Authentication based on DRAM PUFs

Chirag Mahaveer Parmar

***Abstract*—One of the major security concerns regarding IoT is that of counterfeit or tampered hardware. To overcome this problem it is important to employ a hardware-intrinsic security mechanism for authentication of the device. While previous solutions like secure key storage attempt to solve this problem they only pass on the issue to the next stage. This paper outlines the use of Physically Unclonable Functions (PUFs) for hardware-intrinsic authentication. It summarizes well-known works on DRAM-based PUFs and also presents known protocols and schemes to achieve authentication on embedded devices using PUFs.**

I. Introduction

IN light of the IoT era the world has seen a spike in the number of smart devices deployed in the market. These ubiquitous smart devices can be found anywhere, from toasters to cars. While they bring a lot of advantages to the table they add a profound amount of concern regarding security and privacy. [1] The most prominent security concern includes counterfeit or tampered hardware. [2]

Mitigating these security risks requires hardware-intrinsic security mechanisms that support authentication and identification of the device. A known solution requires cryptographic keys to be burned in at manufacturing time and then later used in symmetric/asymmetric protocols for authentication. Although this approach is susceptible to key extraction attacks giving way to cloning. Thus, Physically Unclonable Functions have been proposed as a solution to this issue.

The next section exlpains the basics of PUFs and why they are a better solution to the problem. It also dives into how DRAMs can be used as PUFs and explains various methods of doing so.

II. Theory

This sections goes into the basics of PUFs and workings of DRAM PUFs.

A. PUFs

In practice most PUFs are characterized by challenges and responses. A challenge is the input given to the PUF. The PUF utilizes random manufacturing variations to convert the challenge to a response. The randomness makes the response unpredictable. And because these variations cannot be controlled, even by the manufacturer, the response is unclonable on any other device. This way they attain their property of being **physically unclonable**.

PUFs are mainly categorized into strong PUFs and weak PUFs based on the number of unique challenge response pairs (CRPs) they can generate. Larger the CRP database, the stronger the PUF. Both kinds of PUFs have appropriate supporting authentication protocols as discussed in the next sections.

Advisor: Hedongliang Liu, Associate Professorship of Coding and Cryptography

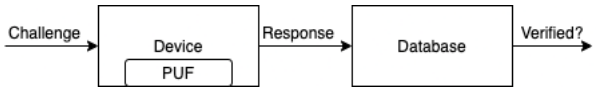

Fig. 1: PUFS

B. DRAM PUFs

There are different methods in which the process variations of DRAM can be leveraged to implement PUFs. The well-known methods are explained below.

Startup: In this method, the startup state of the DRAM is used for unique fingerprinting. As mentioned by Tehranipoor et al. "DRAMs have seemingly random startup values". This is because the bitlines on DRAM are precharged to vdd/2 which makes the sense amplifier equally likely to read a '1' or '0'. But, due to the process variations of manufacturing most bits have either a bias towards a '1' or a '0'. Hence, the startup values of the DRAM are randomized by the random process variations. It is noted though that some bits have a neutral skew/bias i.e. they randomly switch to '1' or '0' over multiple startups. These bits can be used for random number generation. [3]

Write Failures: In this method, process variations affecting the write reliability are exploited. Under normal operation, the write enable signal activates the bit lines i.e. connects them to the data bus which then overrides the bit line voltage. As a result, the DRAM cells would be overwritten by the value on the data bus. For normal operation the duty cycle of the write signal is chosen such that all DRAM cells can be successfully overwritten. But for PUF operation, the duty cycle is deliberately reduced which results in some DRAM cells being overwritten while others are not. If a cell has been overwritten or not depends on the manufacturing variations. The data written under PUF operation can be treated as the challenge and a subsequent read reveals the response affected by the variations. [4]

Refresh Pause: In this method, the refresh of the DRAM cells is paused to allow random decaying of the data stored in memory. [5] [6] [7] [8] For normal operation DRAM cells need to be refreshed i.e. the capacitors holding the charges need to be recharged due to leakage in the circuit. If the refresh is delayed some DRAM cells decay and thereby introduce a bit flip in the stored data. Under such operation, the DRAM can be used as a PUF where the data stored before the delaying refresh can be treated as the challenge and the one after the delay as the

response. An alternate method exploits data remanence properties of the DRAM, where the DRAM is completely switched off and then the cells are read. [9]

