



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

# Run-time Accessible DRAM PUFs in Commodity Devices

Wenjie Xiong<sup>1</sup>, André Schaller<sup>2</sup>, Nikolaos A. Anagnostopoulos<sup>2</sup>,  
Muhammad Umair Saleem<sup>2</sup>, Sebastian Gabmeyer<sup>2</sup>,  
Stefan Katzenbeisser<sup>2</sup>, and Jakub Szefer<sup>1</sup>

1. Yale University, USA

2. Technische Universität Darmstadt and CASED, Germany

Aug 18, 2016

# Physically Unclonable Functions (PUF)

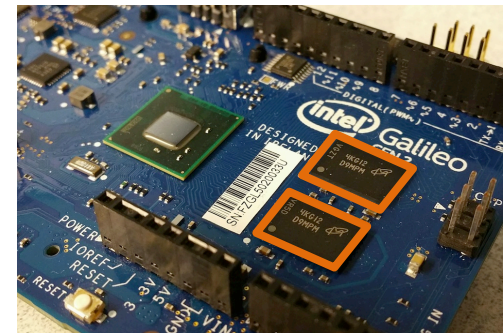
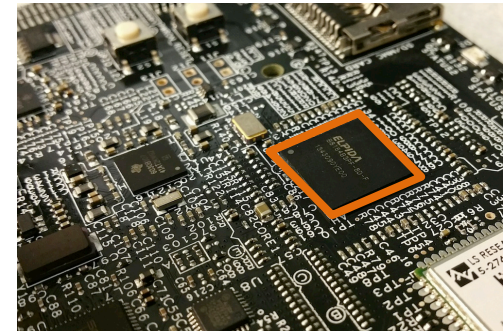
- A function, which is embedded into a physical object  
When queried with a challenge  $x$ ,  
the PUF generates a response  $y$ , which depends on  
1) Challenge  $x$  and 2) special physical properties of the object
- Silicon PUFs use the manufacturing process variations  
e.g. Arbiter PUFs, SRAM-PUFs  
It is almost impossible to clone, even for the manufacturer
- Authentication and identification

# Intrinsic DRAM PUF

- No extra chips needed for PUF
- Exploit hardware which is on-board anyway  
e.g. start up values of SRAM

Is it possible to exploit DRAM as a PUF?

- Most computing devices hold DRAM
- Exploit intrinsic DRAM PUF to derive a unique fingerprint & derive a key
- DRAM has larger capacity than SRAM
- Runtime PUF rather than boot-up time



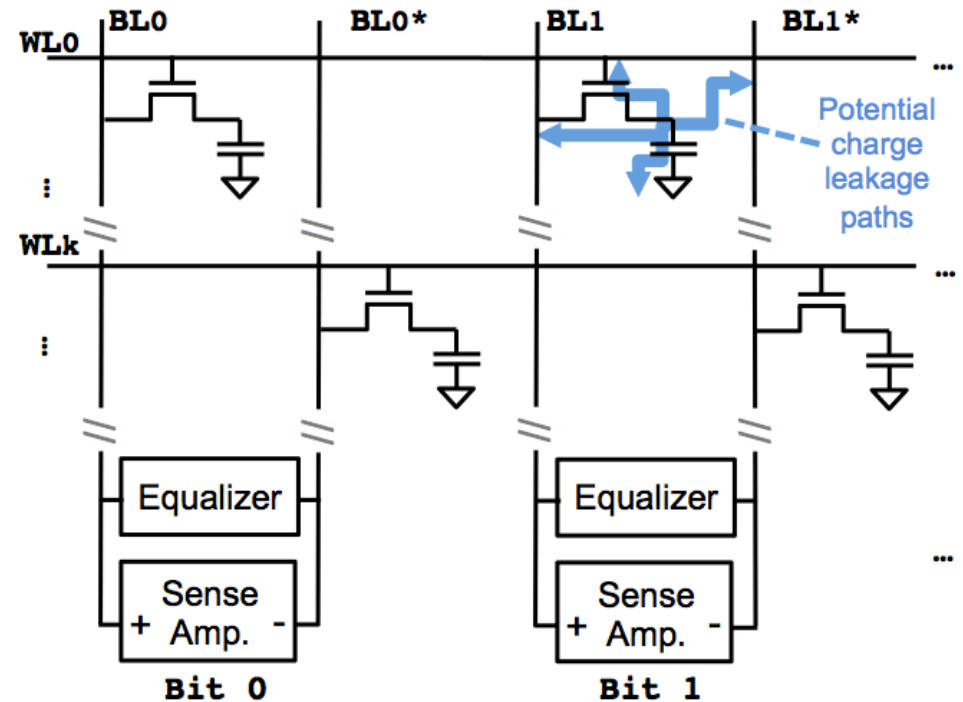
Experimental platforms:  
PandaBoard (top) and  
Intel Galileo (bottom).

# Outline / Contributions

- **Extract decay-based DRAM PUF** instances from unmodified commodity devices during run-time of the Linux system
- **Introduce new metrics** for evaluating DRAM PUFs, based on the Jaccard index
- Through extensive experiments, we show that DRAM PUFs exhibit **robustness, uniqueness, and stability**
- **Design protocols** for device authentication and secure channel establishment that draw their security from the time-dependent decay of DRAM cells

