Improving Side-Channel Resistance of Java Card Implementations

Author: **Mgr. Veronika Hanulíková**, Supervisor: **doc. RNDr. Petr Švenda, Ph.D.**

Faculty of Informatics, Masaryk University

Thesis Contributions

Java Card [1] applets are Java-based applications that are run in the **secure environment** provided by **smart cards**. Still, these devices remain susceptible to **timing side-channel attacks** exploiting execution time differences caused by variations in control flow, memory access, or computations to leak sensitive information. The thesis proposes a methodology for **evaluation and detection of timing leaks** along with precise **time profiling** based on Simple Power Analysis (SPA) for Java Card applets. Furthermore, it explores **countermeasures for mitigating timing side-channel** on the Java Card platform, applied to the reimplementation of the JCMathLib library.

Timing Side-Channel on Java Card Platform

The Java Card applets are written in a subset of the Java programming language. The instruction set in Java Card technology is called bytecode, which is executed by the Java Card Virtual Machine (JCVM) [1].

Types of Timing Leaks on Java Card Platform

- **Java code level leakage**: Leaks can originate from the Java code structures used in the Java Card applet. The control flow of the implementation can execute different sequences of bytecode instructions based on the sensitive data, resulting in execution time deviations.
- **Java bytecode level leakage**: The leak is formed from the bytecode instruction internally processed by the JCVM, possibly depending on the data or the context of its execution.
- **Hardware level leakage**: Timing deviations can be formed during the execution of native methods or low-level instructions by the card's embedded CPU.

The thesis targets **Java code level leakage** preventable by constant-time programming techniques.

Java Card Time Profiling with JCProfilerNext

The JCProfilerNext [2] is an existing tool for profiling Java Card applets. The measurements and visualizations provided by the tool can be employed to evaluate potential side-channel leakage on the card. The timing analysis of the applet relies on time measurements obtained using a standard timer on the host PC connected to the reader with the smart card.

Power Analysis for Time Profiling

The thesis introduces a new profiling mode for the JCProfilerNext tool by integrating **Simple Power Analysis (SPA)**. This approach aims to reduce the impact of noise and measurement inaccuracies.

The SPA timer mode uses delimiters to separate individual operations within the applet. They are inserted between specific lines of code in the measured Java Card applet method. These delimiters generate recognizable patterns in the power trace, allowing for accurate automatic detection of their boundaries.

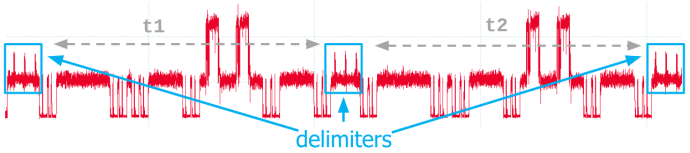

Figure 1. Power trace with delimiters in between operations with execution times $t1$ and $t2$, captured via PicoScope 7 software [3].

The host PC communicates with the card via a **LEIA board** [4]. The LEIA board is connected to the **PicoScope Oscilloscope** [3]. JCProfilerNext establishes communication with the smart card using a custom Java driver. Then, it configures the relevant channels for power measurement and trigger synchronization. After the trace is acquired, the **SPA Cryptographic Operations Extractor** [5] tool searches for the delimiters in the measured power trace.

Fuzzing Applets for Leak Detection

Fuzzing is a dynamic testing technique that generates random or malformed inputs to trigger unexpected behavior or reveal vulnerabilities of the tested code. The thesis employs the **DifFuzz tool** [6], which enables the automatic detection of timing side-channel leaks through **differential fuzzing**. The strategy compares execution behavior across inputs, flags those causing different code paths, and reports them as potentially leaking.

Fuzzing of applets utilizes a **Java Card Development Kit Simulator** [1] for running an applet on a host PC. The two-stage approach uses an instrumented applet on the simulator to identify problematic inputs that are subsequently tested on a physical smart card for actual timing.

Proposed Methodology for Time Leakage Evaluation of Java Card Applet

1. **Code analysis** assesses the implementation, locates sensitive data handling, and defines potential problematic methods.
2. **Identification of potentially leaking inputs** is done by observing random data, performing manual analysis, or differential fuzzing on the simulator.
3. **Time profiling** is performed using JCProfilerNext and SPA timer mode on the physical card.
4. **Statistical testing** by the *tlsfuzzer* project [7] determines if the implementation leaks timing information by comparing distributions across input classes.
5. **Leak quantification** computes the amount of time leak in data using the probability density function.

Java Card Constant-Time Implementation Techniques

The techniques below introduced by the thesis present key strategies for removing execution time dependencies on sensitive data processed in code:

- **Bitwise operations** are used to compare and combine the outcomes of multiple operations using bit masks.
- **Conditional statements** must be modified to perform operations regardless of input.
- **Number of iterations** in `for` and `while` constructs should not depend on sensitive data.
- **Memory handling** must be performed uniformly.
- **Error reporting** should be handled with the return values propagation, while exceptions can be used for unrecoverable errors.

References

[1] Oracle Corporation. *Java Card Platform*. URL: https://www.oracle.com/java/java-card/.
[2] Lukáš Zaoral. "Automatic Performance Profiler for Security Analysis of Cryptographic Smart Cards". Master's thesis. Brno: Masaryk University, Faculty of Informatics, 2023. URL: https://is.muni.cz/th/v7l30/.
[3] Pico Technology Ltd. *PicoScope 4000 Series Programmer's Guide*. Revision 10. 2022. URL: https://www.picotech.com/download/manuals/picoscope-4000-series-programmers-guide.pdf.
[4] Ryad Benadjila et al. "LEIA: The lab embedded ISO7816 analyzer a custom smartcard reader for the ChipWhisperer". In: *SSTIC2019* (2019), p. 30.
[5] Martin Podhora. "Forensic profiles of certified cryptographic smartcards". Master's thesis. Brno: Masaryk University, Faculty of Informatics, 2022. URL: https://is.muni.cz/th/huike/.
[6] Shirin Nilizadeh, Yannic Noller, and Corina S. Pasareanu. "DiffFuzz: Differential Fuzzing for Side-Channel Analysis". In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 2019, pp. 176–187. DOI: 10.1109/ICSE.2019.00034.
[7] *tlsfuzzer: SSL and TLS Protocol Test Suite and Fuzzer*. URL: https://github.com/tlsfuzzer/tlsfuzzer.

hanulikova@mail.muni.cz