PB173 Domain specific development: side-channel analysis

Seminar 10: Work in progress on main goals & how to install

Łukasz Chmielewski
chmiel@fi.muni.cz,

Consultation: A406 Friday 9:00-11:00

CROCS
Centre for Research on Cryptography and Security

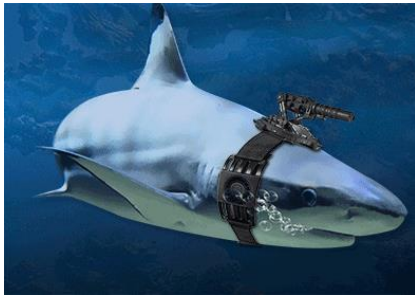
Active Side-Channel

FAULT INJECTION ATTACKS:
DIFFERENTIAL FAULT ANALYSIS ON RSA-CRT

Passive vs Active Side Channels

Passive: analyze device behavior

Active: change device behavior

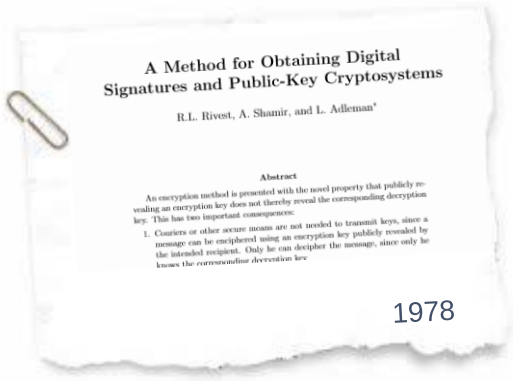


https://escooptics.com/blogs/news/world-space-week-02-lasers

Recall: RSA

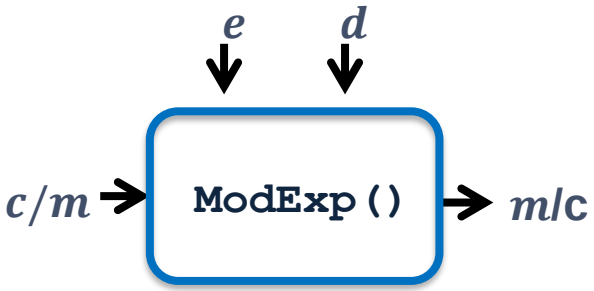
- Two primes p and q

- $N = pq$
- $\varphi(N) = (p-1)(q-1)$
- $e = 3, 5, 7, 17, 257, 65537 \rightarrow \gcd(e, \varphi(N)) = 1$
- $d = e^{-1} \bmod \varphi(N)$



Modular Exponentiation:

- Encryption / Verification: $c = m^e \bmod N$
- Decryption / Signature: $m = c^d \bmod N$



Recall: RSA Exponentiation (1)

```
ModExp(c) {
    A = 1
    for ( i = n-1; i≥0; i--)
        A = A^2 mod N
        if (d_i = =1)
            A = A*c mod N
        end if
    end for
    return A = c^d mod N
}
```