# DRAM Cell Decay

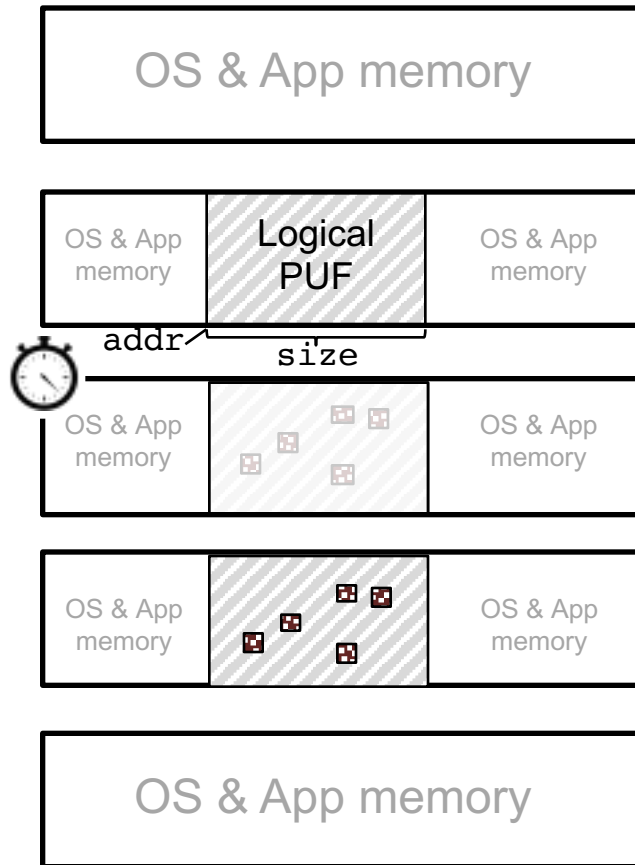
- A DRAM cell consists of a capacitor and a transistor
- Bit is stored as charge
- DRAM access process
- Charge leakage
  - DRAM refresh
  - Access a word will refresh the whole row
- Due to the manufacturing variations among DRAM cells, some cells decay faster than others, which can be exploited as a PUF



Schematic of a DRAM array; arrows indicate leakage paths for dissipation of charges that lead to PUF behavior.

# DRAM PUF Access

DRAM PUF challenge: **Logical PUF** (addr and size), initial value (0 or 1), **decay time**



(1) DRAM for ordinary use

(2) PUF region (in grey) is initialized and the DRAM **refresh is disabled**

(3) PUF cells decay for time  $t$

(4) Read out the DRAM to extract the PUF measurement

(5) DRAM return to normal usage

# Implementations

- Two approaches
  - Firmware
    - DRAM is not used by firmware, so the whole DRAM refresh can be disabled
  - Kernel module
    - Selective DRAM refresh
      - Read a word in each DRAM row, and thus, refresh the DRAM used by the system and applications

memory size	selective refresh time	%CPU time (64ms refresh period)	%CPU time (200ms refresh period)
32MB	7.6ms	12%	4%
64MB	12.1ms	19%	6%
128MB	21.2ms	33%	10%

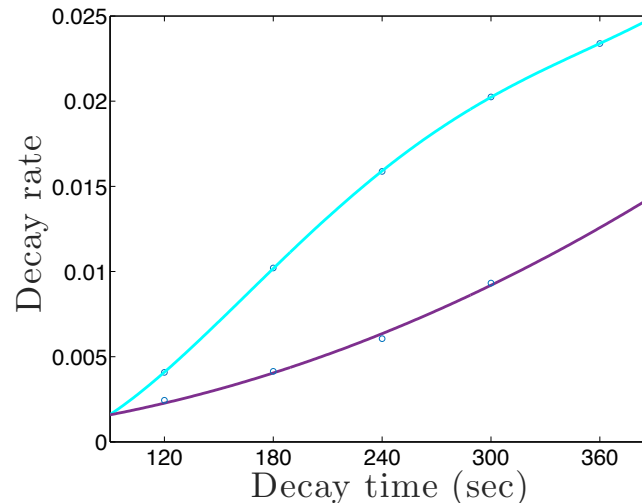
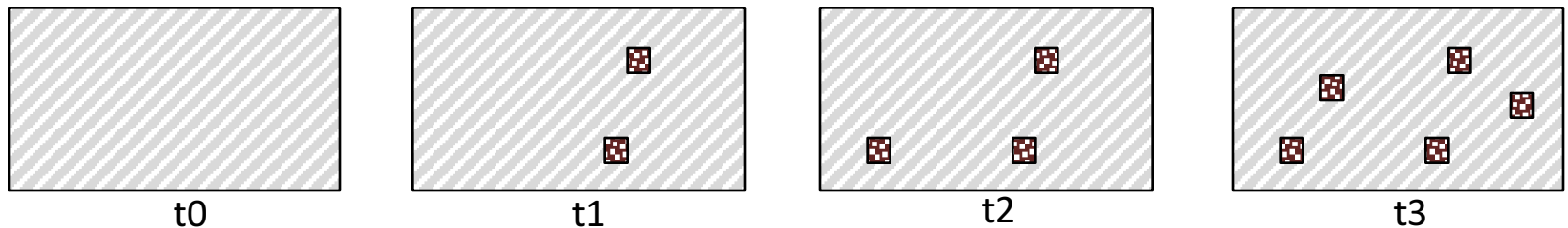
# Outline / Contributions

- **Extract decay-based DRAM PUF** instances from unmodified commodity devices during run-time of the Linux system
- **Introduce new metrics** for evaluating DRAM PUFs, based on the Jaccard index
- Through extensive experiments, we show that DRAM PUFs exhibit **robustness, uniqueness, and stability**
- **Design protocols** for device authentication and secure channel establishment that draw their security from the time-dependent decay of DRAM cells



# DRAM PUF Characteristics

We measured **two** 32KB logical PUFs on **4** PandaBoards and **5** Intel Galileos. Each logical PUF was measured at **five** decay times with **50** measurements each.

