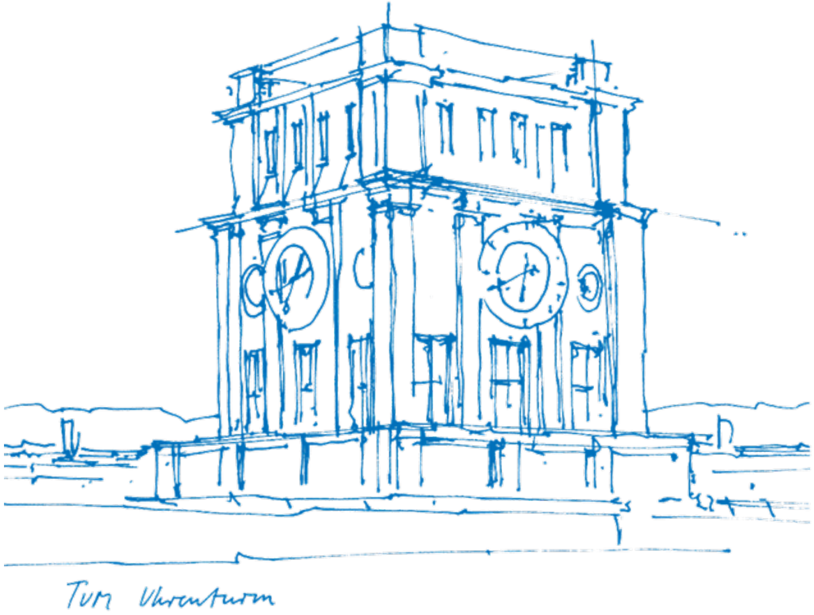
# Authentication using DRAM PUFs

Chirag Mahaveer Parmar[1]
[1]Technical University of Munich
MSc. Communications Engineering
June 24, 2021



# Overview

Motivation

Basics of PUFs and DRAM PUFs

Authentication for Strong PUFs

Authentication for Weak PUFs

Conclusion/Summary

# Motivation

*"Security and privacy concerning embedded devices connected to the internet is increasing..."*

- **Top Concern:** Counterfeit and tampered hardware.
- Traditional solutions do not completely solve the problem
  - ROM keys - Can be physically extracted by intrusive methods and cloned.
- This demands hardware-intrinsic security mechanisms which are **physically unclonable**
- Physically Unclonable Functions a.k.a PUFs

# Basics - PUFs

**Definition (Physically Unclonable Functions)**

A physical object that takes an input(challenge) and provides a response that is **unique** to the object.

- Each PUF instance is able to create a set number of challenge-response pairs depending on its physical characteristics.
- Strong PUF: Large CRP Database
- Weak PUF: Small CRP Database
- PUF constructions need to be hardware-intrinsic. Hence, they usually take advantage of manufacturing variations for uniqueness.
- Ex. DRAM based PUFs, SRAM based PUFs etc.

# Basics - DRAM based PUFs

- **Startup:** The startup state of the DRAM is used for unique fingerprinting.
  - DRAM are precharged to vdd/2 sense amplifier equally likely to read a ‘1’ or ‘0’.
  - Due to the manufacturing variations there is a **bias**.
- **Write Failures:** Process variations affecting the write reliability are exploited.
  - The duty cycle is deliberately reduced which results in some DRAM cells being overwritten while others are not.
  - The data being written can be treated as the challenge and a subsequent read reveals the response affected by the variations.
- **Refresh Pause:** The refresh of the DRAM cells is paused.
  - Random decaying of the data stored in memory based on the manufacturing variations.
  - The data stored before the delaying refresh can be treated as the challenge and the one after the delay as the response.

