Cryptographic Design Choices of OpenSSL Library and Their Automated Analysis

Based on analysis using pyecsca and ECTester

Łukasz Chmielewski **chmiel@fi.muni.cz**

Centre for Research on Cryptography and Security, Masaryk University

Joint work with: *Vaclav Matyas, Petr Švenda, Matúš Nemec, Marek Sýs, Dušan Klinec, Jan Jančár, Adam Janovský, Peter Sekan, Rudolf Kvašnovský, David Formánek, David Komárek, Vladimír Sedláček, Antonín Dufka and others*

CROCS

Centre for Research on Cryptography and Security

What is our approach??
CROCS 's way

- We focus on new attacks on various types of targets

1. Design technique to probe a cryptographic target
2. Implement an open-source tool for testing
3. Perform test on a wide range of targets, e.g., smartcards (usually closed-source)
4. Spot biases and develop an academically publishable exploitation method

- Ideal outcome: method can be published, real-world impact can be demonstrated, an open analysis tool available for others, and future
- Following the above, we have gained **insight** into many implementations, including **OpenSSL**, and developed multiple tools (https://github.com/crocs-muni).

Plan for this talk

ARM mbed, FlexiProvider, Nettle, GnuPG, OpenSSL, cryptlib, OpenJDK, Microsoft, wolfSSL, LibTom, Botan, Crypto++, PGP, FIPS 140-2

G&D Crypto Java Card, Infineon Crypto Java Card, Gemalto Crypto Java Card, NXP Crypto Java Card, Oberthur Crypto Java Card, Feitian Crypto Java Card

sw. libraries, smart cards…

ECDSA Timings

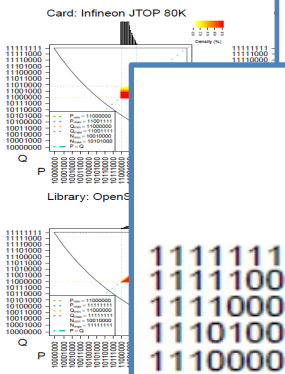
Reverse-Engineering Tooling

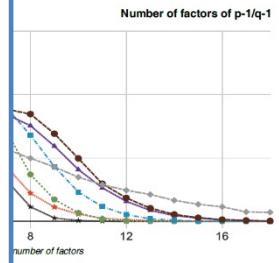
Minerva

ECTester

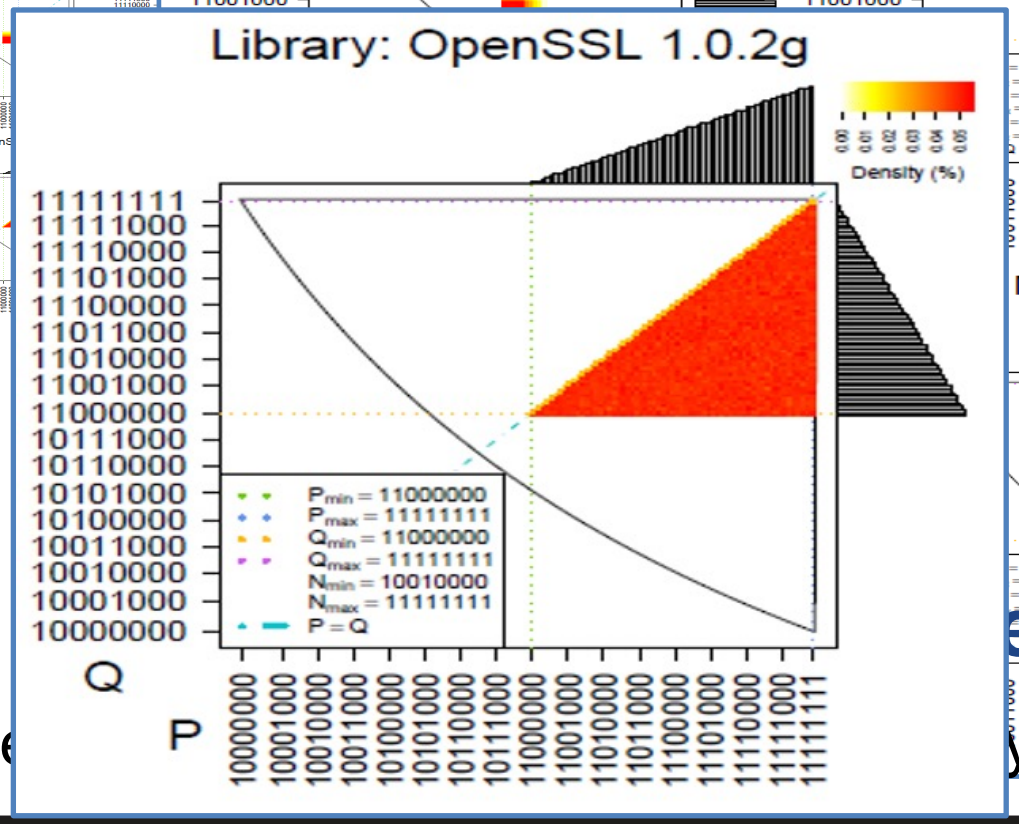
RSA Insight (from 60+ million RSA keypairs)