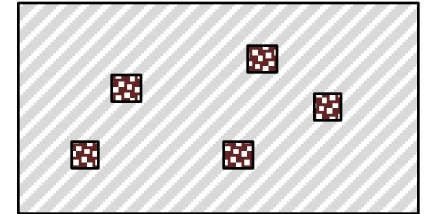


Average decay rate of DRAM modules of (blue) PandaBoard and (purple) Intel Galileo.

If  $t_x \leq t_{x+1}$  and  $\text{addr}_x = \text{addr}_{x+1}$ ,  $\text{size}_x = \text{size}_{x+1}$ , we observe  $m_x \subseteq m_{x+1}$ , up to noise.

# DRAM PUF Characteristics

- PUF measurement: A string of 0's and 1's
  - > A set of bit flips
- Hamming distance -> Jaccard index



$$J(v_1, v_2) = \frac{|v_1 \cap v_2|}{|v_1 \cup v_2|}$$

- Intra Jaccard Index:
  - PUF measurements of the **same** PUF challenge.
  - Ideally, the measurements are the same.  $J_{\text{intra}} \approx 1$ .
- Inter Jaccard Index:
  - PUF measurements of **different** PUF challenges.
  - Ideally, the measurements are completely different.  $J_{\text{inter}} \approx 0$ .

# DRAM PUF Characteristics

## Robustness and Uniqueness

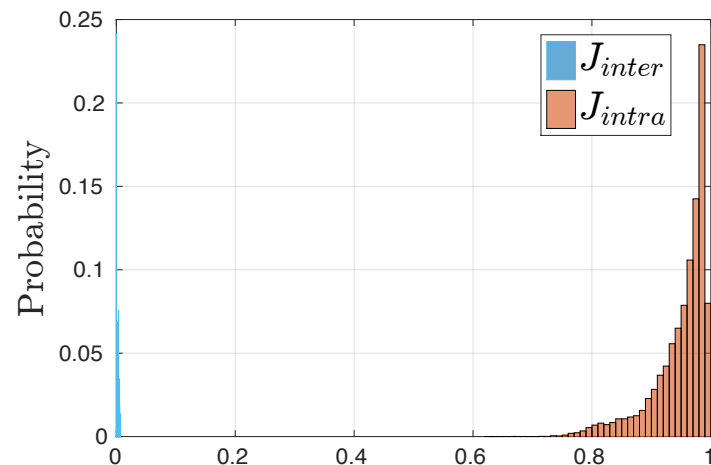
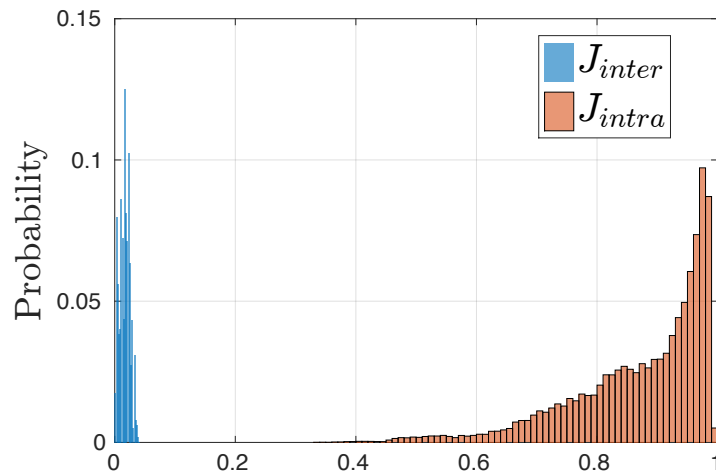
- Robustness: For the same PUF, the same challenge  $x$  should always produce almost the same response  $y$ .  $J_{\text{intra}} \approx 1$
- Uniqueness: For different PUF, the same challenge  $x$  should always produce very different response  $y$ .  $J_{\text{inter}} \approx 0$
- Jaccard index is better at distinguishing DRAM PUF measurements.

Decay time	device	Min $J_{\text{intra}}$	Max $J_{\text{inter}}$	Max fractional intra-HD	Min fractional inter-HD
120s	PandaBoard	0.4634	0.0102	0.0045	0.0038
	Galileo	0.7712	0.0038	0.0003	0.0012
180s	PandaBoard	0.4382	0.0168	0.0083	0.0139
	Galileo	0.8361	0.0044	0.0005	0.0032
240s	PandaBoard	0.4087	0.0258	0.0101	0.0244
	Galileo	0.6261	0.0049	0.0020	0.0057
300s	PandaBoard	0.4222	0.0405	0.0123	0.0238
	Galileo	0.7944	0.0055	0.0013	0.0080
360s	PandaBoard	0.3484	0.0342	0.0206	0.0279
	Galileo	0.8276	0.0072	0.0022	0.0124

# DRAM PUF Characteristics

## Robustness and Uniqueness

- Robustness: For the same PUF, the same challenge  $x$  should always produce almost the same response  $y$ .  $J_{intra} \approx 1$
- Uniqueness: For different PUF, the same challenge  $x$  should always produce very different response  $y$ .  $J_{inter} \approx 0$
- There is a clear gap between  $J_{intra}$  and  $J_{inter}$ . -> Uniqueness

