Hardware-entangled security

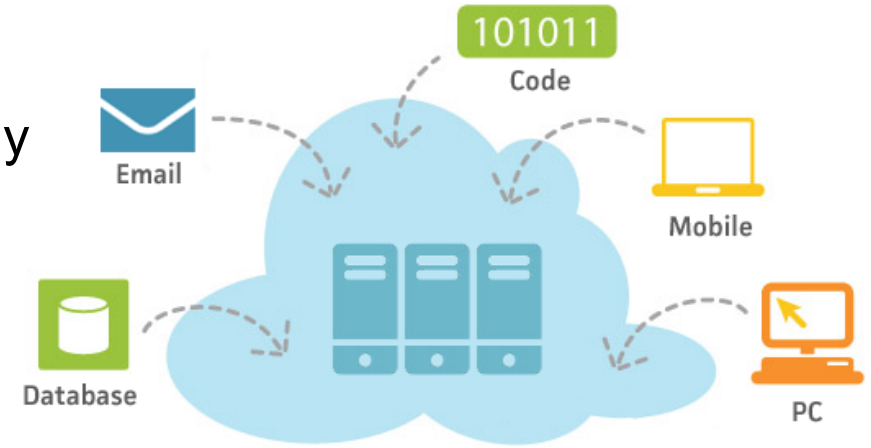
Extracting a secure key from IoT hardware

Nikolaos Athanasios Anagnostopoulos

TECHNISCHE UNIVERSITÄT DARMSTADT

Cloud components and security

- Different components: Services, devices, databases, raw code, software applications, firmware….
- Which components can/should be really considered as secure?
- Where is the user??
- Why do we assume that security established by/from the user (credentials) is equal to a secure computer?

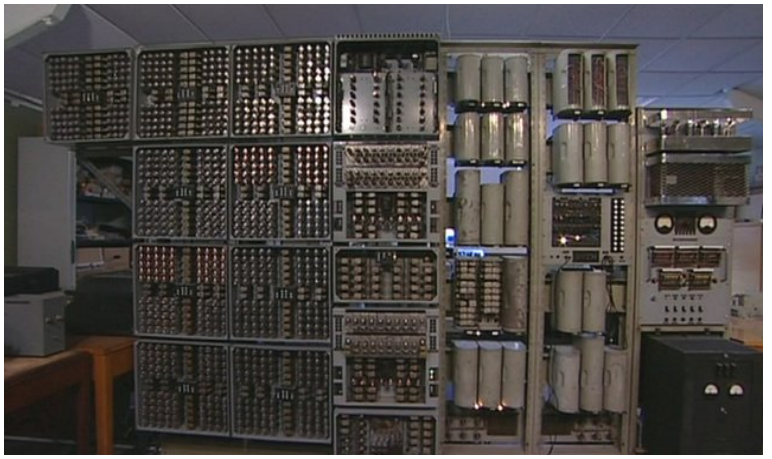


Insecurity in the cloud

- Not all services on the cloud can be considered secure.
- However, users and companies need the cloud because of its availability, flexibility, scalability, ease of service, ….
- Is there a solution?
- If photographs are not safe on the cloud, would you risk your business data?

Hardware-entangled cryptography

- Can we bind services to hardware?
- Hardware will always be present when accessing the cloud.
- However, there may be convenience issues.
- Nevertheless, browser-based services already do flexible binding.
- Advantages:
  - Much more difficult to break.
  - Can be combined with existing solutions.
  - Client-based flexibility.



CROSSING

TECHNISCHE UNIVERSITÄT DARMSTADT

DFG Deutsche Forschungsgemeinschaft

Project P3
Hardware-entangled cryptography

CROSSING

P3

Hardware-entangled cryptography

Physical(ly) Unclonable Functions

- Cryptography based on hardware characteristics
- Functions embedded into physical objects
- Can be used for identification, secure key storage or even in bit commitment and oblivious transfer schemes