III. Implementation Objectives

In order to analyze the various presented methods, measurements have to be made. Almost all surveyed papers use either hamming distance and fractional hamming distance or jaccard index for measuring the properties of the PUFs. The usage of one over the other is debatable although Xiong et al claims jaccard index to be a better metric because it takes into consideration only the bits that flip as opposed to fractional hamming distance which weighs all bits of the CRP equally.

$$HD_{frac}(a,b) = \frac{HD(a,b)}{|a|} \quad \text{and} \quad J(a,b) = \frac{|a \bigcap b|}{|a \bigcup b|}$$

Uniqueness. Uniqueness in PUFs is measured by comparing responses for the same challenges on different PUF instances. Greater difference in responses indicates stronger uniqueness. The comparison is measured using either hamming distance(needs to be high) or the jaccard index(needs to be low). All surveyed work proves the uniqueness through experimentation on PUF instances of different DRAM modules, Sutar et al. goes a step further to analyze the uniqueness of different PUF instances within the same DRAM module.

Robustness. 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**. For robustness we need the HD to be close to 0 and the jaccard index to be close to 1. These measurements of robustness (same PUF instances) are further compared to the measurements of uniqueness (different PUF instances) to show **separability** or non-existence of false positives.

Robustness is also measured against temperature variations and aging. Almost all research work reports a drop in reproducibility as the temperature is increased. Although this drop is inferred differently in different conducted research. Hashemian et al which uses write failures for PUF operation claims the temperature induced variability to be not too profound thereby proving stability. Xiong et al which uses the refresh pause approach makes a relation between the temperature variation and decay rate and claims that the decay characteristics are unaffected. Xiong et al, for the above relation, suggests decreasing the refresh pause interval under increased temperature conditions. Sutar et al recommend adequate error correction and use dynamic thresholds (depending on temperature) . Aging effects on PUFs robustness is found to be negligible by most research papers.

Run-time Access. Depending on the method used, DRAM PUFs have restricted access. For run-time access of DRAM PUFs it is important that the PUF operation doesn't hinder the functioning of the OS and applications. The solution presented by Tehranipoor et al restricts the access to the DRAM PUF to an early boot stage because DRAM startup values cannot be accessed later. Whereas, write failures based DRAM PUFs offer the flexibility of run time access.

Refresh pause based DRAM PUFs can offer run-time access but with modifications to the kernel in some cases. When LPDDR DRAM is being used the PUFs can be accessed at run-time due to "Partial Array Self-Refresh" functionality. [5] When DRAMs are split by different memory controllers they can still provide run-time access by splitting the memory per controller for OS and PUF respectively. Special case arises when the DRAM being used has none of the above features, in such case, *memory ballooning* is used to reserve a DRAM region and *selective memory refreshing* is used to refresh selected regions by initiating a read. [6]

IV. Authentication

Most surveyed papers use the challenge response pairs for lightweight authentication of a client/prover (C) device towards a server/verifier (S) as described below.

Adversary and threat model. It is assumed that all network traffic between client and server is observable by attackers except during the enrollment phase. This is also known as the passive attacker model and here the attacker can observe the previous PUF measurements that were passed to the server.

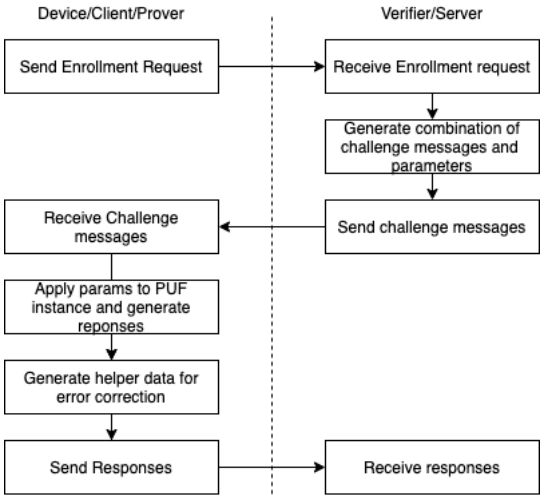

Fig. 2: Enrollment

Enrollment. The flow chart in Fig.2 depicts the enrollment process. The enrollment phase deals with generation of the CRP database. These challenge response pairs are then either stored on a trusted secure database or distributed to verifiers in a secure manner. It has to be made sure that different verifiers receive different parts of the CRP database, otherwise one can spoof to be the device to the other.