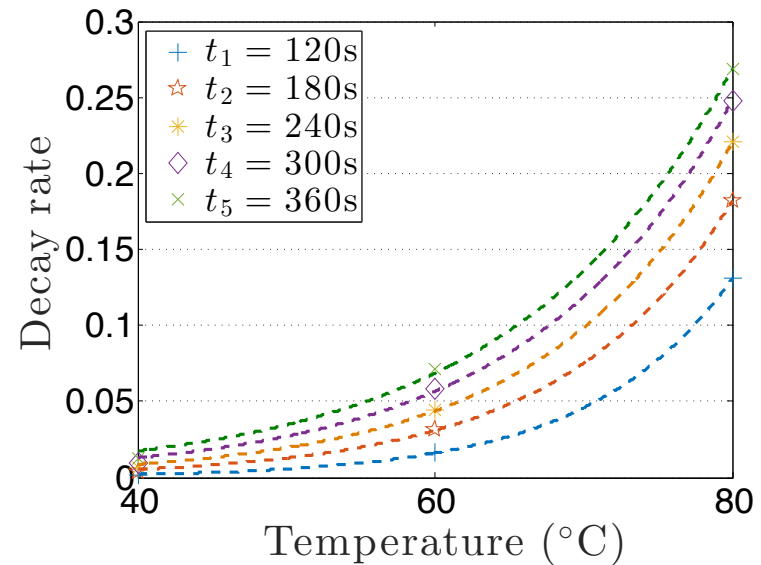
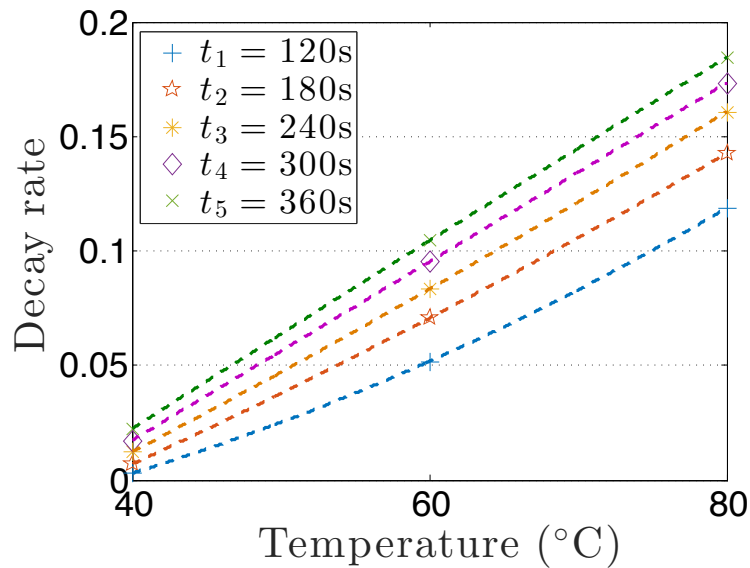


Jaccard index between pairs of measurements      Jaccard index between pairs of measurements

Distribution of  $J_{intra}$  and  $J_{inter}$  values for (left) PandaBoard and (right) Intel Galileo.

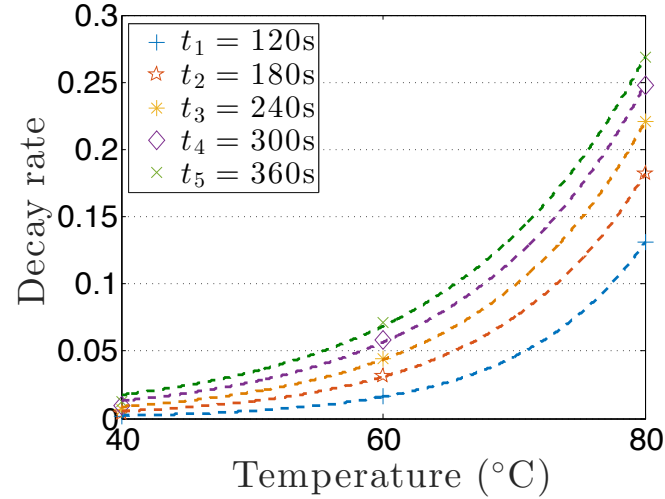
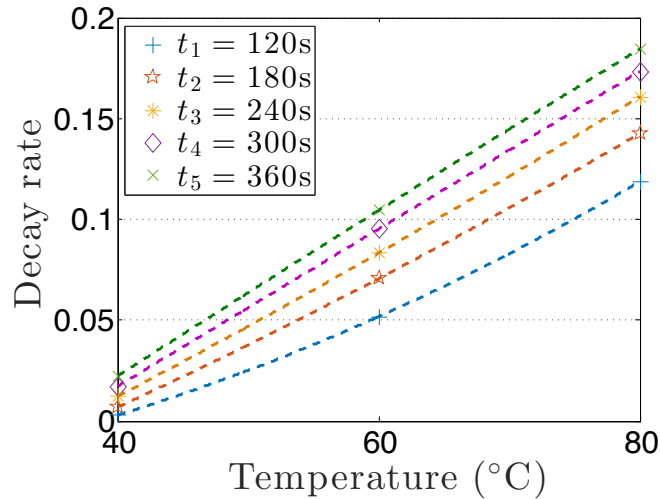
# Temperature Dependency

- DRAM decay also depends on the ambient temperature.
- We conducted temperature experiments with a heater on top of the DRAM.