$d=(101)=5$

$A = 1,$

$d_2=1$

$A = A^2 \bmod N=1$

$A = A*c \bmod N=c$

$d_1=0$

$A = A^2 \bmod N=c^2$

$d_0=1$

$A = A^2 \bmod N=c^4$

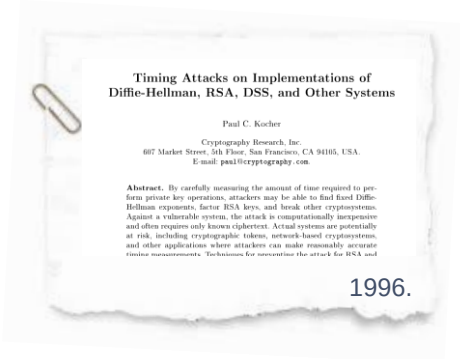
$A = A*c \bmod N=c^5$

Recall: Simple Power Analysis on RSA

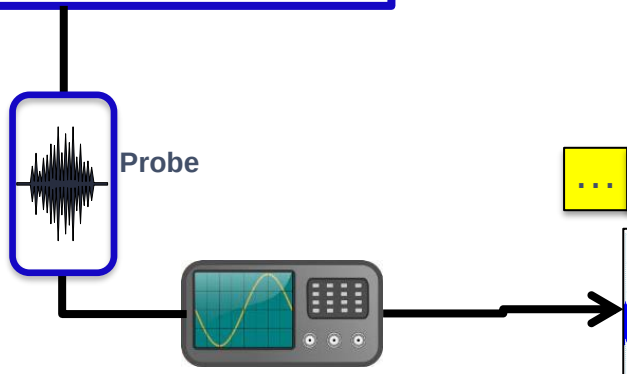
```
ModExp(c) {
    A = 1
    for (i = n-1; i≥0; i− −)
        A = A² mod N
        if (dᵢ = =1)
            A = A*c mod N
        end if
    end for
    return A = c^d mod N
}
```

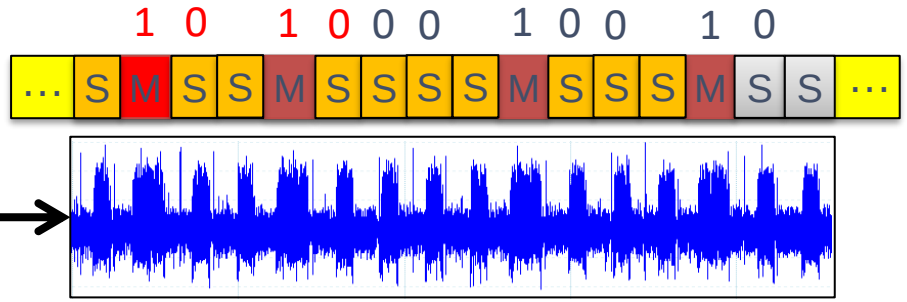
S

M



"By carefully measuring the *amount of time* required to perform private key operations, attackers may be able to find […] RSA keys."





RSA in practice: RSA-CRT

- Optimization of computing a signature giving about 3 or 4-fold speed-up
- Precompute the following values:
  - Find $d_p = d \pmod{p-1}$, computed as $d_p = e^{-1} \pmod{p-1}$
  - Find $d_q = d \pmod{q-1}$
  - Compute $i_q = q^{-1} \pmod{p}$
- Computations using $m_p = m \pmod{p}$ and $m_q = m \pmod{q}$
- Signature or encryption (forgetting about hashing):
  - $s_p = m^{d_p} \pmod{p}$ 
  - $s_q = m^{d_q} \pmod{q}$ 
  - Garner's method (1965) to recombine s_p and s_q:
    - $s = s_q + q \cdot (i_q(s_p - s_q) \pmod{p})$

- Let us see the slides from Seminar 1!

ORGANIZATIONAL

Final Division

- Group 1: Michal, Matus, Filip (?)
  - Topic: Align
  - GitHub repository: https://github.com/mr-akiio/trs-alignment
- Group 2: Michael T, Lubomir, Richard
  - Topic: Standard Processing, Michael might touch also “Parallel computations with acquisition”
  - The group is 3 people since Vendelín left.
  - GitHub repository: https://github.com/LubJur/PB173_standard_signal_processing
- Group 3: Tomas Re, Tomas Ro, Martin
  - Topic: Visualization
  - GitHub repository: https://github.com/reznakt/pb173-sca-visualization

(Modified) Seminars Plan

- 7: today, no points
- 8: evaluation of first steps given last week: 3 points per group (personalized per person based on Github activity) + Giving new tasks
- 9: Checking Progress: helping & trying to run your tools
- **10: 3 points per group (personalized per person based on GitHub) + a short 5-10minuts progress presentation + demo (1 point) + Giving new tasks**
- 11: Checking Progress **[Online]**
- 12: Final seminar: **final short 5-10minuts presentation (1 point) & grading** + grading (**3** points for final tasks) + 2 points for activity.