# Adversary and Threat model

- A client/prover (C) device wants to authenticate itself a server/verifier (S).
- It is assumed that all network traffic between client and server is observable by attackers except **Enrollment** and **Characterization** phases.
- The attacker might have physical access to the but cannot use it for its intended purpose before authentication.
- The server is assumed to be honest in most settings except **Mutual Authentication**

# Enrollment

- The enrollment phase deals with the generation of the CRP database.
- Stored on a trusted secure database or distributed to verifiers in a secure manner.
- **NOTE:** different verifiers receive different parts of the CRP database, otherwise one can spoof to be the device to the other.

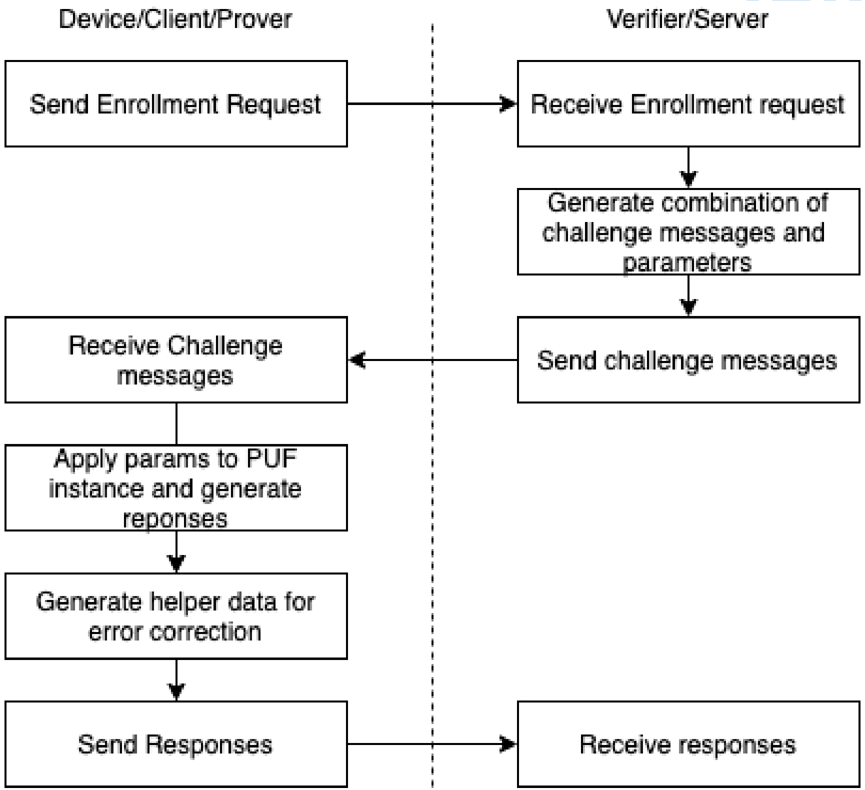



Figure: Enrollment

# Simple Authentication

- The device remeasures a response on obtaining a challenge from the server.
- The server matches the device's remeasured response against the CRP database.
- If they matches, the device is authenticated