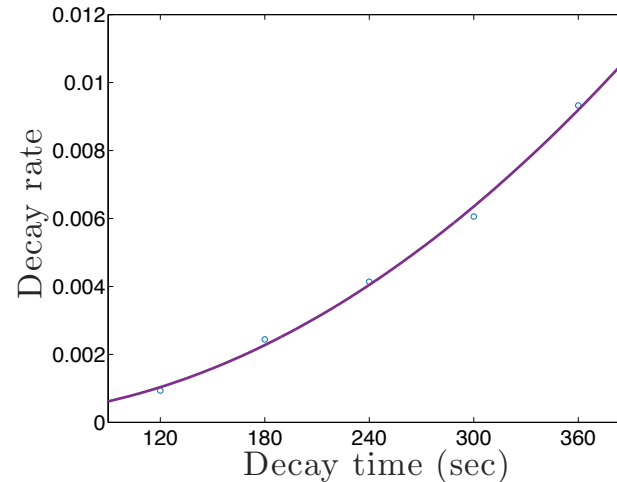
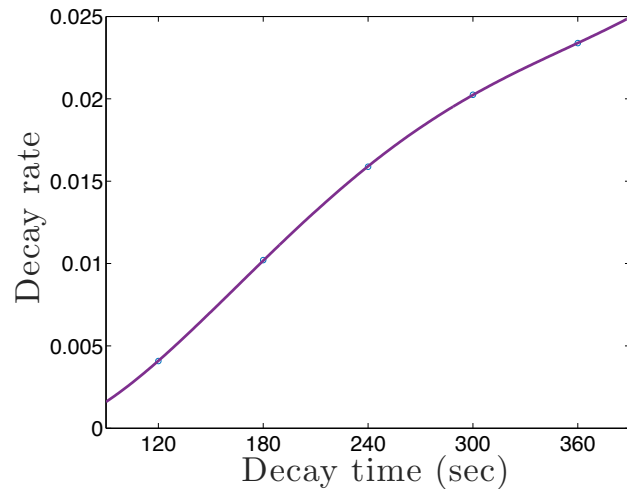


Temperature-dependent decay of (left) PandaBoard and (right) Intel Galileo.

# Temperature Dependency



Temperature-dependent decay of (left) PandaBoard and (right) Intel Galileo.

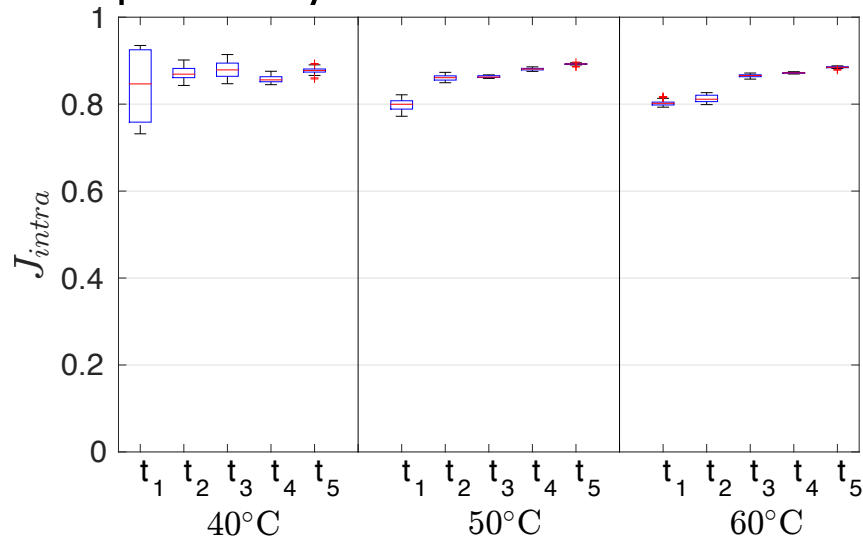


# Temperature Dependency

- High temperature speeds up the DRAM cell decay.

$$t'_{T'} = t * e^{-0.0662 * (T' - T)}$$

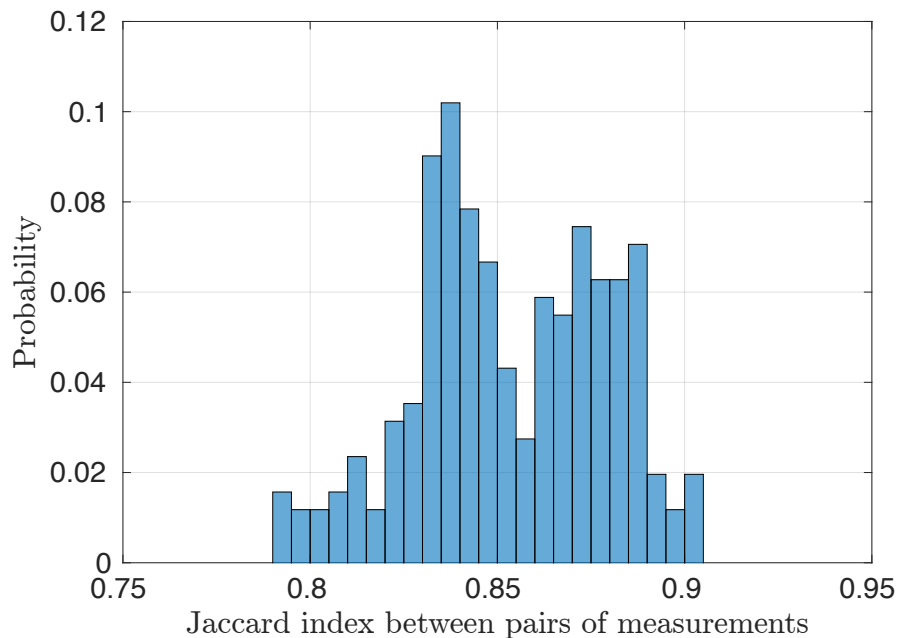
- Under different temperature, with *equivalent* decay time the same decay can be observed.
- The temperature dependency does not affect the robustness of the PUF.



$J_{intra}$  (i.e. similarity) of enrollment measurements taken at 40°C and measurements at  $T' = \{40^\circ\text{C}, 50^\circ\text{C}, 60^\circ\text{C}\}$  on Intel Galileo.

# Stability

- We took measurements from same PUF 4 months apart.
- The minimum Jaccard index is no worse than  $J_{\text{intra}}$ .  
-> The PUF is stable over 4 months.



Distribution of Jaccard index of measurements taken from the same logical PUF on Intel Galileo over 4 months with decay time 200s.



