Analysis and use of RSA keypair generation bias

Learning from the discovery, disclosure, and mitigation of vulnerable RSA key generation on smartcards

Marek Sýs ***syso@fi.muni.cz***

Centre for Research on Cryptography and Security, Masaryk University

Joint work with: *Petr Švenda, Matúš Nemec, Dušan Klinec, Vasilios Mavroudis, Andrea Cerulli, Dan Cvrček, George Danesis, Peter Sekan, Rudolf Kvašnovský, David Formánek, David Komárek and Vashek Matyáš*

CROCS
Centre for Research on Cryptography and Security

About CRoCS lab

- Masaryk University, Brno
- Smartcards analysis from 2002

- Cryptographic smartcards are pervasive (SIM, EMV, eID, tokens…)
- Yet smartcard industry is very closed
  - NDA just to see detailed specifications, proprietary APIs, no design details…
- Security certifications performed by testing labs (FIPS, CC)
  - But details are not public
- Idea in 2014: Infer details using keys similarity to open-source libraries

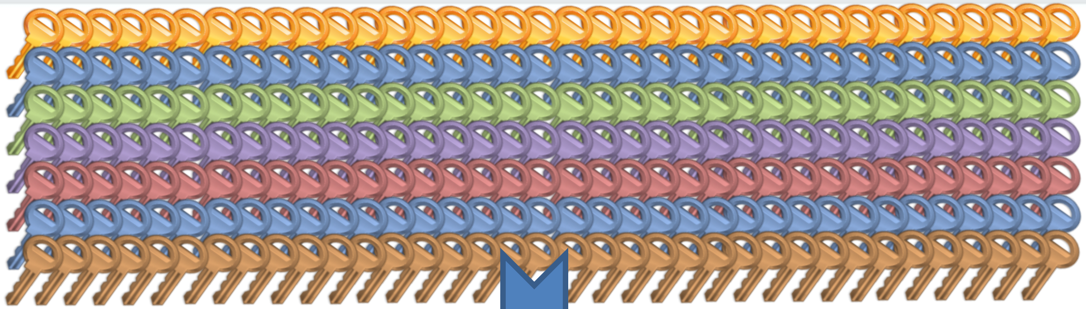
RSA primer – what does it mean and why should I care?

- RSA is widely used public-key cryptosystem (1977)
- Used for digital signatures (mail, software distribution, contracts…)
- Used for key exchange (HTTPS/TLS, PGP…)
- Private part: random primes **P** and **Q**, private exponent **d**
- Public part: modulus **N**, public exponent **e** (often 65,537)

$$P \times Q = N$$

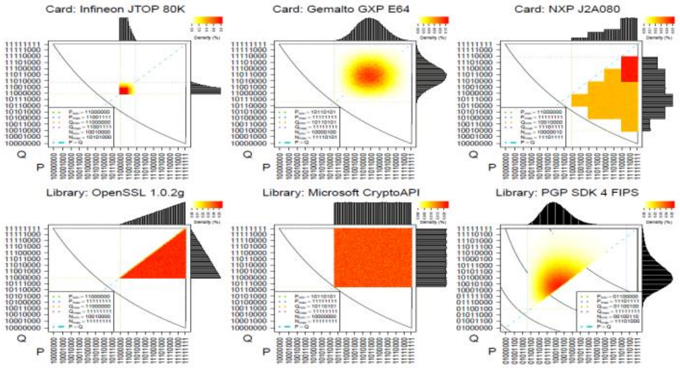
- Factorization attack: compute primes **P** and **Q** from the knowledge of **N**
- Problem: How to generate a large prime (1024- or 2048-bit length)?

60+ million fresh RSA keypairs

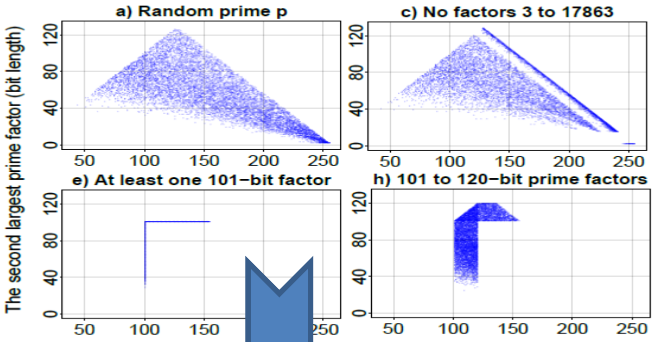


22 sw. libraries
16 smart cards

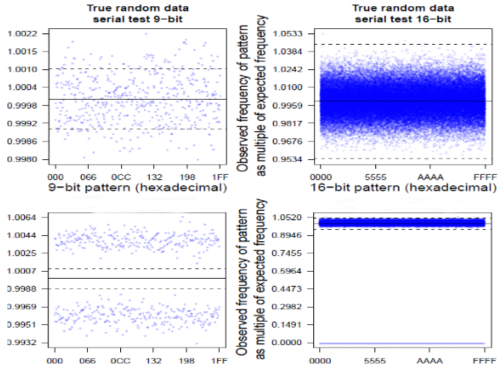
Distribution of primes (MSB)



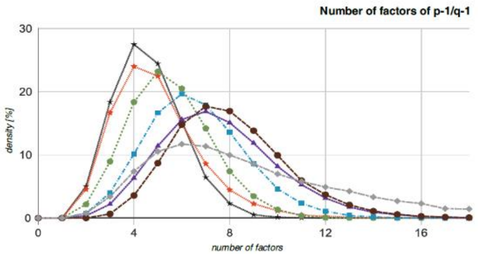
Large factors of p-1 / p+1



Bit stream statistics



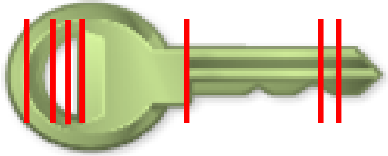
Number of factors



and more…

7 implementation choices observable in public keys

(biased bits of public modulus, “mask”)