Short Presentations: 5min + demo

- What are you solving?
- What language or libraries are you using?
- What do you have now?
- What are you going to do?

- Short demo

PRESENTATIONS BY GROUP NUMBER (GRADING THE PRESENTATION)

WHAT WAS DONE + GIVING NEW TASKS (GRADING THE WORK)

Group 1: Installation

- No installation or examples; please add something!

README.md

Traces Alignment

This project is trying to align traces by using peak-based alignment and correlation-based alignment.

Each python file will create new file with changed traces, with appropiate naming (usualy as +file_name). On the beggining of each python file, there are some global variables you can change. To see the traces use `visualize.py`.

Run in terminal as:

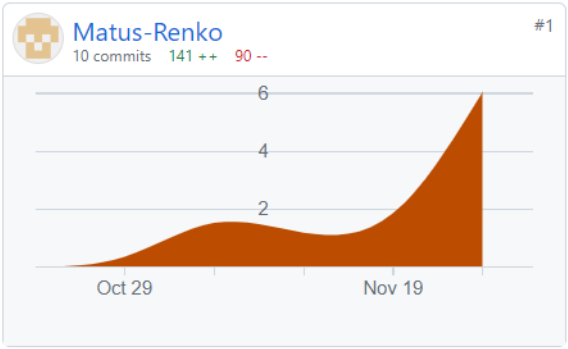
> python file.py traces.trs

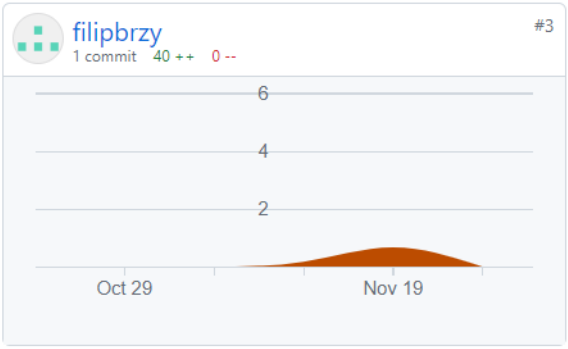
You can find here already two files with misalign traces. One contains 5 traces and other 10. You can run them trough `peak_alignment.py` or `cross-correlation.py` to see the results.

There is one more file with 10 traces. These are not misalign and you can do with them what you want. You can use two more python files to work with traces.

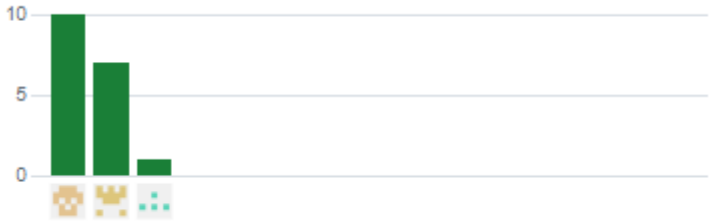
- `misalign.py` will misalign traces it gets as parameter.
- `saveAs.py` will save number of traces you will set in global variables

Group 1: Work Distribution





Excluding merges, **3 authors** have pushed **18 commits** to main and **18 commits** to all branches. On main, **11 files** have changed and there have been 704 **additions** and 1 **deletions**.



Group 1: Main Goals

- Main Tasks:
  - Test more peak-based alignment
  - Correlation-Based Alignments
  - Improve Efficiency
  - Two from:
    - Trace alignment algorithm for suppressing the clock jitter, see pages 45-50 of: https://ged.biu-montpellier.fr/florabium/jsp/win_main_biu.jsp?nnt=2014MON20039&success=%2Fjsp%2Fwin_main_biu.jsp&profile=anonymous
    - Elastic alignment algorithm or
    - Round Based Alignment

Group 1: What to do next

- TODO together

Group 2: Installation

README.md

In this repository, there should be a lot of functions, that handle many standard signal processing.