# Outline / Contributions

- **Extract decay-based DRAM PUF** instances from unmodified commodity devices during run-time of the Linux system
- **Introduce new metrics** for evaluating DRAM PUFs, based on the Jaccard index
- Through extensive experiments, we show that DRAM PUFs exhibit **robustness, uniqueness, and stability.**
- **Design protocols** for device authentication and secure channel establishment that draw their security from the time-dependent decay of DRAM cells

# Protocol for Authentication

- Threat model:

A passive attacker, who is able to observe the network traffic

- Enrollment:

A defined set of decay times

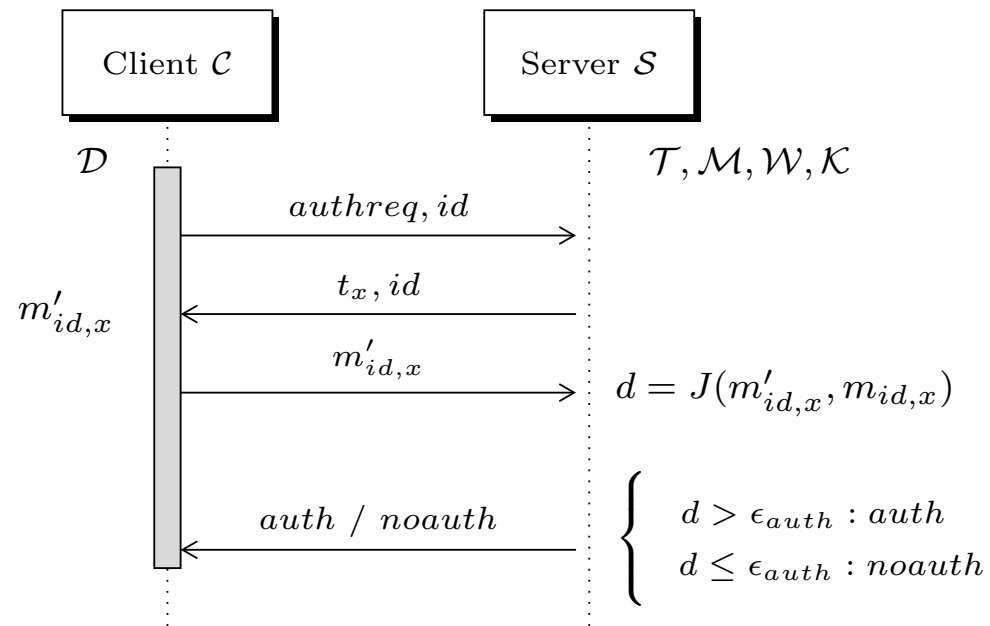
$$T = \{t_0, t_1, \dots, t_n\}$$

Measurements for each logical PUF

$$M = \{m_{id,0}, m_{id,1}, \dots, m_{id,n}\}$$

- Authentication:

The server chooses the smallest decay time  $t_x$  not previously used for the logical PUF  $id$ .



# Secure Channel Establishment

If there exists a secure fuzzy extractor for our DRAM PUF.

- Enrollment

A defined set of decay times

$$T = \{t_0, t_1, \dots, t_n\}$$

Measurements for each logical PUF

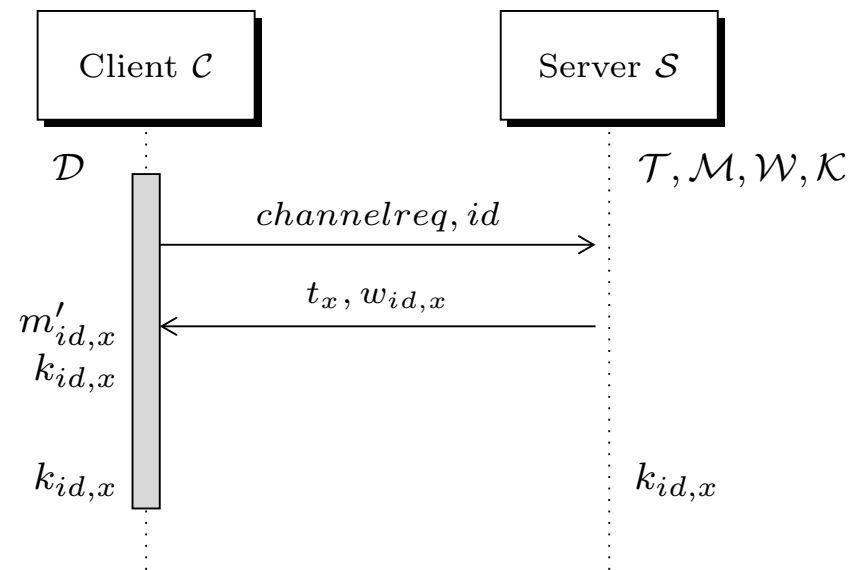
$$M = \{m_{id,0}, m_{id,1}, \dots, m_{id,n}\}$$

A set of random keys

$$K = \{k_{id,0}, k_{id,1}, \dots, k_{id,n}\}$$

Helper data

$$W = \{w_{id,0}, w_{id,1}, \dots, w_{id,n}\}$$



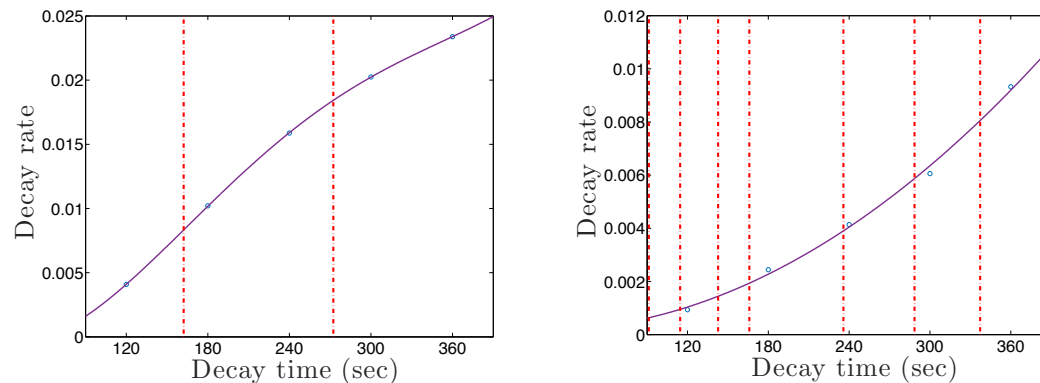
# Time Dependent Decay

- How to choose the set of decay times  $T=\{t_0, t_1, \dots, t_n\}$ ?