OpenSSL

Input key

-----BEGIN CERTIFICATE-----
MIIG9zCCBd+gAwIBAgIIJOR2wFUwc20wDQYJKoZI
hvcNAQELBQAwSTELMAkGA1UEBhMCVVMxEzAR
BgNVBAoTCkdvb2ds BJ mMxJTAjBgNVBAMTHE
dvb2ds cm5ldCBF XRob3JpdHkgRzIwHhc
NMTY Nz YwOTI4MDgwMzA
wk2zl qn NRoO 9zPk/rEp4miQ9aVgC6k7i
bLukl4c e0kCQr8kNUBhH25DS6HpekTmO1s
9q81KbtS2E7+4Q/57xgdghBLiaTEv7O7+gskLQ/qJa
TouwiDPM6SHIVU6X2Ca1lNKg2wbx8h2Q63SDIwFJ
52HsNAClKp4ADvjvvImYoWVitcLlhpXogOAzbLz3HIs
6Jk=
-----END CERTIFICATE-----

111001010

Precomputed matrix

Mask value	Group I	Group II	…	Group XII	Group XIII
000000000	0.124	0.347		0.105	0.012
000000001	0.004	0.038		0.236	0.454
000000011	0.046	0.002		0.447	0.112
…					
111111110	0.394	0.044		0.320	0.002
111111111	0.046	0.347		0.015	0.312