To use this functionality you need to download the python library trsfile on github. You can find more information about this library at this link: https://github.com/Riscure/python-trsfile.

Call the script using "python main.py {method} {file_path}" we now support only format .trs, but we will maybe expand our possibilities. In case you are lost, you can always use "python main.py print_commands" to get all the available methods.

Sources

https://numpy.org/doc/stable/index.html

https://trsfile.readthedocs.io/en/master/

RUN

Copy this code to terminal to run all the requirements

pip install trsfile

pip install typer

pip install matplotlib

pip install numpy

To run the application itself, run in terminal:

python3 main.py [used script] [trace filename] (--visualize)

(--visualize is optional parameter) for example like this

python3 main.py standart-deviation test_traces.trs --visualize

We also support multiple operations all at once, just be careful with what type do you work and that the types fit, otherwise it is undefined behaviour.

To run multiple operations use command like this

python3 main.py multiple-options absolute,average --visualize

(traces in this command are set for default = test_traces.trs)

TODO

D = Designed IP = In Progress QA = Done, testing

spectral intensity

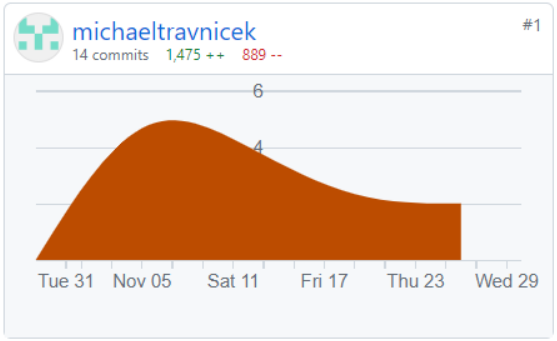
first: easy modules - average (QA) - standard deviation (QA) - histogram (IP) - absolute value (QA) - Band-pass filter (D) - Signal-To-Noise Ratio and others metrics (D) look at SaveAs.py and correlation.py

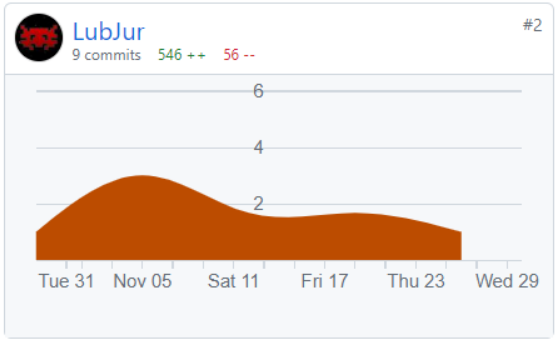
try implementing computing spectrum

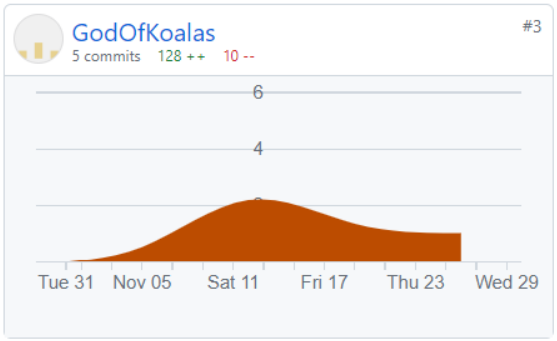
first create computing spectrum png with fft, like in presentation (heatmap)

average which saves it to a file

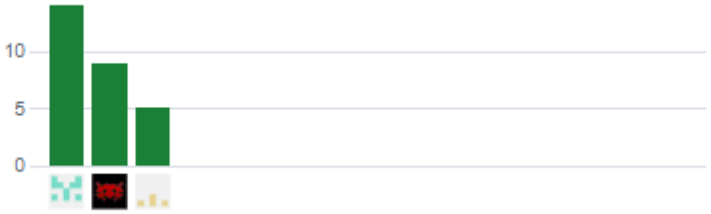
Group 2: Work Distribution







Excluding merges, **3 authors** have pushed **28 commits** to main and **28 commits** to all branches. On main, **0 files** have changed and there have been **0 additions** and **0 deletions**.