- Security is in the newly flipped bits in  $t_{x+1}$  compared to  $t_x$ .
- Security parameter  $\epsilon_{\text{bits}}$  : number of newly flipped bits.

Knowing  $m_x$ , the probability of a random guess of  $m_{x+1}$  being successful is smaller than  $2^{-128}$ .



Red lines indicate possible decay time challenges. Intel Galileo can provide 7 challenges, and PandaBoard can provide 2 challenges with 32KB logical PUF and decay time  $t < 360$ s.

# Conclusions

- Extract decay-based DRAM PUF instances from unmodified commodity devices.
  - Two platforms: the PandaBoard and the Intel Galileo
  - Two approaches: a customized firmware, and a kernel module
- Introduced new metrics for evaluating DRAM PUFs, based on the Jaccard index.
- Showed that DRAM PUFs exhibit robustness, uniqueness, and stability with the decay time as part of the PUF challenge.
- Designed protocols for device authentication and secure channel establishment that draw their security from the time-dependent decay of DRAM cells.

# Future work

- Construct fuzzy extractor for DRAM PUF
  - Jaccard index
  - Biased PUF
- Better understand DRAM PUF characteristics
  - Temperature dependency
  - Voltage dependency
- Improve read out time
  - In the order of minutes

# Acknowledgements

This work has been co-funded by the DFG as part of project P3 within the CRC 1119 CROSSING, and partly funded by CASED.

Thanks to Kevin Ryan and Ethan Weinberger for their help with building the heater setup.

Thanks to Intel for donating the Intel Galileo boards used in this work.

Thanks to anonymous CHES reviewers, and especially our shepherd, Roel Maes.

# Q&A

- Extract decay-based DRAM PUF instances from unmodified commodity devices.
  - Two platforms: the PandaBoard and the Intel Galileo
  - Two approaches: a customized firmware, and a kernel module
- Introduced new metrics for evaluating DRAM PUFs, based on the Jaccard index.
- Showed that DRAM PUFs exhibit robustness, uniqueness, and stability with the decay time as part of the PUF challenge.
- Designed protocols for device authentication and secure channel establishment that draw their security from the time-dependent decay of DRAM cells.





# DRAM cell decay

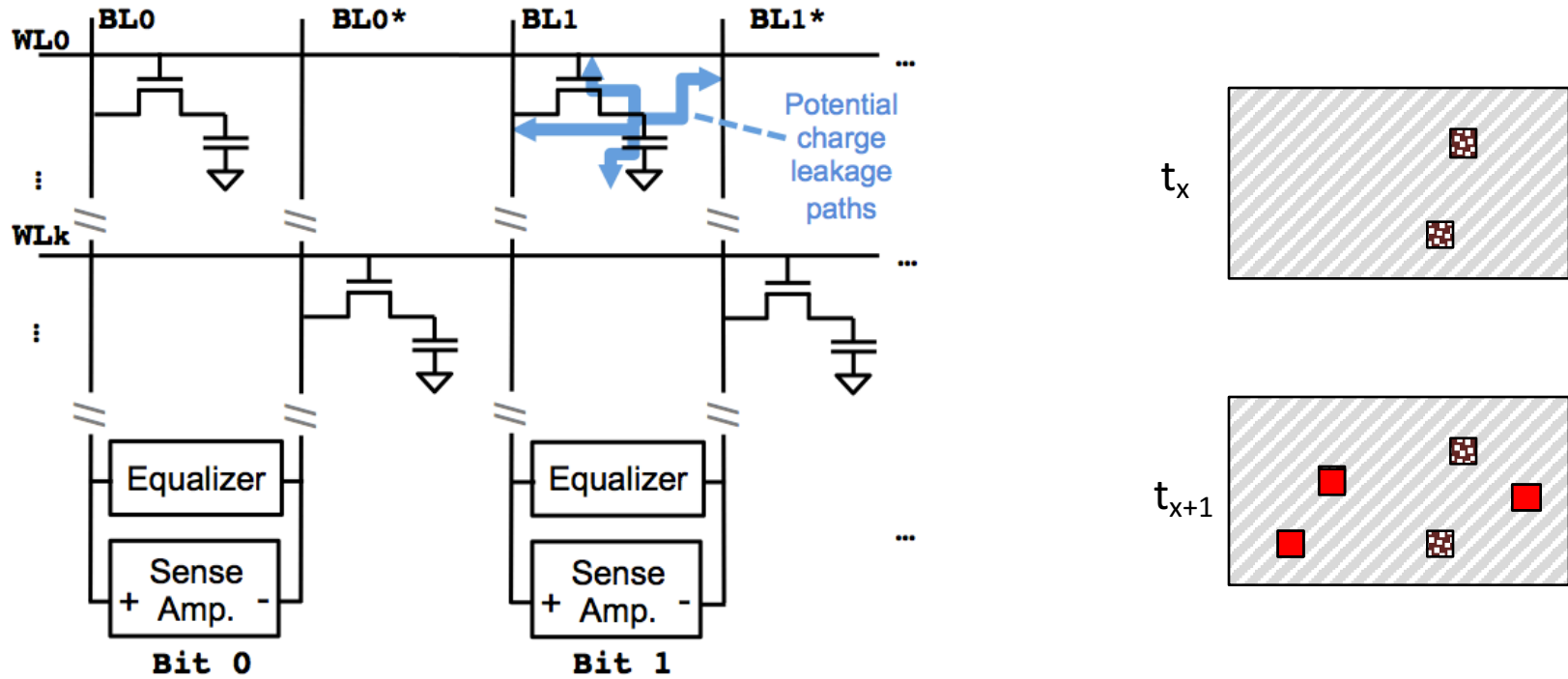


Figure 1: Schematic of a DRAM array; arrows indicate leakage paths for dissipation of charges that lead to PUF behavior.