Stefan Katzenbeisser

Nikolaos Athanasios Anagnostopoulos

CROSSING

DFG Deutsche Forschungsgemeinschaft

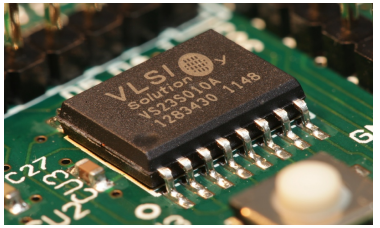
TECHNISCHE UNIVERSITÄT DARMSTADT

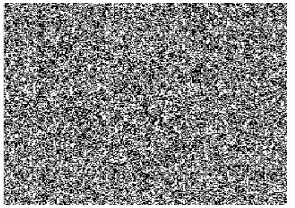
Use cases for PUFs

- Authentication and identification

- Integrity of devices
  - Anti-counterfeiting
  - Tamper-evidence

- Lightweight security





CROSSING

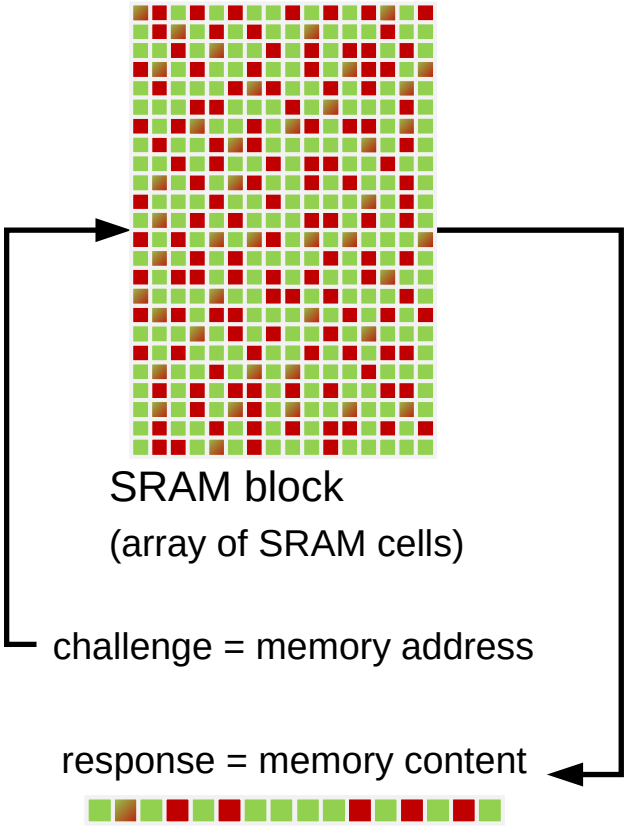
DFG Deutsche Forschungsgemeinschaft

TECHNISCHE UNIVERSITÄT DARMSTADT

PUFs

Physical Unclonable Functions (PUFs)

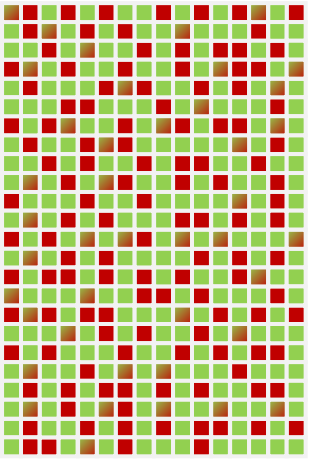
- Functions embedded into physical objects
- Manufacturing process variations
  → unique identity for ICs
- When queried with a challenge, a PUF generates a response (Challenge-Response Pair; CRP)
- The response depends on
  - the challenge **and**
  - specific physical properties of the object



Weak PUFs

Weak PUFs

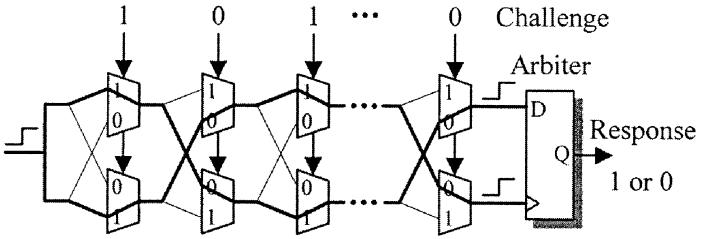
- A single or very few challenge-response pairs
- Memory-based PUFs
- In production stage
- Inherent to most devices