Classification

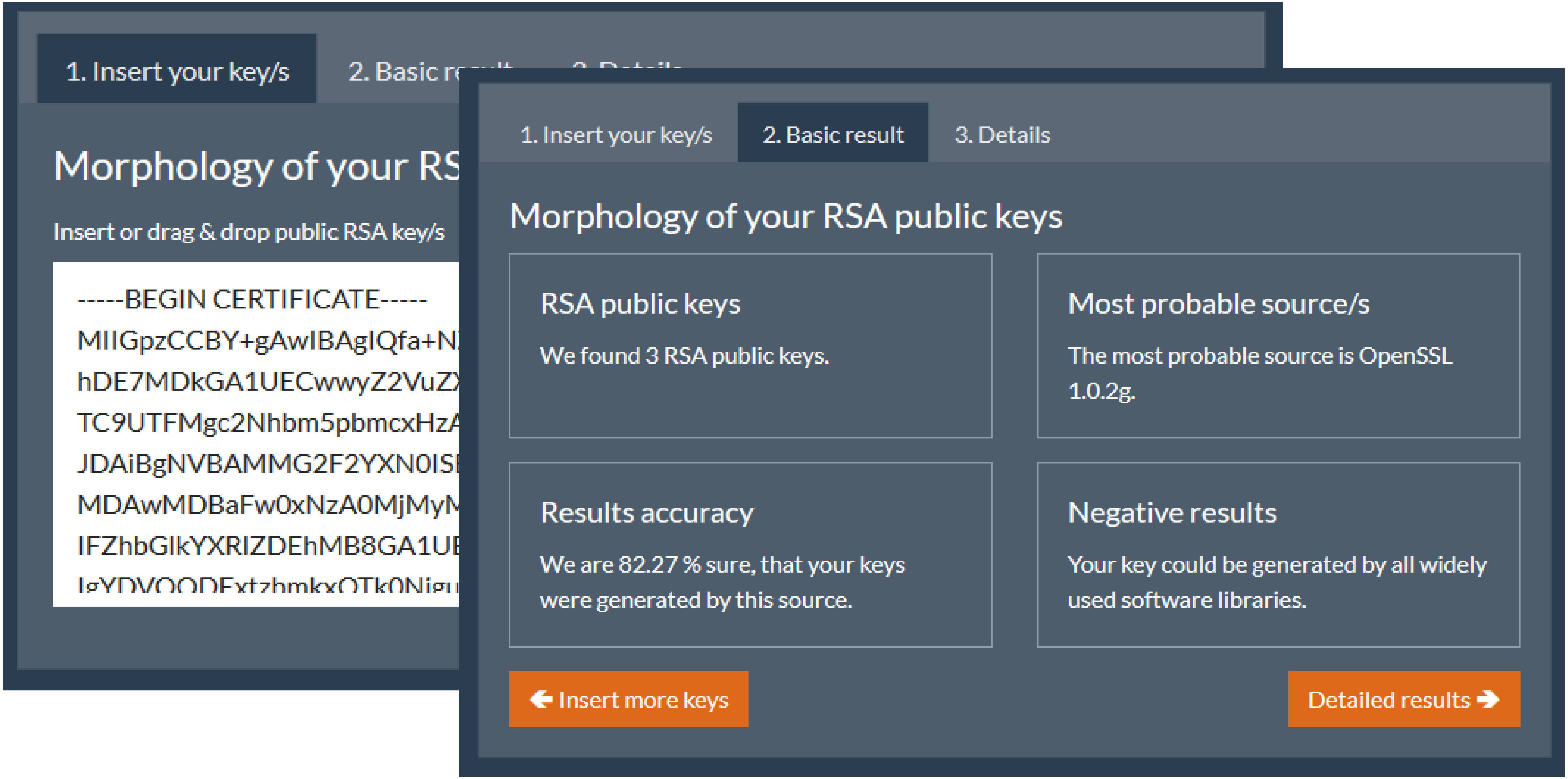
44% OpenSSL 's group

11% POLAR SSL 's group

9% PGP 's group

…

Try at https://rsa.sekan.eu



Impact (of the possibility) of public key classification

- Information leakage vulnerability

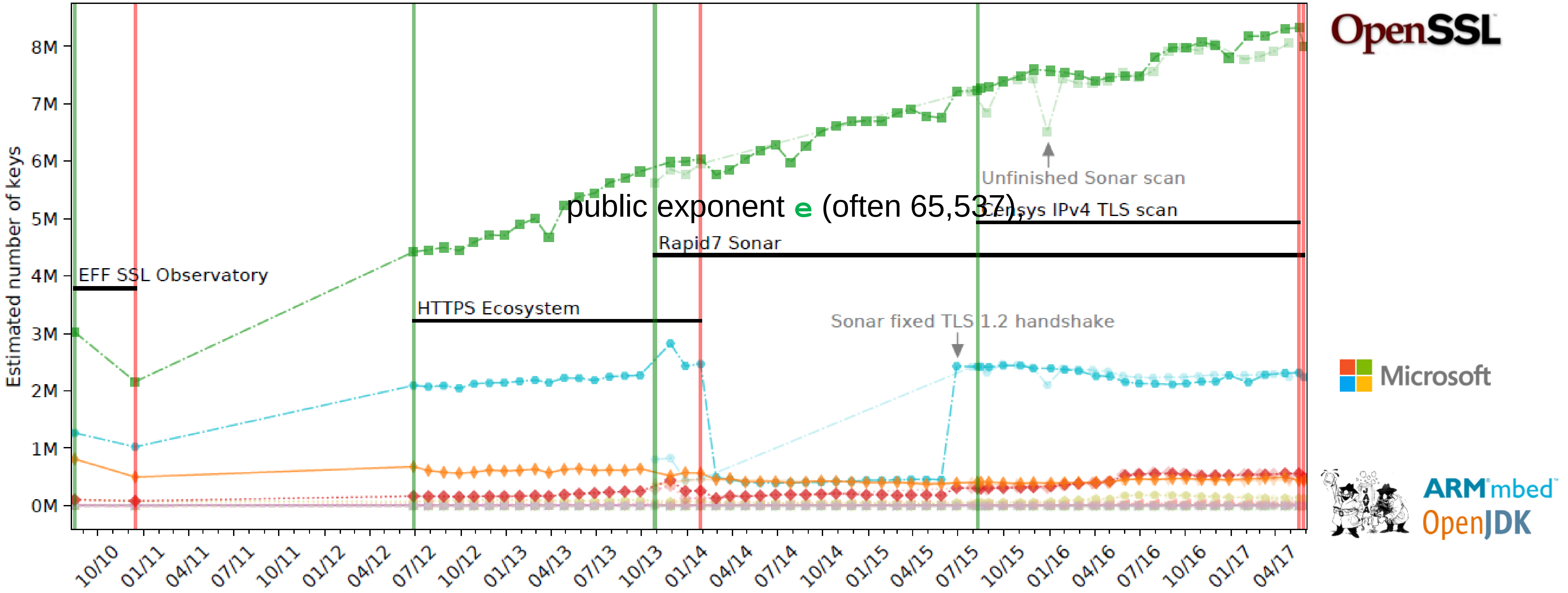
Impact (of the possibility) of public key classification

- Information leakage vulnerability

- Statistics: current library usage trends

RSA key classification
(USENIXSec'16, ACSAC'17)

Internet-wide TLS IPv4 scan



public exponent **e** (often 65,537),

Impact (of the possibility) of public key classification

- Information leakage vulnerability

- Statistics: current library usage trends

RSA key classification
(USENIXSec'16, ACSAC'17)

- Audit: identify source libs in target organization

EE eID injected keys
(Arnis Paršovs, 05/2018)

EE eID injected keys
(Arnis Paršovs, 05/2018)

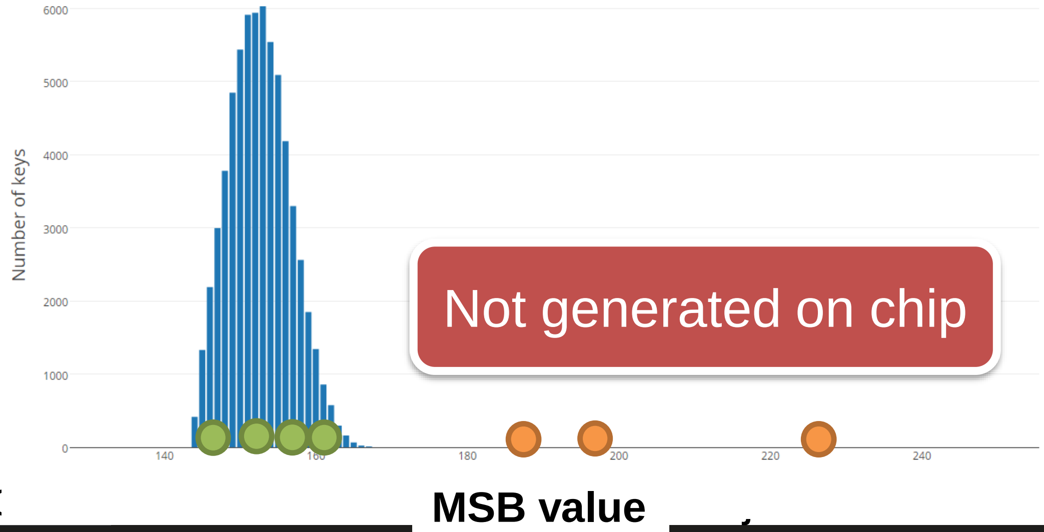
Problem reported from Estonia (17.5.2018)

The ID-card maker has violated the most important security principle and 12,500 cards need to be replaced by people.