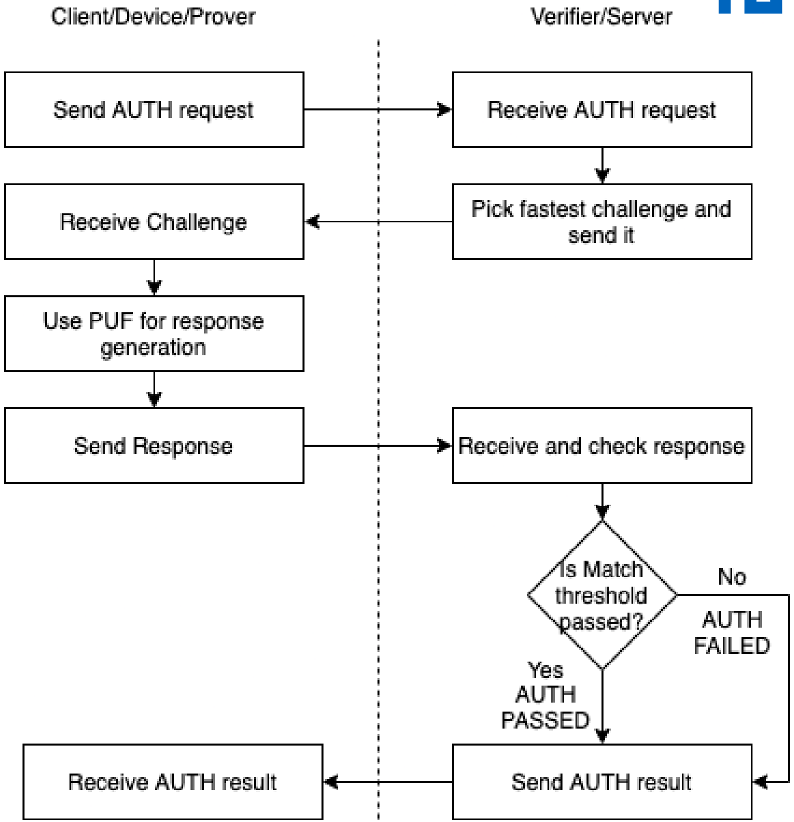



Figure: Simple Authentication

# Mutual Authentication

- Requires the storage of CRPs in form of addresses of flipped bits rather than the bit pattern itself.
- The device mixes random addresses with the flipped addresses before sending the response
- The server verifies the device if the number of matched addresses and checking is above a threshold.
- The server then filters out the known flip addresses from the response and sends them to the device.
- The device verifies the server's response by matching it with its own response.

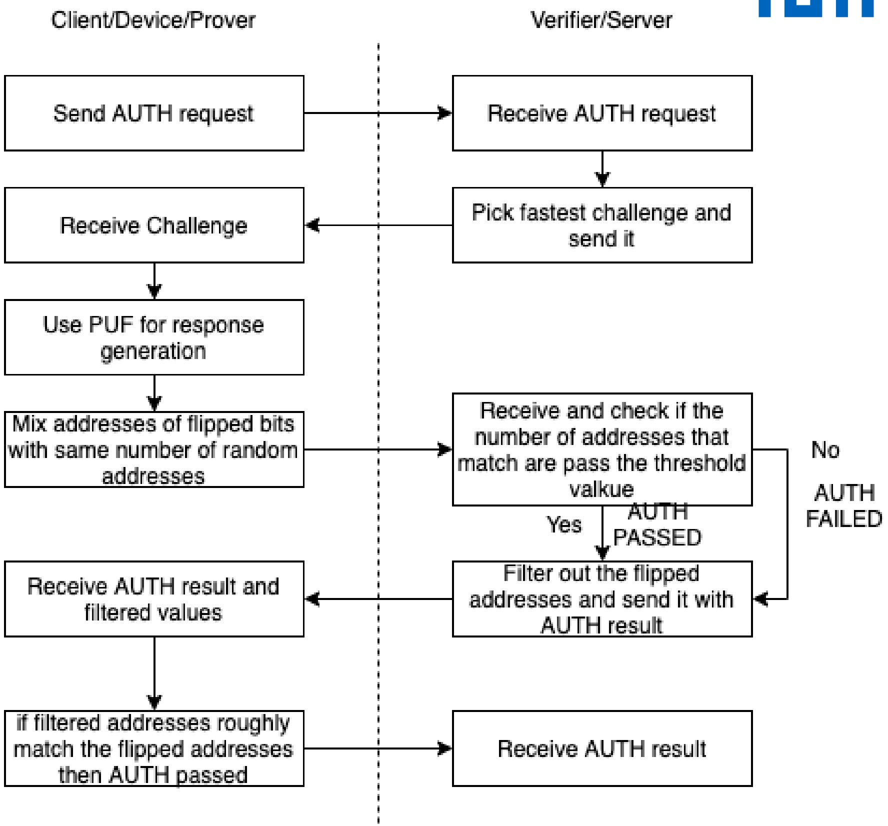



Figure: Mutual Authentication

# Reconfigurability and Characterization

- A CRP is never used twice to prevent replay attacks. Hence you will need a large CRP database.
- **Parameter Reconfigurability** allows changing parameters from the same PUF instance and regenerating a new CRP database on it.
- Current implementations are slow, faster physical parameters are needed.
- **Characterization** exhaustively measures PUF behavior in different conditions and selects the fastest parameters to generate a response. 6x improvement.