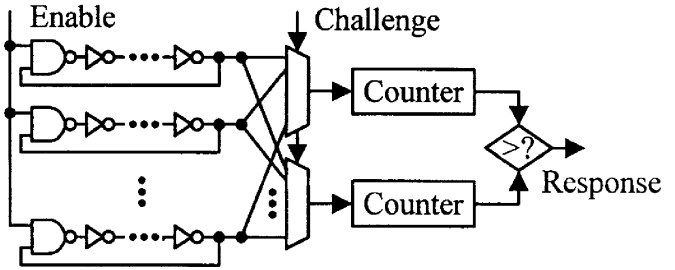


Strong PUFs

Strong PUFs

- Multiple challenge-response pairs
- Delay-based PUFs
- Still on the prototype stage
- Require dedicated circuitry

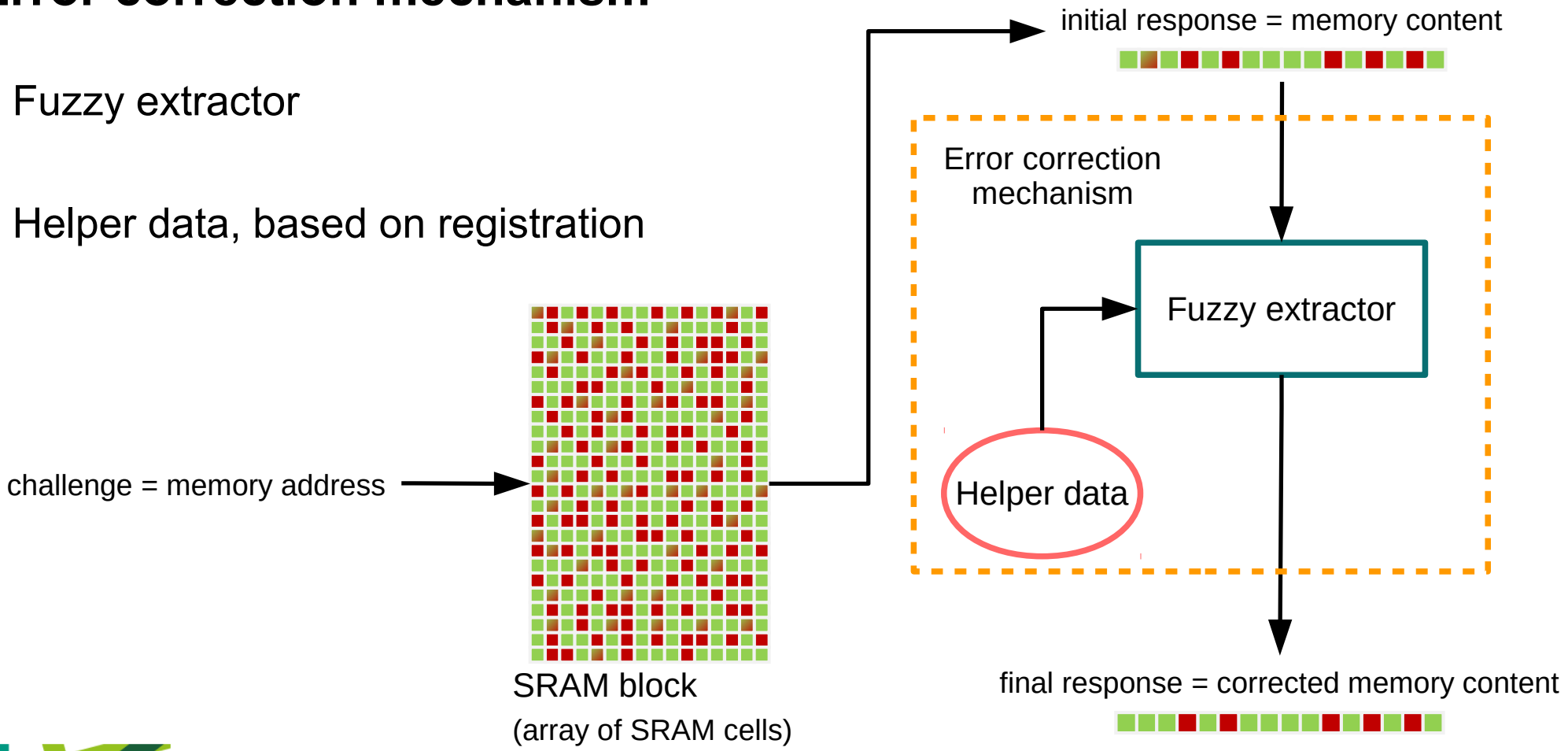




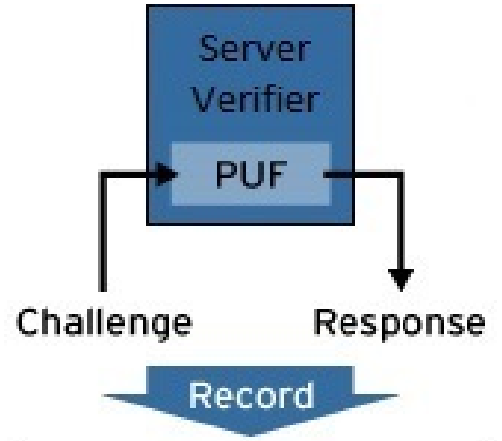
Error correction for PUFs

Error correction mechanism

- Fuzzy extractor