One Way Authentication. The flowchart in Fig.3 depicts the one-way authentication process. In short, the server matches the device's remeasured response against the CRP database. If matches, the device is authenticated

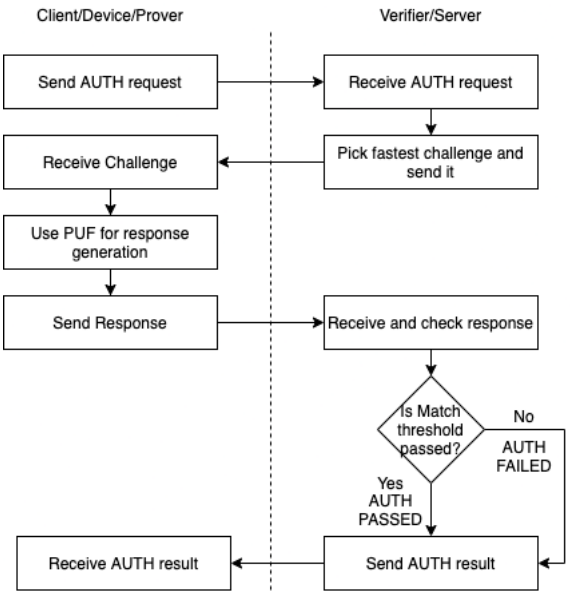

Fig. 3: 1-way Authentication

Mutual Authentication. This is an extension of the 1-way authentication which requires 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 to the server. This is done to test the knowledge of the server. The server verifies the device by counting the number of matched addresses and checking it against a threshold. The server then filters out the known flip addresses from the response and sends it to the device. The device verifies the server's response by matching it with its own response.

In both authentication protocols described above, it is important to note that:

- A CRP is never used twice to prevent replay attacks. Hence a large CRP database or a strong PUF is required for continued authentication.
- For fast authentication it is critical to pick a CRP with the fastest possible parameters(low refresh pause interval). This requires knowledge of the PUF characteristics beforehand.

These requirements pave the way for improvements in the usage of DRAM PUFs, *Parameter Reconfigurability* and *Characterization*.

Parameter Reconfigurability. The main objective of parameter reconfigurability is to increase the size of the CRP database. For this, different parameters that can change the response behaviour of the PUF have to be recognized.

Sutar et al in their work show the reconfigurability of the refresh pause interval in order to obtain new challenge-response pairs on the same PUF instance. A similar experiment is also conducted by Xiong et al and Schaller et al. After the CRPs of a particular refresh pause interval have been exhausted, the interval can be increased to a higher one depending on the number of "new" bit flips. Both these works also propose increasing the size of the PUF instance, in order to maintain the low refresh pause interval at the cost of DRAM space exhaustion.

Apart from the refresh pause interval and the size of the PUF instance, Sutar et al and Miskelly et al demonstrate the usage of different input patterns. Sutar et al also demonstrate the effect of peripheral bits, a.k.a wrapper pattern, surrounding the block of PUF instances on the response.

With these parameters at disposal, one can reconfigure the PUF instance to generate new sets of CRP pairs and thereby increase the size of the CRP database.

Characterization: For proper usage of the reconfiguration parameters, the challenge response behaviour must be analyzed before the authentication can take place. For achieving this the DRAM module has to be *characterized*. In simple terms, characterization is the process of trying all kinds of combinations of parameters and observing the difference in responses. [5] After such an exhaustive experimentation the best suitable parameters are selected and the CRP database is generated accordingly.