Distributior of factors





more…



es observable in public keys

Be y 2016

Impact (of the possibility) of public key classification

- Information leakage

- Statistics: current library usage trends

RSA key classification

- Audit: identify origin libs in the target organization

EE eID injected keys

- Forensics: source lib/device of weak keys
- Quick search for other keys from the vulnerable library

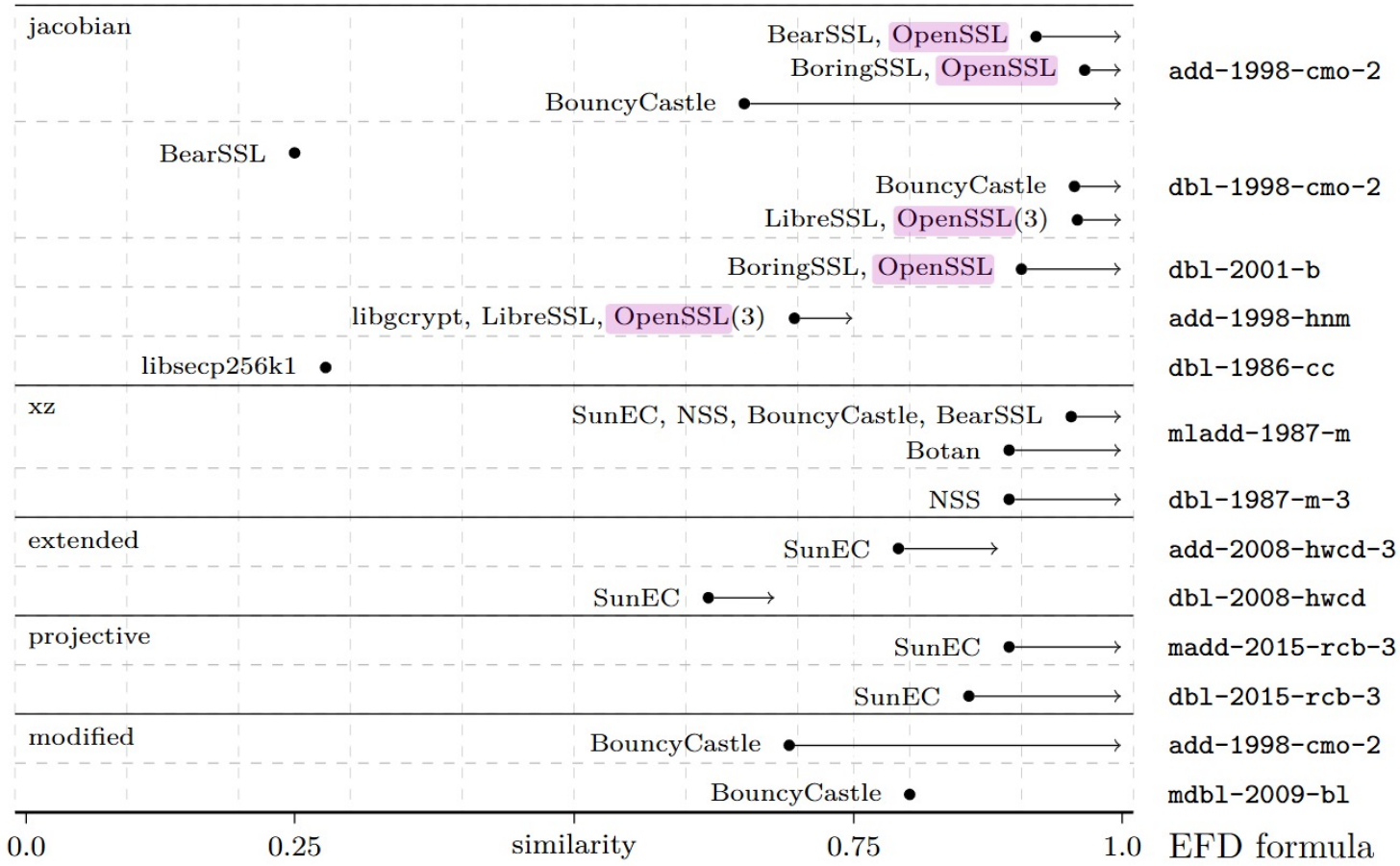
ROCA vulnerability

Insights into ECC Implementations

pyecsca

- The ECC implementations' ecosystem is a wild jungle of different implementation choices: curve models, coordinate systems, addition formulas, etc. Totals to: **139 489** imp choices.
- OpenSSL provides a variety of implementations of: ECDH, ECDSA, x25519, Ed25519. For details, see:
  - https://pyecsca.org/libraries/openssl.html
- OpenSSL has great value because it is open-source and auditable
  - Contrary to the MOST certified products