Hans Lõug
05/27/2018 at 13:58

share Meeldib 32

- Estonian eIDs generate private key always on chip (by design)
  - Some keys found to be injected from outside
- Found by observed discrepancy in public key properties (MSB)
  - Expected: MSB $\in \{144, 145, \ldots, 167\}$
  - Observed: outside



https://geenius.ee/uudis/id-kaartide-tootja-rikkus-tahtsaimat-turvap

Impact (of the possibility) of public key classification

- Information leakage vulnerability

- Statistics: current library usage trends
- Audit: identify source libs in target organization
- Forensics: source lib/device of weak keys
- Quick search for other keys from vulnerable library

RSA key classification (USENIXSec'16, ACSAC'17)

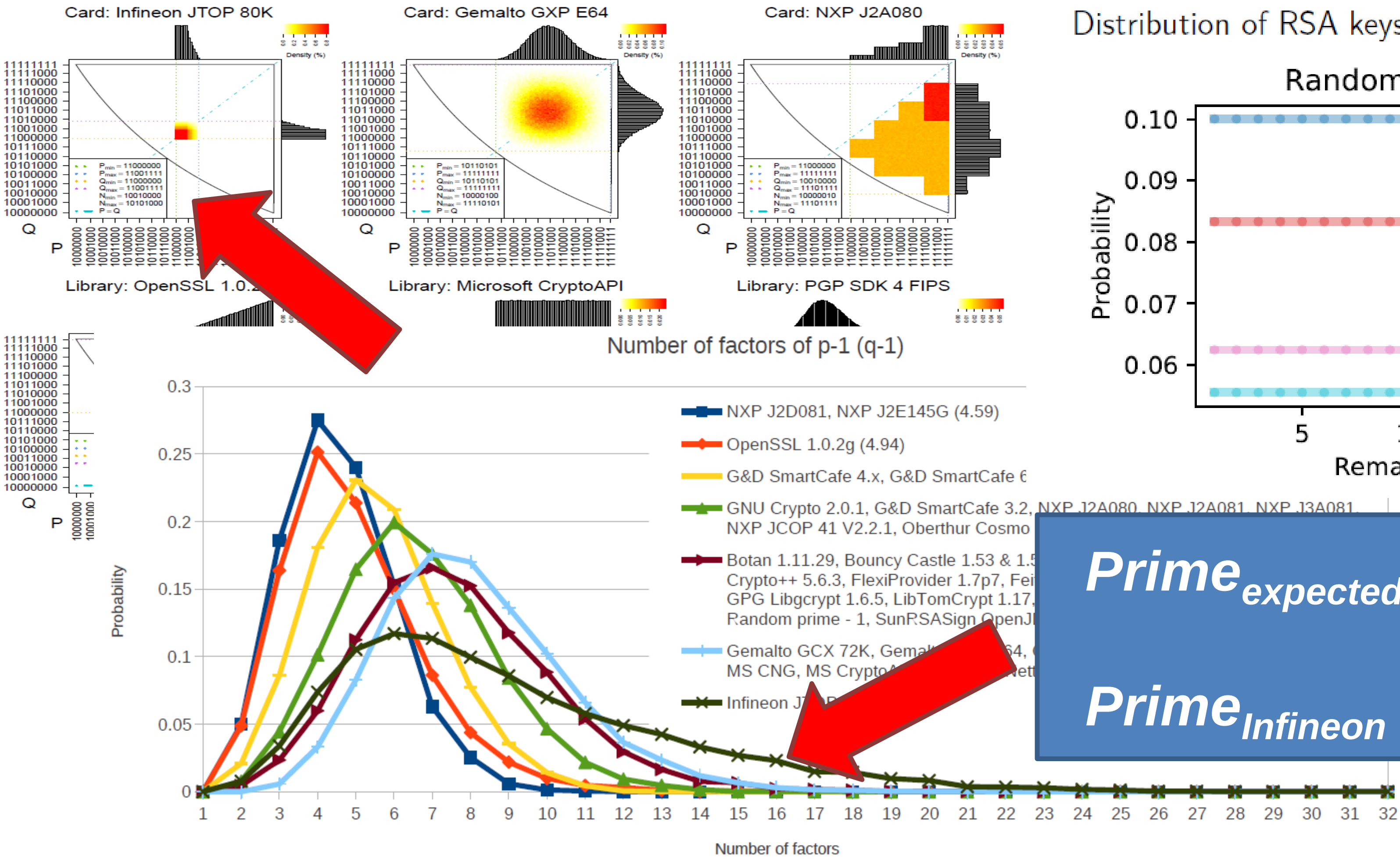
EE eID injected keys (Arnis Paršovs, 05/2018)