For example, Miskelly et al. experiment with an all '0' and all '1' input pattern and selects the input pattern with a higher number of bit flips. The responses for the two are biased due to the presence of true-cells and anti-cells. Apart from maximizing entropy, characterization is also used to increase robustness of the PUF. Tehranipoor et al present a solution which selects stable bits only when all neighbouring bits (spatially) are stable. Other works have also reported a 6x increase in authentication speed by optimally selecting a refresh pause interval. [5] [7]

Note that characterization is assumed to take place in a trusted and secure environment. Any information leakage of the characterization parameters reveals the characteristics of the PUF and hence make them susceptible to cloning.

V. Secure Channel Establishment

When a PUF cannot generate a large CRP database, it is used with a Fuzzy Extractor a.k.a Helper Data System [11] for secure key storage. A fuzzy extractor corrects errors in the PUF response [5] and enables regeneration of a cryptographic key. This way, the reproduced key can be used to setup a secure channel through symmetric protocols. Once a secure channel is established, authentication can be done through asymmetric protocols. A general construction of a fuzzy extractor contains:

- *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 $k' = k$

For using a fuzzy extractor with refresh based DRAM PUFs the decay behaviour must be discretized. Schaller et al present a viable method using refresh pause interval in which DRAM cells, during the characterization phase, are categorized into extremely fast cells **F** and slow cells **S**.

The idea is that is a refresh pause interval that is greater than the decay rate of fast cells and lesser than decay rate of slow cells would guarantee a bitflip in **F** cells and no bitflip in **S** cells. In such a scenario it would be safe to mark the **F** cells as '1' an **S** cells as '0', hence, discretizing the behaviour of DRAM PUFs decay.

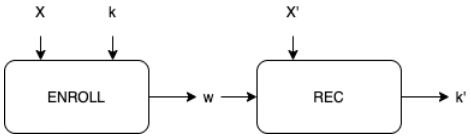

Fig. 4: Fuzzy Extractor

One of the earliest implementations of fuzzy extractors is the Code Offset Method. [10] COM uses an error correcting code at the core of its construction. The helper data $w = X \oplus Enc(k)$, where $Enc()$ is the encoding function of the error correcting code. The reconstructed key $k' = Dec(X' \oplus w)$, where $Dec()$ is the decoding function of the error correcting code. The important point to note here is that w plays the role of a syndrome. The error correcting code extracts the key k by correcting errors in X'. This is clear when the formula is rewritten as $k' = Dec(X' \oplus X \oplus Enc(k))$. Any remaining bits of the $X' \oplus X$, if there are any, is corrected by $Dec()$

Using such a fuzzy extractor would also require sufficient entropy in the input X. Schaller et al overcomes this problem by selecting equal number of **F** cells and **S** cells. The entropy in the input is required so that no information is leaked through the helper data, w.

Another version of the COM scheme, Syndrome-COM or SCOM, [10] 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. During reconstruction, The expression as shown in listings below evaluates to the syndrome of the error vector which is consequently corrected by the decoder.

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: $\hat{X} = X' \oplus SynDec(W \oplus Syn(X'))$

In this scheme the helper data $w = Syn(X)$, where $Syn()$ is the syndrome function. The reconstructed key $k' = SynDec(w \oplus Syn(X'))$.

Schaller et al in their work use the same construction of SCOM with a modification of using a key derivation function on the reconstructed input X'. This is done to ensure no bias from the PUF is transferred into the cryptographic key.

Other constructions, of HDSs make use of concatenation of two linear codes. [11]

$$C_1 = [n_1, k_1, d_1] \quad \text{and} \quad C_2 = [n_2, k_2, d_2]$$

In this construction, Code C_1 needs to maintain a high entropy, which directly translates into high cardinality or high k_1. And, Code C_2 needs to have a high error correction rate which directly translates to higher distance, d_2.

VI. Conclusion

This paper highlights why and how PUFs fulfill the demands of hardware-intrinsic authentication. It explains the well known methods of implementing PUFs on DRAMs and the advantages of DRAM PUFs. It outlines the objectives an implementation of PUF has to meet in order to be fit for use in authentication. It summarizes a simple lightweight protocol that can meet strict resource constraints for embedded processors. It also dives into the details of using PUFs for secure key storage using helper data systems (HDS) and summarizes different constructions for HDS.

References

[1] 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.

[2] 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.

[3] 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

[4] 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.

[5] 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

[6] 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

[7] 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.

[8] 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.

[9] 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.

[10] 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

[11] 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.