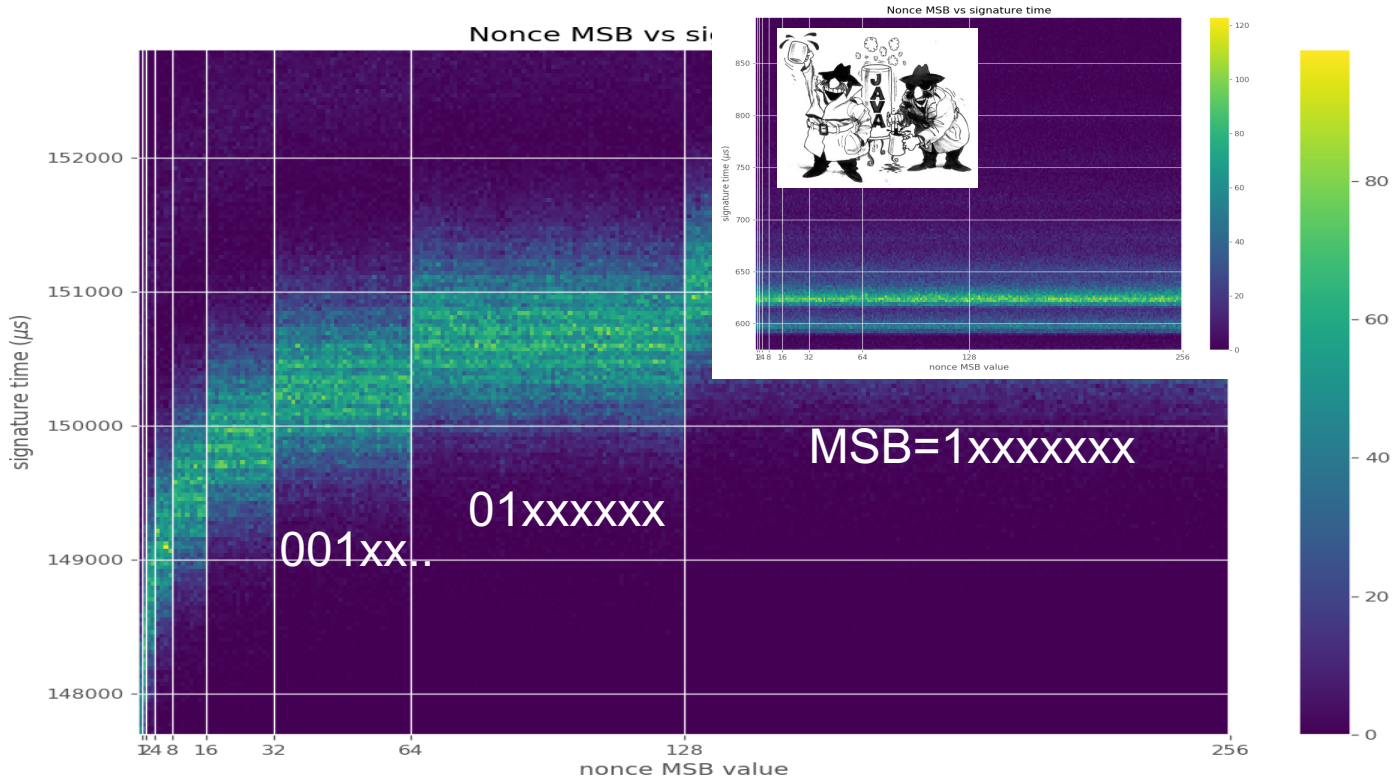
Jungle of Implementations - Example



Inside into ECC timings

Run ECC operations → MSB/time → Bias found in ECDSA?

Signature time (µs)



Minerva

EC Tester

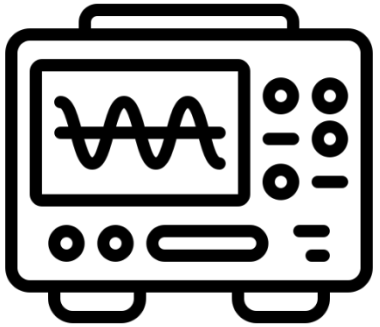
Tools for Side-Channel Analysis

We develop many tools when executing research:

- pyecsca: https://pyecsca.org/ pyecsca
- ECTester: https://github.com/crocs-muni/ECTester ECTester
- JCProfilerNext: https://github.com/lzaoral/JCProfilerNext

If you also want to use pyecsca for analysis of physical side-channel traces:

- Check the tutorial “Side-channel-based Reverse-Engineering of ECC implementations” from Ches 2024
- Setup: Docker, Binder, etc.,