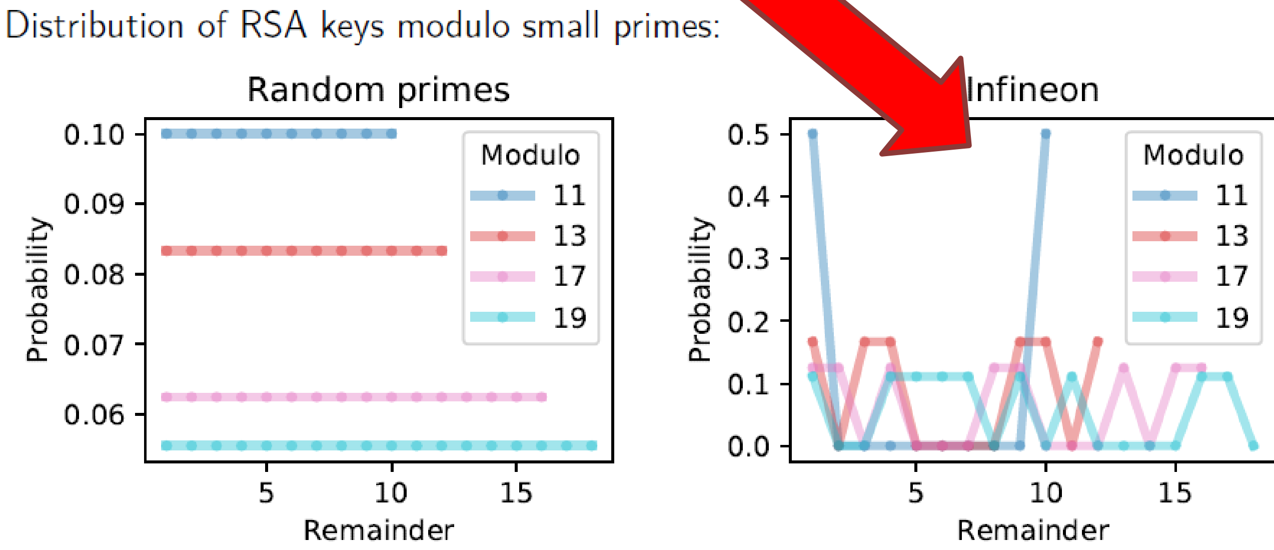
Source of factorable TLS keys (I. Mironov, 2012)

ROCA vulnerability (ACM CCS'17)

ROCA vulnerability

But we were unaware on another issue that time





Prime$_{expected}$ = random ✓

Prime$_{Infineon}$ = k * M + 65537^{a} mod M

Prime generation

slow ! - primality tests (modular exponentiation)

Algorithms

1. Random sampling – **generate** & test, **generate** & test, …
   - Many iterations – random number has typically small divisors
   - 50% that 2 is divisor, 33% that 3 is divisor …
2. Incremental search – **generate** & test, **increment** & test, increment…
   - skip numbers with small prime factors
   - similar methods Joye & Pailier algorithm , “Fast Prime” algorithm (Infineon)

Structure of Infineon primes

$$prime = k.M + 65537^{a} \bmod M,\ \ M = 2 * 3 * 5 * 7 \ldots.$$

- Entropy loss in prime:

Infineon	k	a	Determined by a
Random	Random bits		

Consequences:

- Strong fingerprint of RSA keys
- Practical factorization of RSA keys is possible

Detection of vulnerable keys

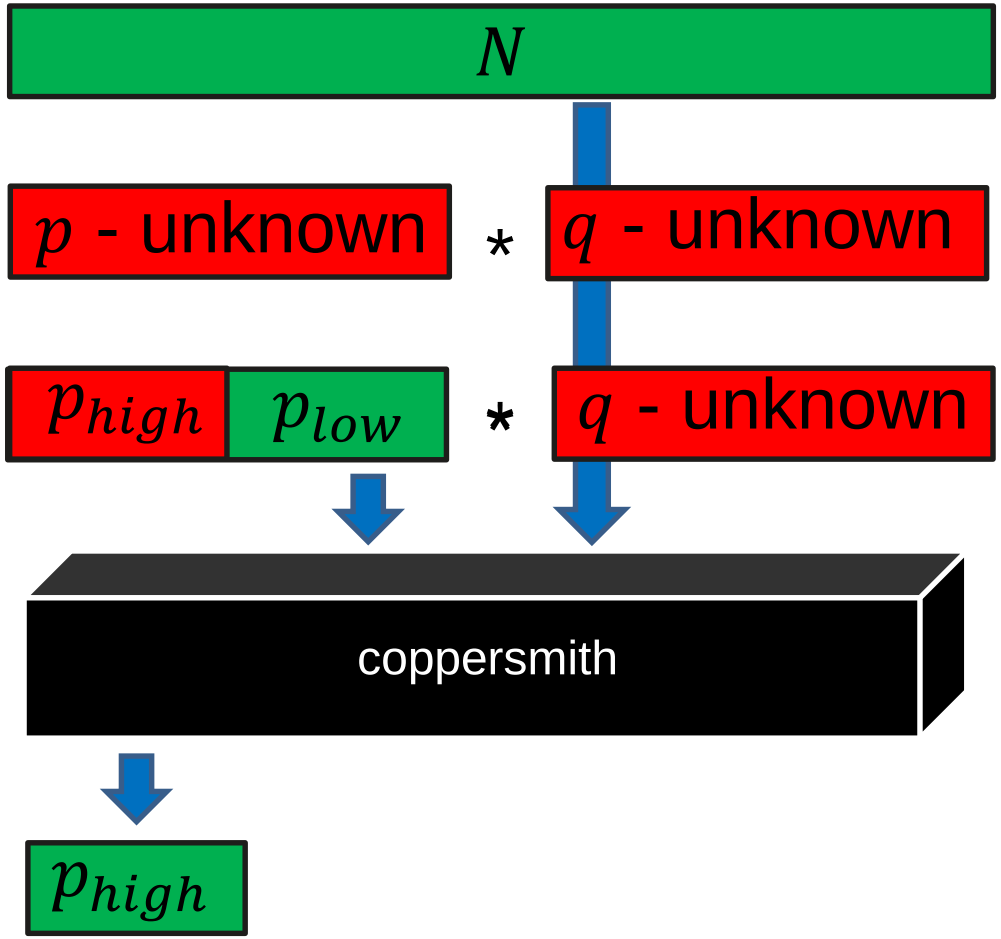
- Based on public modulus N

1. Vulnerable if c exists: $N \equiv 65537^c \; mod \; M$
2. Equivalent to c_i exist for **all** $p_i \mid M$: $N \equiv 65537^{c_i} \; mod \; p_i$
   – small $p_i \Rightarrow$ very fast - microseconds

- Errors:
  – False negatives - all Infineon primes have the specific form
  – False positives - negligible probability ($Pr < 2^{-150}$)

Coppersmith's attack as a black box

1. Modulus N
2. Unknown factors p, q
3. Partial knowledge of prime (**at least** ½ of bits of $\boldsymbol{p}$)
4. Apply Coppersmith's algorithm



Naïve algorithm (RSA -2048)

- $p = k.M + 65537^a \bmod M$
- Guess a

256 bits

a

$k.M +$ $65537^a \bmod M$

54 bits

970 bits = size of M

- compute k using Coppersmith's alg.
  (requires ½ of known bits – much more than that – large M)

- **Infeasible** – large a

How to make attack practical ?