- Helper data, based on registration

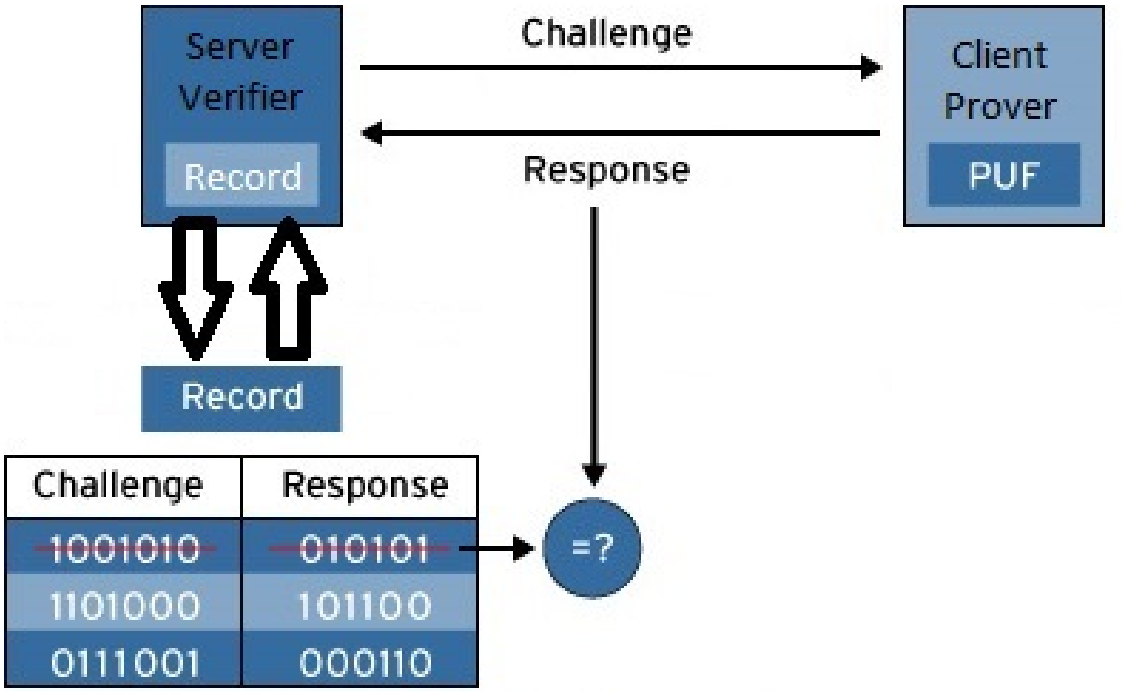


Simple PUF protocol example



Challenge	Response
~~1001010~~	~~010101~~
1101000	101100
0111001	000110

Phase 1: Enrollment

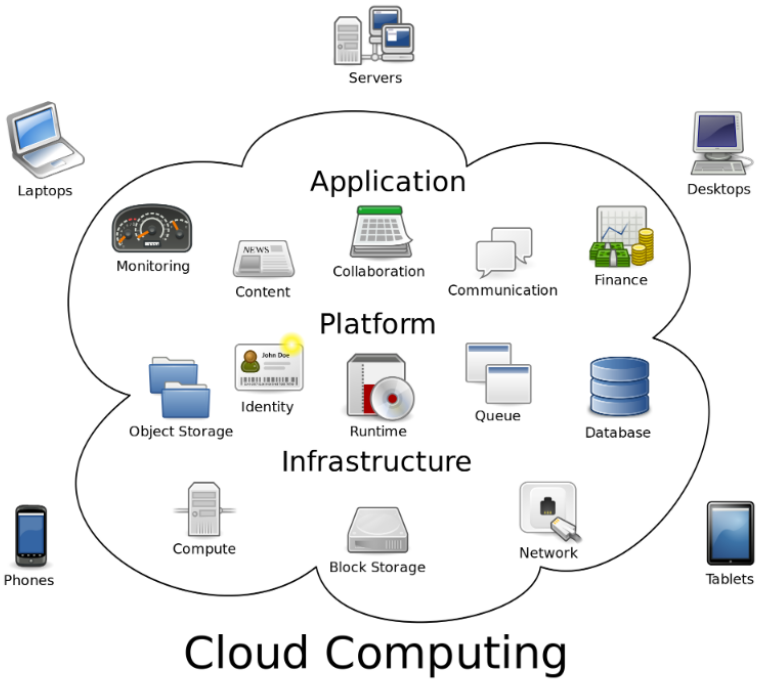


Challenge	Response
~~1001010~~	~~010101~~
1101000	101100
0111001	000110

Phase 2: Authentication