# Robustness and Uniqueness

Decay time	device	Min $J_{intra}$	Max $J_{inter}$	Max fractional intra-HD	Min fractional inter-HD
120s	PandaBoard	0.4634	0.0102	0.0045	0.0038
	Galileo	0.7712	0.0038	0.0003	0.0012
180s	PandaBoard	0.4382	0.0168	0.0083	0.0139
	Galileo	0.8361	0.0044	0.0005	0.0032
240s	PandaBoard	0.4087	0.0258	0.0101	0.0244
	Galileo	0.6261	0.0049	0.0020	0.0057
300s	PandaBoard	0.4222	0.0405	0.0123	0.0238
	Galileo	0.7944	0.0055	0.0013	0.0080
360s	PandaBoard	0.3484	0.0342	0.0206	0.0279
	Galileo	0.8276	0.0072	0.0022	0.0124

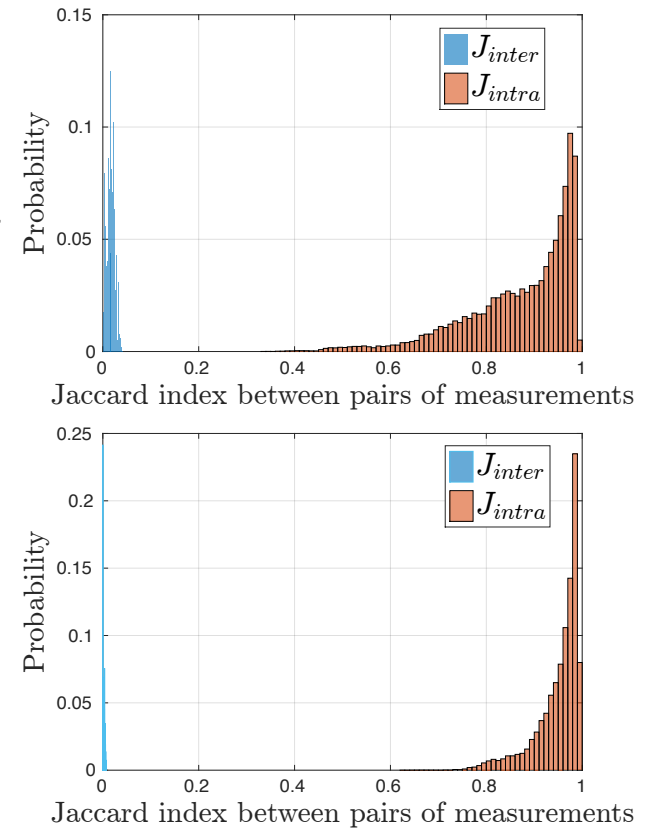


Figure 3: Distribution of  $J_{intra}$  and  $J_{inter}$  values for (left) Pandaboard and (right) Intel Galileo.

# Temperature dependency

- High temperature speeds up the DRAM cell decay.

$$t'_{T'} = t * e^{-0.0662 * (T' - T)}$$

- Under different temperature, with *equivalent* decay time the same decay can be observed.
- The temperature dependency does not affect the robustness of the PUF.

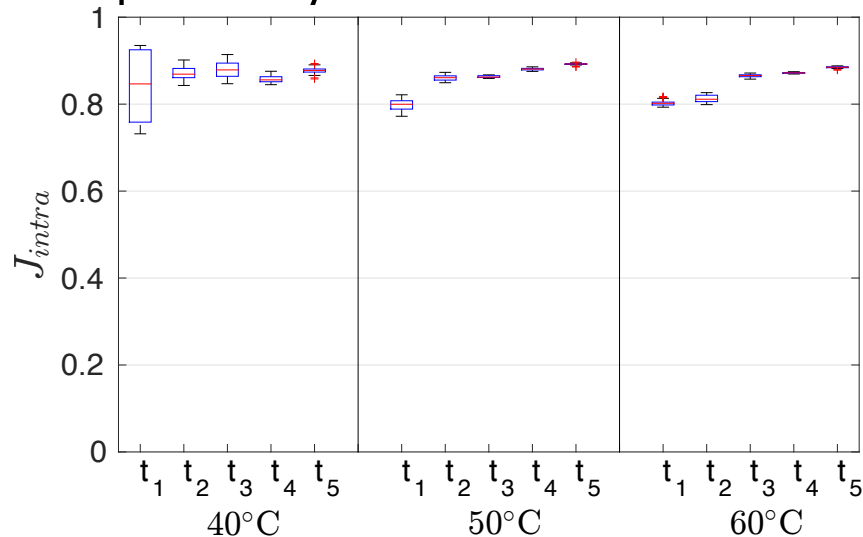


Figure 5:  $J_{intra}$  (i.e. similarity) of enrollment measurements taken at 40°C and measurements at  $T' = \{40^\circ\text{C}, 50^\circ\text{C}, 60^\circ\text{C}\}$  on Intel Galileo.