# Authentication for Weak PUFs

- Since weak PUFs cannot be used for simple authentication.
- Used with Fuzzy Extractors a.k.a Helper Data Systems for secure key storage.
- A fuzzy extractor corrects errors in the PUF response and enables regeneration of a cryptographic key.
- The stored key then be used to establish a secure channel for authentication

# Fuzzy Extractors

- *Enrollment Algorithm:* The enrollment algorithm takes as input a PUF response $X$ and a random cryptographic key $k$ and outputs helper data $w$.
- *Reconstruction Algorithm:* The reconstruction algorithm takes as input a fresh and noisy measurement of the PUF response $X'$, and helper data $w$, and outputs a reconstructed key $k'$. If the noise in the PUF response is within a threshold $k' = k$

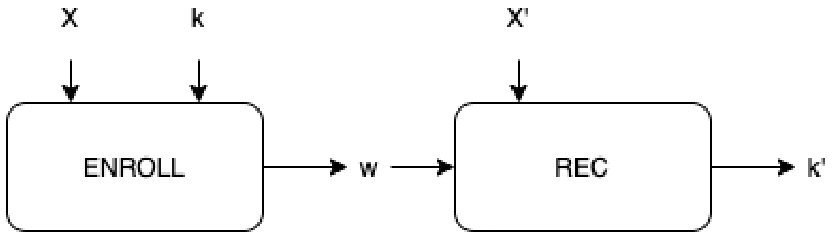



Figure: Enrollment

# The Code Offset Method

- One of the earliest implementations of fuzzy extractors.
- COM uses an error-correcting code at the core of its construction.
- **Enroll:** $w = X \oplus Enc(k)$
  - $Enc()$ is the encoding function of the error-correcting code.
- **Reconstruct:** $k' = Dec(X' \oplus w)$
  - $Dec()$ is the decoding function of the error-correcting code.

## Error Correction Magic

Rewriting the reconstruction formula as $k' = Dec(X' \oplus X \oplus Enc(k))$ shows us that any remaining error bits introduced by $(X' \oplus X)$ is corrected by $Dec()$. $w$ plays the role of a syndrome.

# Syndrome-COM

- Syndrome-COM or SCOM leverages syndrome decoding to use the input, $X$, itself as the cryptographic key.
- This is done by setting the helper data to the syndrome of $X$.
- **NOTE:** Requires sufficient entropy in the input $X$ to not leak information through helper data, $w$.
- Schaller et al overcome this problem by randomly arranging the order of flipped addresses.

**Algorithm 1** Enroll

1: Measure $X \in 0, 1^n$
2: Compute helper data $W = SynX$.
3: Publicly store $W$

**Algorithm 2** Reconstruct

1: Read $W$
2: Measure $X' \in 0, 1^n$
3: $\hat{\mathrm{X}} = X' \oplus SynDec(W \oplus Syn(X'))$

## Error Correction Magic

Rewriting the reconstruction formula, $(W \oplus Syn(X'))$ is just equal to $Syn(X + X')$. So the syndrome decoder extracts the error bits out from X'. This when XORed with $X'$ will give us the required cryptographic key X $\hat{=}$ $X$

# Other FE Constructions

## Concatenated Codes as FE

Make use of concatenation of two linear codes, $C_1 = [n_1, k_1, d_1]$ and $C_2 = [n_2, k_2, d_2]$
Code $C_1$ needs to maintain a high entropy, which directly translates into high cardinality or high $k_1$.
Code $C_2$ needs to have a high error correction rate which directly translates to higher distance, $d_2$.