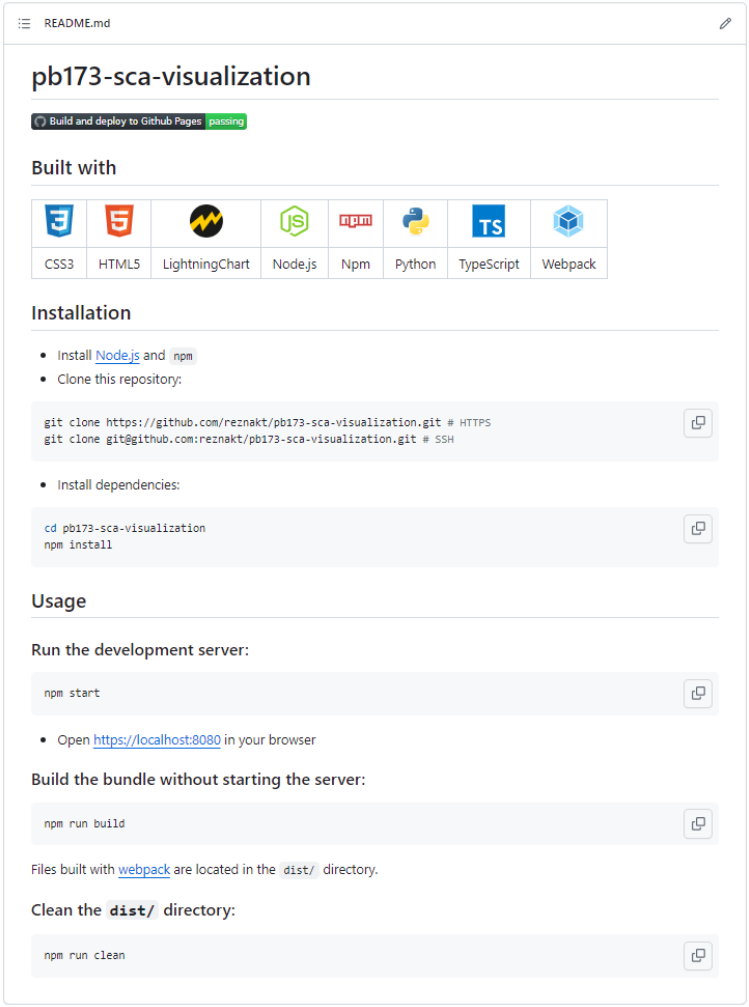
Group 2: Main Goals

- Main Tasks:
  - Standard Deviation, Average, FFT
  - Spectrogram
  - Incremental Correlation: https://eprint.iacr.org/2022/253.pdf
  - Pipelining
  - Signal-To-Noise Ratio and other metrics

Group 2: What to do next

- TODO together

Group 3: Installation



README.md

pb173-sca-visualization

Build and deploy to Github Pages passing

Built with

CSS3 | HTML5 | LightningChart | Node.js | Npm | Python | TypeScript | Webpack

Installation

- Install Node.js and `npm`
- Clone this repository:

```
git clone https://github.com/reznakt/pb173-sca-visualization.git # HTTPS
git clone git@github.com:reznakt/pb173-sca-visualization.git # SSH
```

- Install dependencies:

```
cd pb173-sca-visualization
npm install
```

Usage

Run the development server:

```
npm start
```

- Open https://localhost:8080 in your browser

Build the bundle without starting the server:

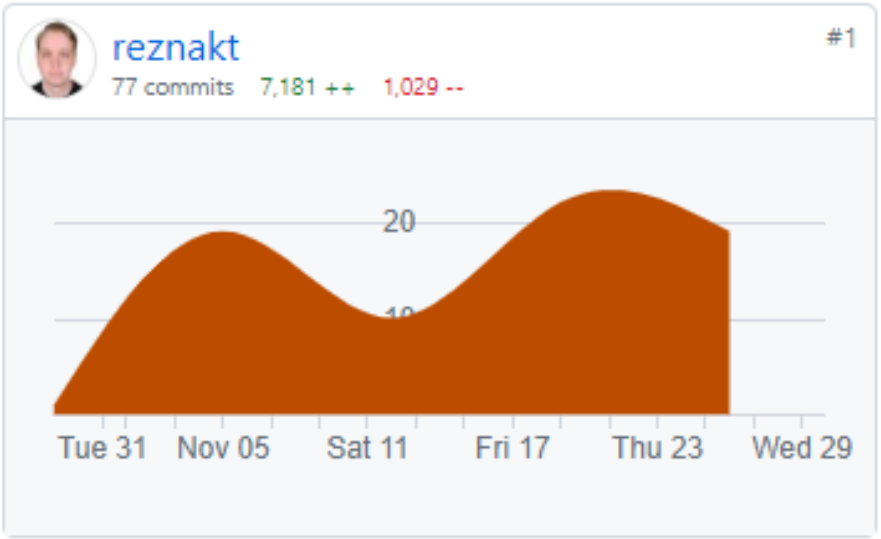
```
npm run build
```

Files built with webpack are located in the `dist/` directory.

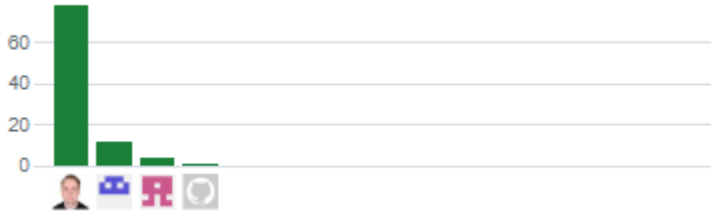
Clean the `dist/` directory:

```
npm run clean
```

Group 3: Work Distribution



Excluding merges, **4 authors** have pushed **89 commits** to main and **95 commits** to all branches. On main, **0 files** have changed and there have been 0 **additions** and 0 **deletions**.



Group 3: Main Goals

- Main Tasks:
  - Displaying Traces
  - Moving traces around?
  - Selecting part of the trace to run something (any code)?
  - Comparison to other libraries

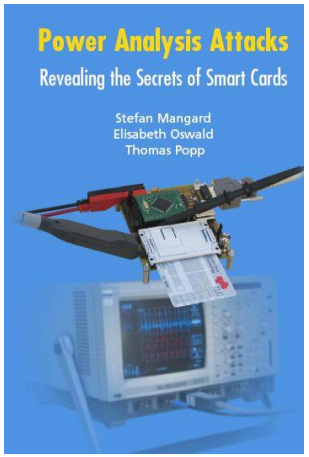
Group 3: What to do next

- TODO together

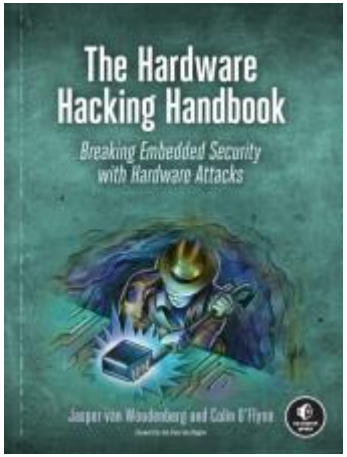
WALK-AROUND & HELPING & DISCUSSIONS

Reading

- For interested people
- Side-Channel Analysis – blue book:
  - http://dpabook.iaik.tugraz.at/
  - The books is available at the uni.
  - Look online



- The Hardware Hacking Handbook:
  - https://nostarch.com/hardwarehacking
  - I have an epub version.



Questions