Idea: ½ known (= size of M) bits of $\boldsymbol{p}$ is sufficient

- smaller M' ⇒ smaller (or equal) a'

- p of the **same** form ⇒ $M'|M$

$$p = \boxed{k} \,.\, M + \underbrace{\boxed{65537^{a} \bmod M}}_{\text{of size } M}$$

a

½

$$p = \boxed{k'.} \; M' + \underbrace{\boxed{65537^{a'} \bmod M'}}_{\text{of size } M'}$$

a'

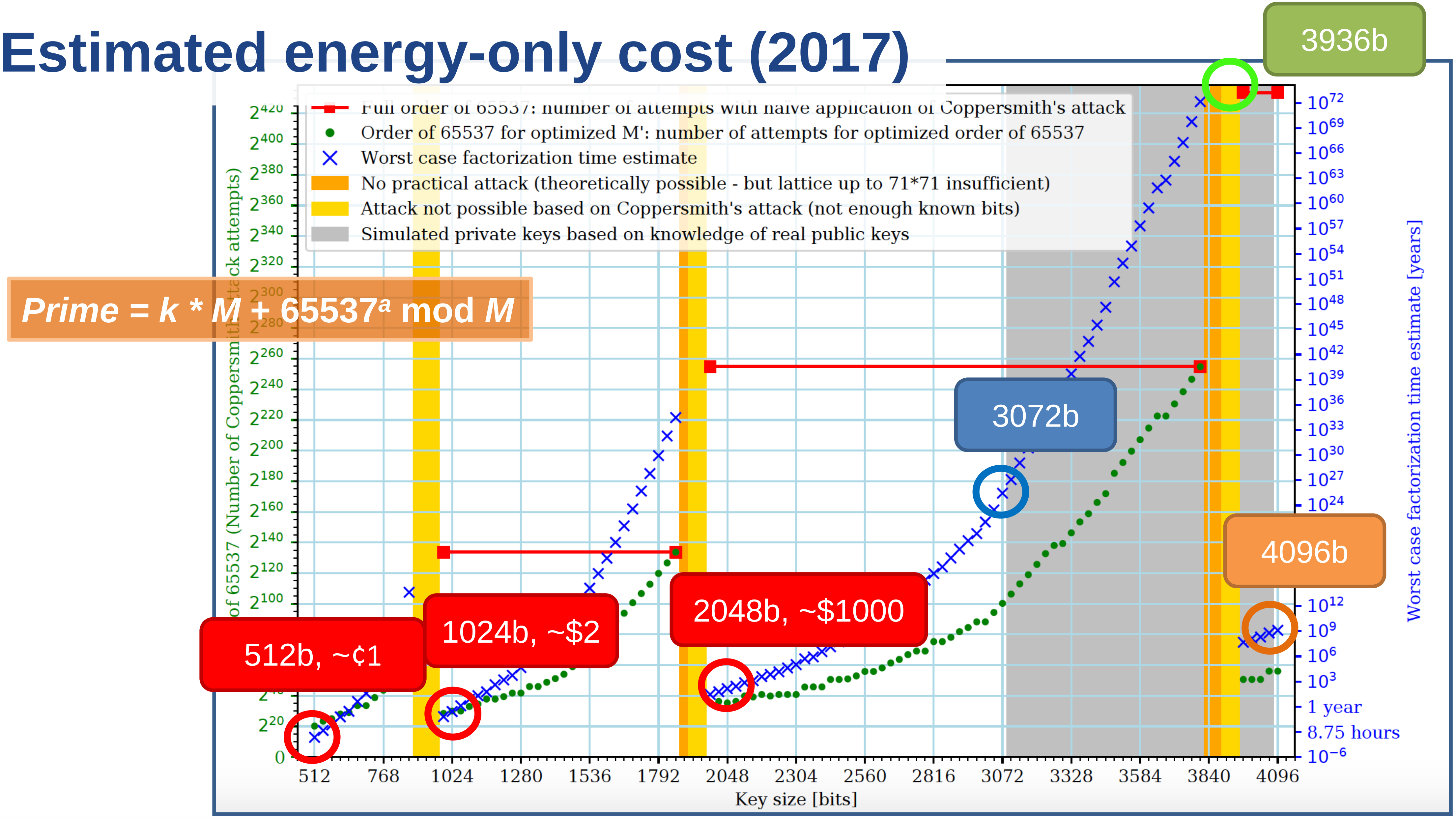
ROCA in general

Algorithmic flaw in Infineon’s RSALib (CVE-2017-15361)

- All keys generated by vulnerable Infineon library are affected
- Practical factorization of common lengths 512/1024/2048b (+ others)
  - Randomly selected 512 and 1024b keys factorized (Masaryk University)
  - Randomly selected 2048b (Estonian RIA, April 2018, “several thousands euro”)
- All public keys have unique “fingerprint” (easy to scan for)
  - Tool for detection, https://github.com/crocs-muni/roca/
- Tool for factorization (made public by Lange&Bernstein, 5th Nov)
  - Random 2048b key: 6442450944000000 vCPU years
  - Infineon 2048b key: 140 vCPU years

Attack is perfectly parallelizable
1000 cores => 1000x speedup

Estimated energy-only cost (2017)



What is the cost of an attack on RSA 2048b (year after)?