https://github.com/J08nY/pyecsca-tutorial-ches2024

File Edit View Run Kernel Tabs Settings Help

jovyan@jupyter-j08ny-pyecs | implementations.ipynb | implementations.ipynb

Download GitHub Binder Code Open in... Python 3 (ipy

Filter files by name

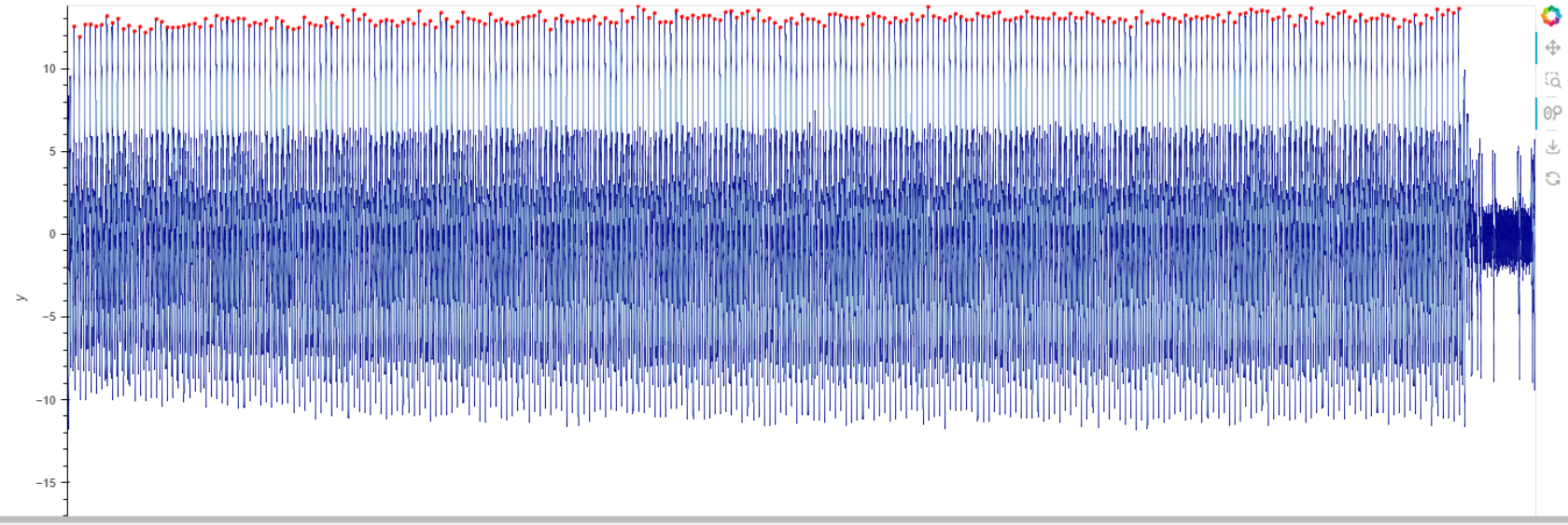
/ notebooks / solutions /

Name	Modified
implementations.i...	last yr.
rpa.ipynb	last yr.
start.ipynb	last yr.
zvp.ipynb	last yr.

```
[6]: traces_A = PickleTraceSet.read("../traces_A.pickle")

[7]: traces_A

[7]: PickleTraceSet(num_traces=30, )

[ ]: # Task: Plot two traces from the unprotected implementation.
     # Hint: Look at the .meta attribute of the traces.
     plot_traces(traces_A[0], traces_A[1])

[ ]: # Task: Plot traces from the other implementations.
     plot_traces(traces_A[10])

[ ]: plot_traces(traces_A[20])

[8]: # Task: Use rolling_mean and find_peaks to count the iterations in the unprotected implementation
     # Note: Before applying the rolling mean, make sure to transform the dtype of the trace by doing trace.astype(np.float32) and using the result.

     meany = rolling_mean(traces_A[0].astype(np.float32), 5000)
     peaks, other = find_peaks(meany.samples, height=10, distance=5000)
     print(len(peaks))
     plot_trace_peaks(peaks, meany)
```

```
254
```

[8]:



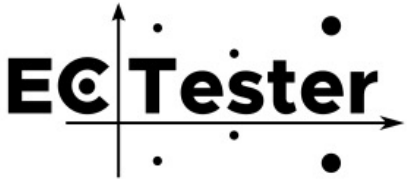
Simple 1 2 Python 3 (ipykernel) | Idle Mem: 1.79 / 2.00 GB Mode: Command Ln 2, Col 12 implementations.ipynb 1

pyecsca

Conclusions

- Automated, large-scale testing of cryptographic implementations
- Many insights resulting in the discovery of various RSA/ECC issues in a range of applications
- Importance of open-source auditable implementations
- Open tools are crucial for independent research

pyecsca.org



crocs-muni/ECTester

crocs.fi.muni.cz