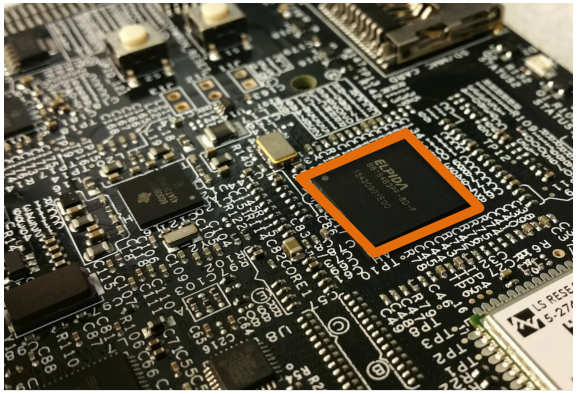
Cloud, fog or mist?

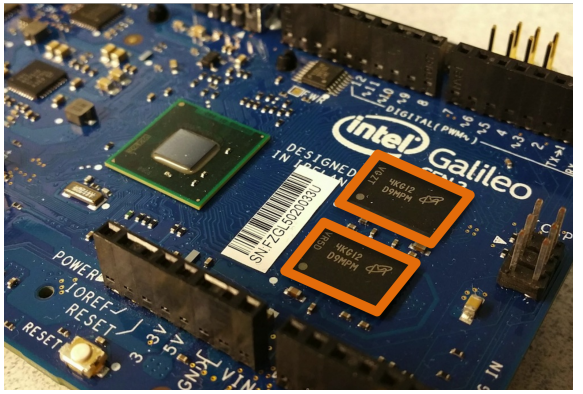
- What about the diversity of devices?
- We would need to use a weak PUF.
- But, an additional problem that has been hard to tackle is using weak PUFs at runtime.
- Is hardware flexible enough?
- If the cloud is not always secure, can we trust the fog or the mist?