- Our paper (2017): $20,000 average price on Amazon AWS
  - Estimate: energy-only price is likely around $1000
- **Lange, Bernstein (2017) – 25% faster attack (LLL chaining)**
  - **Found in 3 days and without an access to our paper!**
- Estonian RIA (04/2018): “several thousand euros” energy price
- Our work (WAC 2018): algorithmic improvement, 2x faster
- Implementation speedups by graphic cards, FPGA…
  - Not (publicly) tested (typical speed-up factor 3-10x)

Recipients

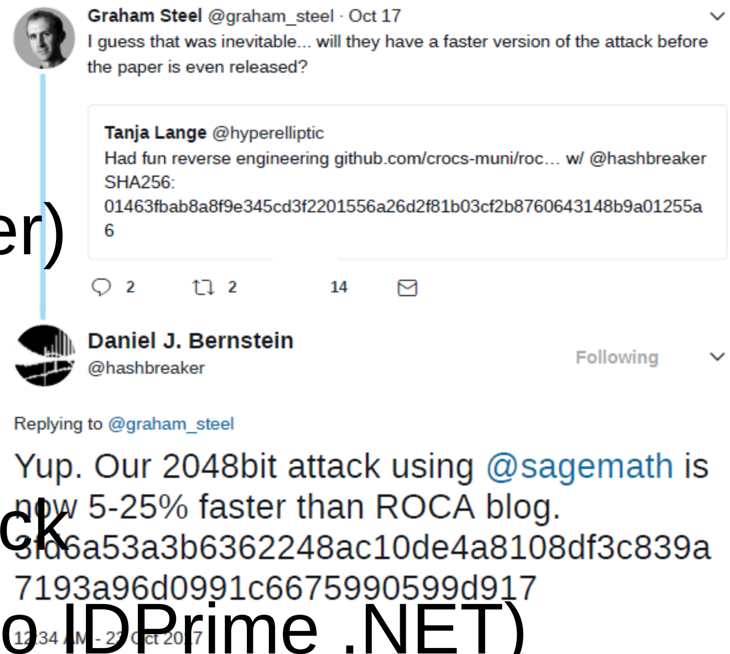
Austria, Belgium, Bulgaria, Cyprus, Czech Republic, Denmark, Estonia, Finland, France
Greece, Hungary, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Pola
Romania, Slovakia, Slovenia, Spain, Sweden, United Kingdom, Liechtenstein, Iceland,

Responsible disclosure I.

- (NIST responsible disclosure guidelines followed)
- End of January 2017: Proof of Concept attack (1024b keys factorized)
- Feb 1st: Infineon notified (email to contact at crypto group)
- Mid May: First Infineon's customers contact us back for verification
- Jun 20th: Incident report ID 163484, Austria eHealth certs revoked
  - eIDAS regulation and Article 19
  - Countries around Europe should have been notified
  - BUT: unspecific third party failure, concrete vendor named (but not Infineon)

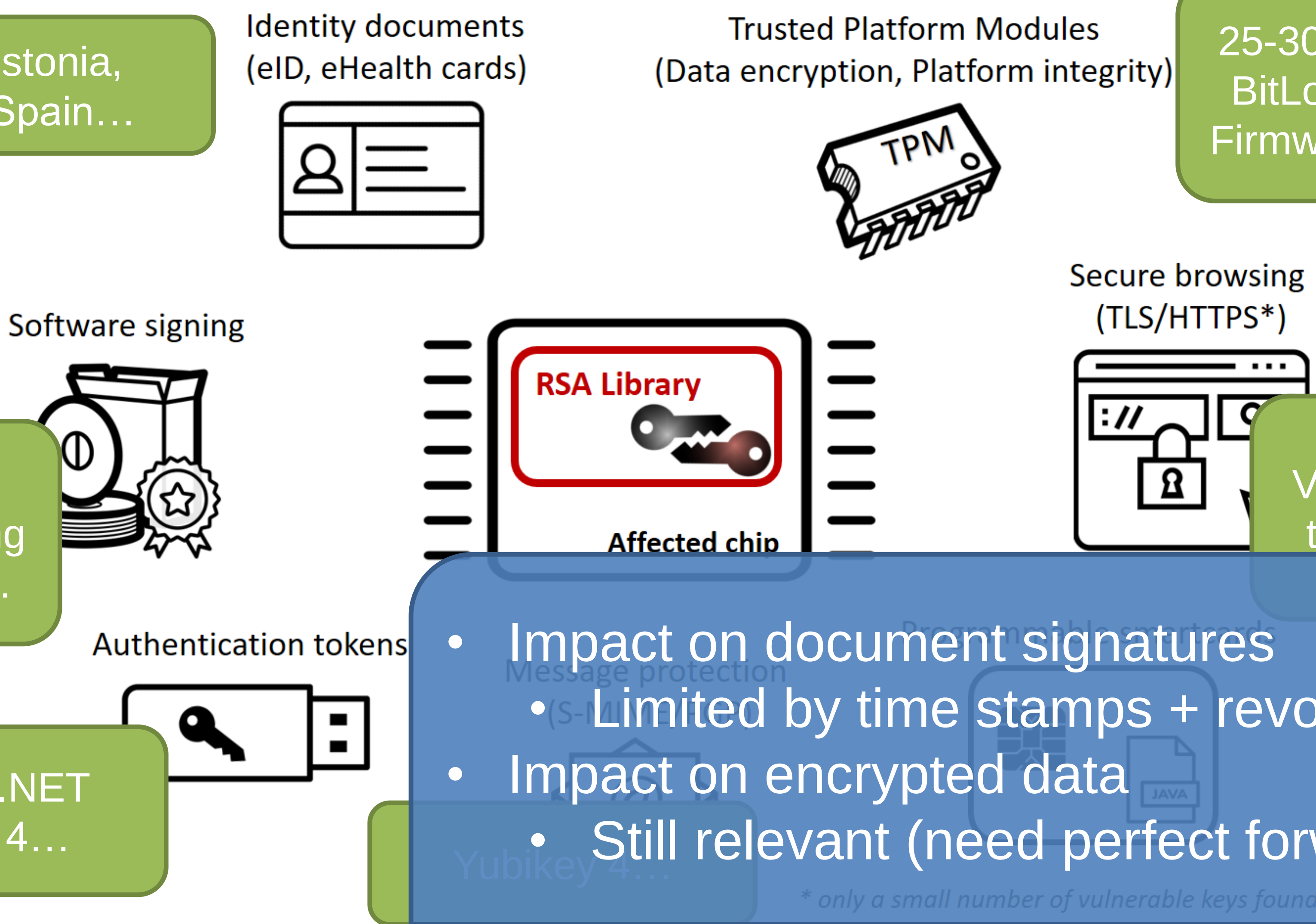
Responsible disclosure II.

