Side-Channel Finder for AVR

Florian Dewald, Heiko Mantel, Alexandra Weber (MAIS, TU Darmstadt)

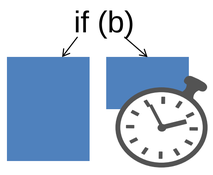
CHES 2018, Poster Session, Amsterdam, The Netherlands, September 9–12, 2018

Motivation

Side Channels

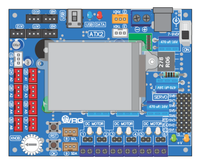
- unintended communication channels, e.g., running time, power consumption, …
- can reveal secrets, e.g., crypto keys
- timing side channels are a dangerous subclass of side channels
  because they can be exploited remotely [Brumley/Boneh,ComputerNetworks2005]

```
public int modExp(int c, int k) {
  int r = 1;
  for (int i = 0; i < 32; i++) {
    if (k % 2 == 1) r = (r*c) % N;
    c = (c * c) % N;
    k >>= 1; }
  return r % N; }
```



Attacks on Devices in the Internet of Things

- Internet of Things (IoT) devices are attractive targets for attacks
- recently, smart light bulbs were compromised
  [Ronen/O'Flynn/Shamir/Weingarten,S&P2017]
- the vulnerable light bulbs contain an AVR processor,
  a popular processor in IoT devices



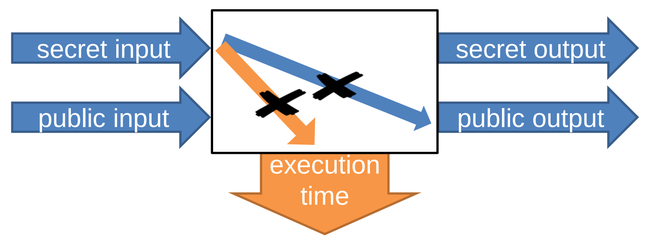
Our Focus

- systematic detection of timing side channels in AVR programs

Approach

Timing-Sensitive Noninterference (TSNI)

- two program executions that start in attacker-indistinguishable states
  take the same time and the final states are attacker-indistinguishable



Timing-Sensitive Security Type System

- assign security types to registers, memory, and stack (secret or public)
- type system only allows to type a program if
  - the program is free of timing side channels, and
  - there is no direct or indirect information flow
- soundness: all typeable programs satisfy timing-sensitive noninterference
- precision: secret-dependent branches are allowed if the branches take the same time

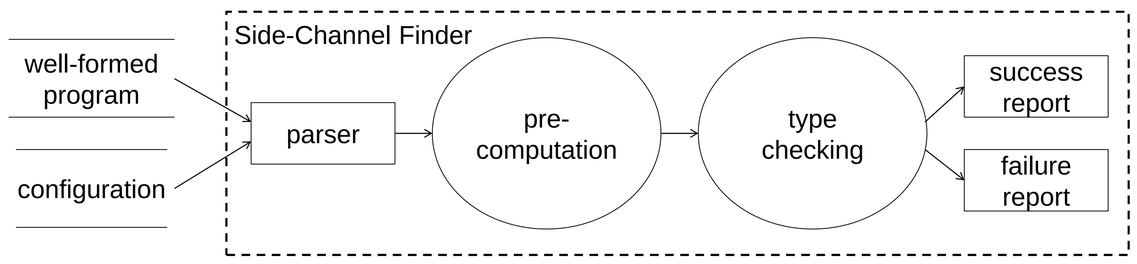
Faithful Formal Semantics for AVR Assembly

- operational semantics that faithfully captures the execution of instructions
- in particular, exact timing (clock cycles) of instructions is modelled faithfully
  - based on vendor specification in instruction set manual
- soundness proof is with respect to this semantics

Tool Support

Side-Channel Finder (SCF)

- takes a well-formed AVR program
  - syntactically correct, single return instruction, in supported sublanguage
- takes a configuration
  - specification of secret and public inputs and outputs
- checks for absence of timing side channels using the timing-sensitive security type system
- implemented in roughly 1250 lines of Python code



Reports to the User of Side-Channel Finder

- success report if program is typeable
  - program satisfies TSNI, i.e., is free from timing side channels
- failure report if program is not typeable
  - shows the reason for failure, e.g., secret-dependent loop
  - points to location of potential vulnerability in the code