Data remanence DRAM PUF

- **Extract decay-based DRAM PUF** instances from unmodified commodity devices during run-time of the Linux system
  - No extra chips needed for PUF
  - Exploit hardware which is on-board anyway
- Through extensive experiments, we determine that DRAM PUFs exhibit **robustness, uniqueness, and stability**
- It is thus possible to **design protocols** for device authentication and secure channel establishment that draw their security from the time-dependent decay of DRAM cells

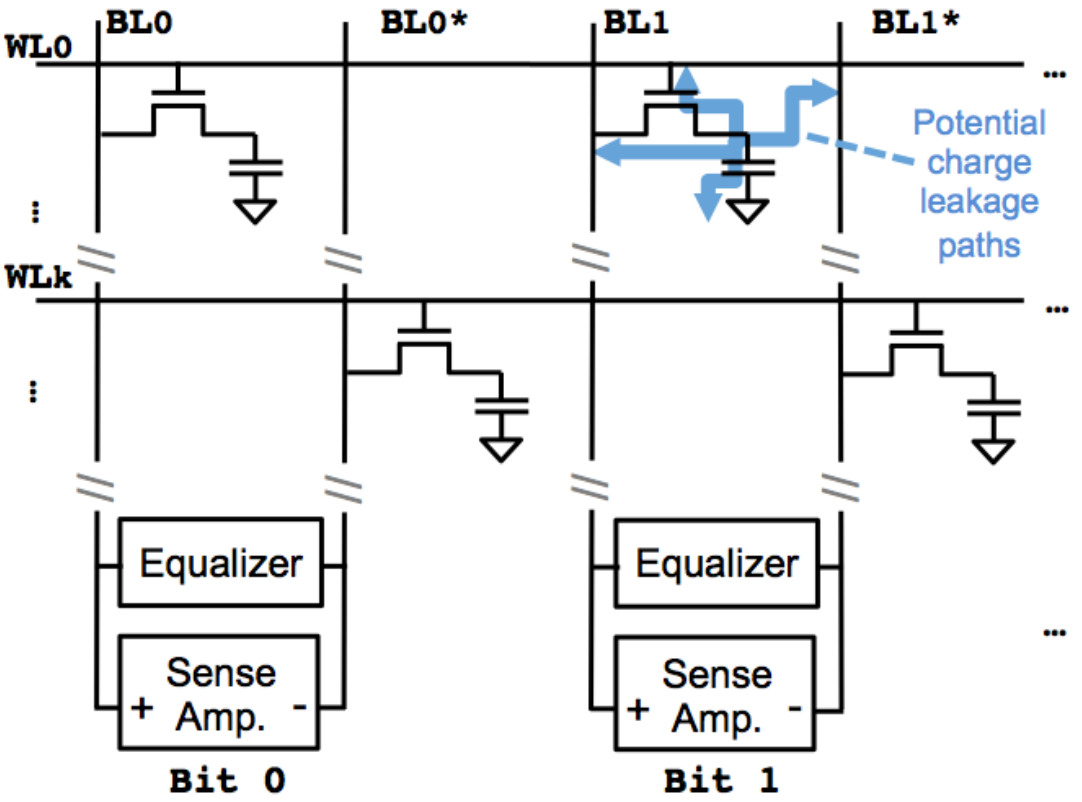




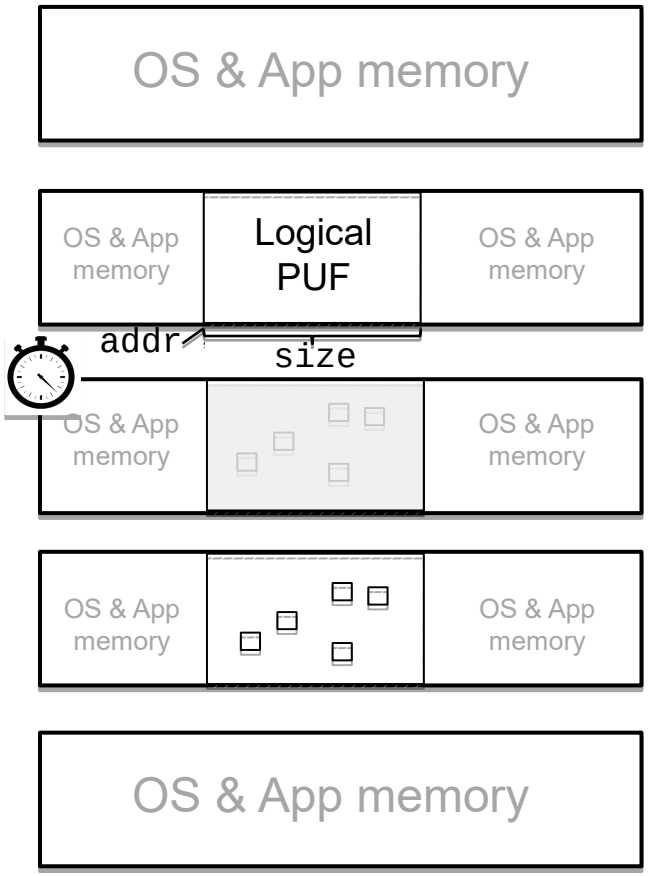
Experimental platforms
- Pandaboard (top)
- Intel Galileo (bottom)

