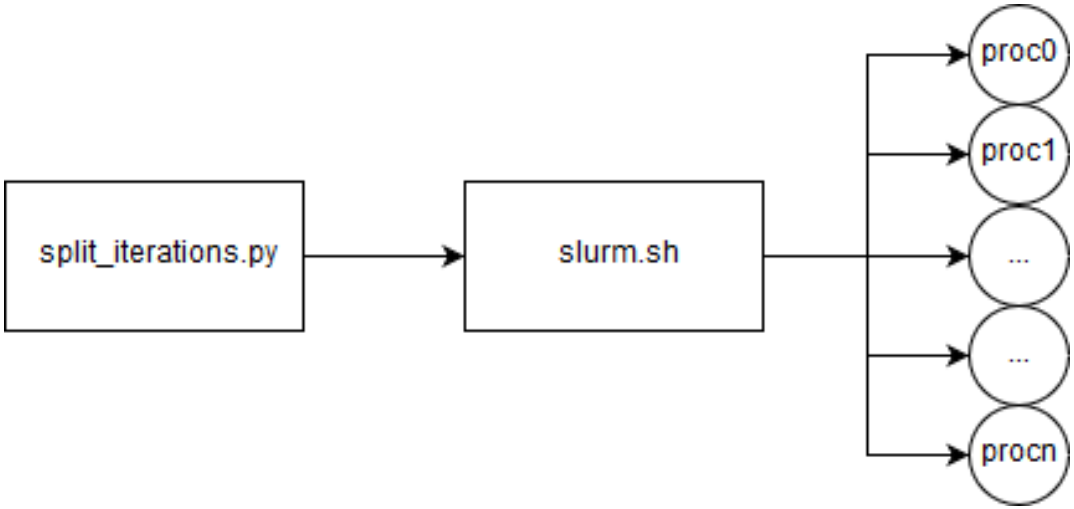

Figure: Splitting range for process allocation with given CPU cores available