Case Studies

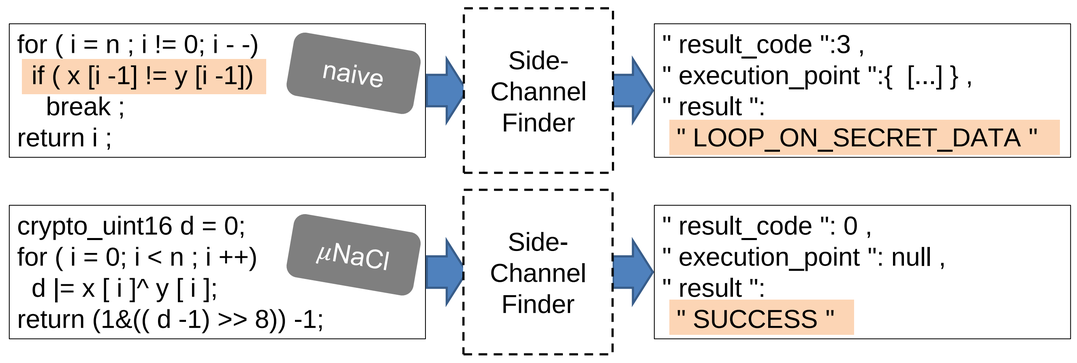
Crypto Implementations from μNaCl

- μNaCl is a crypto library for AVR microcontrollers
  [Hutter/Schwabe,AFRICACRYPT2013]
- Side-Channel Finder was applied to multiple crypto implementations from μNaCl
  - Salsa20 (stream cipher)
  - XSalsa20 (variant of Salsa20 with longer nonces)
  - Poly1305 (message authentication code)

Verification Results for μNaCl

- Side-Channel Finder successfully verified that the crypto implementations satisfy TSNI,
  i.e., the implementations are secure against timing side channels

Example of SCF Output: Analysis of String Comparison



More Information

Researchers

F. Dewald H. Mantel A. Weber

Publication and Supplementary Material

- Florian Dewald, Heiko Mantel and Alexandra Weber.
  AVR Processors as a Platform for Language-Based Security.
  ESORICS, pages 427-445, 2017
- Appendix with formalizations and proofs available online
  - faithful operational semantics of AVR assembly
  - timing-sensitive security type system
  - soundness proof
- Side-Channel Finder tool with case studies available online

SCAN THIS CODE
and download
Side-Channel Finder

Ongoing Work

- improving Side-Channel Finder, e.g., by extending the scope to the full 8-bit AVR instruction set

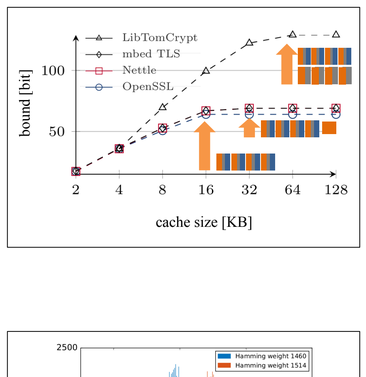
Further Side-Channel Research at MAIS

Our Vision for Side-Channel Security

- traditionally: attack-driven detection and often reactive mitigation of side channels
- our vision: systematize the detection and mitigation of side channels
- complement the attack-driven approach with program analysis

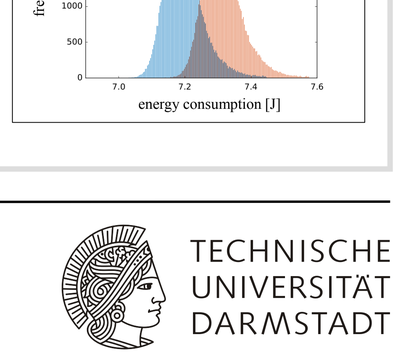
Quantitative Program Analysis

- approach based on reachability and information theory
  - e.g., systematic study of cache side channels across
    AES implementations
    [Mantel/Weber/Köpf,ESSoS2017]
  - e.g., detection and mitigation of 4 cache side channels
    in the implementation of the lattice-based signature
    scheme ring-TESLA
    [Bindel/Buchmann/Krämer/Mantel/Schickel/Weber,FPS2017]



Systematic Distinguishing Experiments

- approach based on statistical methods and information theory
  - e.g., clarification of the security implications of green IT
    (in the domain of software-based energy side channels)
    at the example of BouncyCastle RSA and Intel RAPL
    [Mantel/Schickel/Weber/Weber,ESORICS2018]
  - e.g., clarification of security-performance-tradeoff
    for program transformations against timing side channels
    [Mantel/Starostin,ESORICS2015]



www.mais.informatik.tu-darmstadt.de