DRAM model

- A DRAM cell consists of a capacitor and a transistor
- Each bit is stored as charge
  - Charge leakage
  - DRAM refresh
- Accessing 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



DRAM PUF model



(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

Implementation

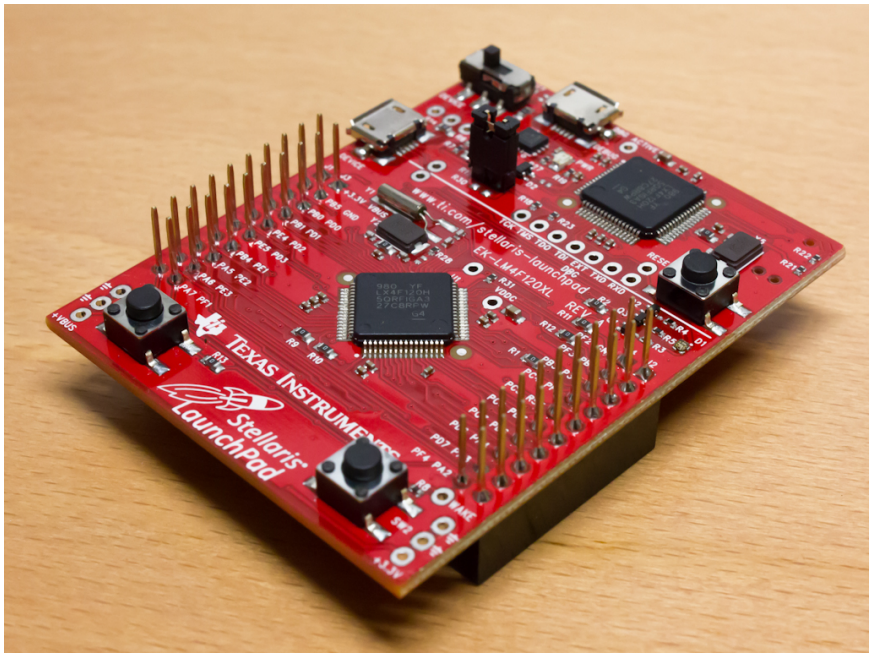
Two approaches

- 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

Data remanence in intrinsic SRAM PUFs

- Data remanence
  - Dependent on temperature
  - Exists on both DRAMs and SRAMs
  - Can be very short-lasting on SRAMs (tens of milliseconds)

- SRAMs are no longer individual modules
  - Usually Package-on-Package
  - Sometimes on-die
  - May serve as cache
  - User-space programs have access, but attacks are more difficult



PUFs, IoT and the Cloud

- If we can use PUFs at runtime on IoT devices, why not use this also on the Cloud?
- We do need better security solutions for the cloud; more scalable, more agile.
- Hardware can offer this: Different devices get different privileges on the same account.
- If we can provide security for machines, we should be able to provide security also for people.

Advantages and potential PUF solutions

- Inherent solutions: Low cost
- Automation: The machine authenticates itself
- Server-client security
- Scalability
- Turn attacks into security features: Rowhammer PUF
- Use communication modules for security: WLAN/WiFi modules
- Use communication channels as PUFs

TECHNISCHE UNIVERSITÄT DARMSTADT

DFG Deutsche Forschungsgemeinschaft

CROSSING

P3

Discussion