- It is obvious that we need high input entropy. For this, We cannot increase the codeword size due storage constraints.
- So we increase $k$ but the error correction of the code $C_1$ takes a hit due to the singleton bound.
- To the offset this we concatenate a second code $C_2$ with better error correction capability or high $d$

# Questions?

# PUF Implementations Objectives

- **Uniqueness** in PUFs is measured by comparing responses for the same challenges on different PUF instances. A greater difference in responses indicates stronger uniqueness. This measurement can be using either Hamming Distance or the Jaccard Index.
- **Robustness** of a PUF is measured by comparing the responses for the same challenges on the same PUF instances but on consequent executions. Hence robustness, in this context can also be seen as **reproducibility**. Reproducibility is also checked in varying temperatures and against aging.
- **Run-time access** is required for convenient authentication of the device alongside a running OS. Startup based methods are hence inconvenient because they only provide boot time access. Other approaches use a separate module/memory controller or use selective memory refreshing.

# References


- J. Viega and H. Thompson, ”The State of Embedded-Device Security (Spoiler Alert: It’s Bad),” in IEEE Security & Privacy, vol. 10, no. 5, pp. 68-70, Sept.-Oct. 2012, doi: 10.1109/MSP.2012.134.
- M. Pecht and S. Tiku, ”Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics,” in IEEE Spectrum, vol. 43, no. 5, pp. 37-46, May 2006, doi: 10.1109/MSPEC.2006.1628506.
- Fatemeh Tehranipoor et al. 2015. DRAM based Intrinsic Physical Unclonable Functions for System Level Security. 25th edition on Great Lakes Symposium on VLSI (GLSVLSI ’15). ACM, New York, NY, USA, 15–20. DOI:https://doi.org/10.1145/2742060.2742069
- M. S. Hashemian et al. ”A robust authentication methodology using physically unclonable functions in DRAM arrays,” 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015, pp. 647-652, doi: 10.7873/DATE.2015.0308.

# References

- Soubhagya Sutar et al. D-PUF: An Intrinsically Reconfigurable DRAM PUF for Device Authentication and Random Number Generation. ACM Trans. Embed. Comput. Syst. 17, 1, Article 17 (January 2018), 31 pages. DOI:https://doi.org/10.1145/3105915
- Xiong W. et al. (2016) Run-Time Accessible DRAM PUFs in Commodity Devices. In: Gierlichs B., Poschmann A. (eds) Cryptographic Hardware and Embedded Systems – CHES 2016. CHES 2016. Lecture Notes in Computer Science, vol 9813. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-53140-2_21
- A. Schaller et al., "Decay-Based DRAM PUFs in Commodity Devices," in IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 3, pp. 462-475, 1 May-June 2019, doi: 10.1109/TDSC.2018.2822298.
- J. Miskelly and M. O'Neill, "Fast DRAM PUFs on Commodity Devices," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 11, pp. 3566-3576, Nov. 2020, doi: 10.1109/TCAD.2020.3012218.

# References


- C. Keller et al., ”Dynamic memory-based physically unclonable function for the generation of unique identifiers and true random numbers,” 2014 IEEE International Symposium on Circuits and Systems (ISCAS), 2014, pp. 2740-2743, doi: 10.1109/ISCAS.2014.6865740.
- Gutmann, P. Data remanence in semiconductor devices. In Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10 (Berkeley, CA, USA, 2001), SSYM’01, USENIX Association.
- Jaccard, P. (1901) étude Comparative de la distribuition florale dans une portion des Alpes et des Jura. Bulletin de la Société Vaudoise des Sciences Naturelles, 7, 547-579.
- Skoric, B., & Vreede, de, N. (2014). The spammed code offset method. IEEE Transactions on Information Forensics and Security, 9(5), 875-884. https://doi.org/10.1109/TIFS.2014.2312851
- J. Delvaux, D. Gu, D. Schellekens and I. Verbauwhede, ”Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 6, pp. 889-902, June 2015, doi: 10.1109/TCAD.2014.2370531.