On the Effectiveness of the Remanence Decay Side-Channel to Clone Memory-based PUFs

Christian Wachsmann
christian.wachsmann@trust.cased.de
Intel CRI-SC at TU Darmstadt, Germany

Joint work with:

Ahmad-Reza Sadeghi
Technische Universität Darmstadt / CASED
Germany

Yossef Oren
Tel Aviv University
Israel
Memory-based Physically Unclonable Functions (PUFs) often considered as lightweight alternative to secure non-volatile memory

Typical assumptions on memory PUF-based systems

- Reading out the secret PUF state is hard
- Re-use of existing device memory minimizes implementation costs

We show: Re-use of device memory allows reading out secret PUF state
In This Talk

• Cloning attack against memory-based PUFs
  • Exploits data remanence decay as side-channel
  • Applies differential fault analysis [Biham and Shamir, CRYPTO’97] to extract secret PUF state

• Experimental and practical validation of the attack

• Countermeasures
What is a Memory-Based PUF?
Major class of PUFs based on instability of volatile memory

Such as SRAM cells, flip-flops or latches

Our focus: SRAM-based PUFs

Goal: Extract unique device-specific fingerprint
SRAM block
(array of SRAM cells)

SRAM cell: pair of cross-coupled inverters
• Inverters designed identically
• Identical inverters mean state 0 and 1 is equiprobable at power-up (when bit lines are undefined)

Manufacturing variations affect properties of inverters
• Most cells are biased towards 0 or 1 at SRAM power-up
• Some cells are metastable (take 0 or 1 with equal probability)
What are Memory-Based PUFs used for?
Typical Application: Secure Key Storage

Common assumptions

- PUF response can only be read by post processing algorithm
- Post processing and security mechanism do not leak key or PUF response

These assumptions are not sufficient!
Why are these assumptions insufficient?
SRAM stores data

Power off ⇒ Data slowly decays to PUF state

Power on ⇒ Decay stops
How to turn this into an attack?
Fault Injection Attack

Assumptions

- Adversary knows value written to the SRAM
- Adversary controls power supply of device
- Adversary can observe device behavior (e.g., a device response)

Adversary can force the security mechanism to use a wrong key that depends on a partially known memory state.
How to exploit this to extract the secret PUF state?
Differential Fault Analysis

[Biham and Shamir, CRYPTO’97]

Two phases:

1. Data Collection Phase
   Observe and record device behavior for different partially known memory states

2. Analysis Phase
   Recover secret PUF state in a step-by-step fashion
Data Collection Phase

Increase power-off time in each experiment

Record the device behavior for each power off time

Response $X_0$

Response $X_1$

Response $X_2$

Response $X_f$

Recorded Device Responses

$\{X_0, X_1, X_2, \ldots, X_f\}$
Analysis Phase

Recorded Device Responses \( \{X_0, X_1, X_2, \ldots, X_f\} \)

Exhaustive Search

- \( X_0 \)
- \( X_1 \)
- \( X_2 \)
- \( X_f \)

Requirement: Difference between two consecutive memory states must be small
Does this work in practice?
PUF ASIC

- ASIC manufactured in TSMC 65 nm CMOS multi-project wafer run
- Includes four 8Kbyte SRAM-PUFs (amongst other PUF types)

Test setup

- Workstation
- FPGA
- Evaluation Board with PUF ASIC
- Pulse Generator
- Control and PUF Data
- Control
- ASIC Supply Voltage
Each SRAM cell has a characteristic decay time

Careful control of power-off time minimizes number of bit-changes between two consecutive experiments
What about real systems?
Effectiveness Against Real System

• **Target system: PUF key storage and authentication scheme**
  - 8 KByte SRAM used as PUF
  - Uses repetition code and linear encoding [Bösch et al., CHES’08]
  - Generates 128 bit key from PUF response
  - Key used in standard challenge/response authentication protocol

• **Attack complexity**
  - 128 bit key stored in PUF can be recovered with $\approx 2^{56}$ operations
  - Key recovery can be parallelized
How to prevent the attack?
Countermeasures

**Use dedicated read-only SRAM for the PUF**
- Contradicts idea of using existing memory for lightweight implementations
- Not suitable for low-end embedded devices (e.g., sensors)

**Wait until all memory cells have returned to PUF state**
- Takes considerable amount of time
- Decay-time depends on operating conditions (e.g., temperature)

**Obfuscate device behavior**
- Seems to increase complexity of the algorithms and protocols
- May exceed capabilities of low-end embedded devices (e.g., sensors)
Conclusion and Future Work

We presented

• First non-invasive cloning attack against memory-based PUFs based on the data remanence decay side channel
• Experimental and practical validation of the attack
• Performance improvement of TARDIS time-keeping mechanism for clock-less devices [Rahmati et al., USENIX’12] (see paper for details)

Current and future work

• Improving the attack
  • More precise control of decay effect (use voltage-based approach)
  • Optimize analysis phase (exploit properties of PUF post processing algorithms)
Thank you!

Christian Wachsmann
christian.wachsmann@trust.cased.de