- Last week Aug: vulnerable new EE certs detected (LDAP scan)
- Aug 30th: EE CERT formally contacted by us
- Sept 5th: Estonia publicly announced eID issue
- Oct 10th: Microsoft Patch Tuesday (TPMs, Bitlocker)
- Oct 16th: Public disclosure (coincide with KRACK)
  - Impact announced by us, detection tool released
- Oct 23rd: Lange& Bernstein announced faster attack
- Vulnerable devices from year 2007 found (Gemalto IDPrime .NET)
- Oct 30th: Full paper with details published (ACM CCS)
- 2/3.11. Slovakia/Estonia revokes 300k/760k certificates (10M in Spain)



M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas: The Return of Coppersmith's Attack..., ACM CCS 2017

The usage domains affected by the vulnerable library



Austria, Estonia, Slovakia, Spain…

25-30% TPMs worldwide, BitLocker, ChromeOS… Firmware update available

Commit signing, Application signing GitHub, Maven…

Very few keys, but all tied to SCADA management

Gemalto .NET Yubikey 4…

- Impact on document signatures
  - Limited by time stamps + revocation
- Impact on encrypted data
  - Still relevant (need perfect forward secrecy)

What were impacted parties typically struggling with?

- Is this attack really practical or “just” theoretical?
- How to mitigate / update already distributed cards/tokens?
  - Estonia remote update of eIDs JavaCard application (RSA $\rightarrow$ ECC)
  - Slovakia RSA 2048b $\rightarrow$ RSA 3072b
  - Yubico: free token replacement
  - Gemalto .NET auth cards?
- Is migration to 3072b safe? (BSI says ok)
- What is actually certified? (TRNG$\rightarrow$primes$\rightarrow$key$\rightarrow$use of private key)
- How to revoke large number of certificates?

Conclusions

- Certified != Secure
- Follow standards and methods!
- Every leak is problematic but
  - ½ leaked bits (Public key crypto) => complete break
- Secret design => delayed flaw discovery => higher impact
- Be prepared to revoke, patch and update everything

Questions?

Are there any positives from ROCA vulnerability?

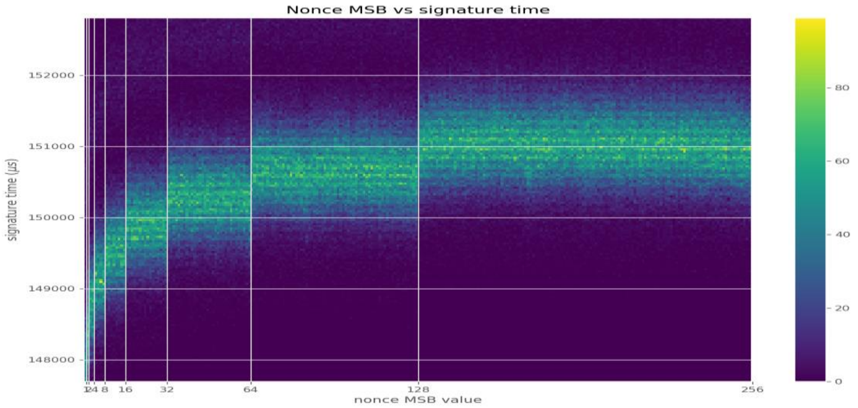
- Critical, long-present vulnerability mitigated
  – Vulnerable keys testing incorporated in administrators tools (Let's Encrypt…)
- Speed-up transition to ECC or at least longer RSA keys
- Changes to standard - verifiable RSA keypair generation from seed
- Changes to certification process - more scrutiny for key generation
- Sparked discussion about more efficient information sharing (eIDAS)
- …

Another argument for more openness and certification transparency?

Minerva vulnerability (10/2019)

https://minerva.crocs.fi.muni.cz/

- Discovered by ECTester (https://github.com/crocs-muni/ECTester)
- Athena IDProtect smartcard (EAL 4+)
  - FIPS140-2 #1711, ANSSI-CC-2012/23
  - Inside Secure AT90SC28872 Microcontroller
  - (possibly also SafeNet eToken 4300…)
- Libgcrypt, wolfSSL, MatrixSSL, Crypto+



- SunEC/OpenJDK/Oracle JDK
- Small time difference leaking few top bits of nonce
- Enough to compute whole EC private key in 20-30 min
  - ~thousands of signatures