Replacing error correction by key fragmentation and search engines To generate error-free cryptographic keys from PUFs

Bertrand Cambou; Christopher Philabaum; Duane Booher
School of Informatics, Computing, and Cyber Systems
Northern Arizona University, Flagstaff, Arizona, USA
Bertrand.cambou@nau.edu; cp723@nau.edu; Duane.booher@nau.edu

Abstract: Rather than consuming computing power at the client level to correct erratic cryptographic keys generated by physical unclonable functions, search engines at the server level can independently uncover matching keys with no correcting schemes. However, when the defect density is high, the latencies associated with such search engines can be prohibitive. We are proposing a fragmentation scheme to significantly reduce the latencies, and to find matching cryptographic keys, even when the error rates are greater than 15%. A predictive model that anticipates latencies is proposed, and validated experimentally, with commercially available 32Kbyte SRAM-based physical unclonable functions, a microcontroller development board, and using a Window-10 PC as "server".

1- Introduction and background information

Physical Unclonable Functions (PUFs) [1-6], the fingerprints of microelectronic components, are subject to aging, temperature drifts, electromagnetic interactions, and various environmental effects. Typically, this produces 2-10% error rates between the initial readings of the PUFs that are stored as references, and the responses generated by these PUFs. Error correcting (ECC) algorithms need to correct 100% of all errors, to generate usable cryptographic keys from the PUFs, as single-bit mismatches are not acceptable [7-11]. ECC algorithms use helper data from the transmitting party, and iterative methods such as fuzzy extractors by the receiving party, which consume computing power, and could thereby leak information to the opponents. Response based cryptographic methods (RBC) eliminate the need to use error correction at the client level, as it generates cryptographic keys directly from the un-corrected responses of the PUFs [12-13]. This technology relies on the implementation of an efficient search engine, driven by a secure server interacting with the network of client devices, which finds the uncorrected PUF responses, rather than correcting them.

The RBC matching algorithm compares the cipher text sent by the client device with the cipher text computed by the server. The cipher text of the client device is computed with the key directly generated from the responses of the PUF. The cipher text of the server is computed with the key generated from the PUF challenges that are stored in a look up table as reference. If the two ciphers are different, the server generates cipher texts from all keys having a Hamming distance of one from the PUF challenges, and compares them to the cipher text transmitted by the client device from the responses. The process is iterated with keys having higher Hamming distances to find the matching cipher. The RBC algorithm is efficient when the error rates are small enough. In this work, we use Advanced Encryption Standard (AES) to generate the cipher texts from 256-bit long cryptographic keys. The latencies of the RBC search engines are too slow to process PUFs with error rates higher than 1%, which is the case of most PUFs without other correcting methods.

2- Fragmentation of the cryptographic keys

In order to enhance effectiveness at higher error rates, we propose the fragmentation of the keys generated by the PUF into sub-keys, also 256-bit long, which are padded with known random numbers.

In a fragmentation by two, the first sub-key is generated by keeping the first 128 bits of the 256-bit long key generated by the PUF, filled with a 128-bit long pad containing no errors. The last 128 bits of the PUF, also combined with a 128-bit long pad, generate the second sub-key. Statistically, the two sub-keys show error rates that are half those of full key error rates. The client device sends thereby two cipher texts generated by the two sub-keys. The RBC search engine can process the resulting two ciphers much faster to find the erratic key generated by the PUF. For example: If the 256-long keys contain 4 errors, and the latency of one encryption and matching cycle is τ_o, the average latency τ of the RBC search is given by:

$$\tau = \tau_o + \tau_o \binom{256}{1} + \tau_o \binom{256}{2} + \tau_o \binom{256}{3} + \tau_o \binom{256}{4}/2 \qquad \text{eq1}$$

$$\tau \approx \tau_o \binom{256}{4}/2 = 8.74\ 10^7\ \tau_o \qquad \text{eq2}$$

Assuming that the four errors are evenly distributed between the two sub-keys, the latency τ_2 of the RBC search, with a fragmentation by two is given by:

$$\tau_2 \approx 2 \times \tau_o \binom{128}{2}/2 = 8.13\ 10^3\ \tau_o \qquad \text{eq3}$$

Assuming that the first sub-key has three errors and that the second sub-key has one error, the latency τ_2 of the RBC search, with a fragmentation by two is given by:

$$\tau_2 \approx \tau_o \binom{128}{3}/2 = 1.7\ 10^5\ \tau_o \qquad \text{eq4}$$

Assuming that the first sub-key has four errors, and that the second key is error free, the latency τ_2 of the RBC search with a fragmentation by two is given by:

$$\tau_2 \approx \tau_o \binom{128}{4}/2 = 1.7\ 10^5\ \tau_o = 5.3\ 10^6\ \tau_o \qquad \text{eq5}$$

Assuming a normal statistical distribution of the four errors of the main key into 2/2, 3/1, and 4/0, the average latency τ_2 , can be computed from eq3, eq4, and eq5, and is much smaller than τ.

In a fragmentation by k ∈ {2, 4, 8, 16, 32}, the first sub-key is generated by keeping the first 256/**k** bits of the key generated by the PUF, filled with a (256-256/**k**)-bit long pad containing no errors. The subsequent ***k*** sub-keys are generated in a similar ways, also with a (256-256/**k**)-bit long pad. Assuming that the four errors are evenly distributed between the ***k*** sub-keys, the latency τ_k of the RBC search with a fragmentation by ***k*** is given by:

$$\tau \approx k\ \tau_0 \binom{256/k}{a/k}/2 \text{ , and } \quad D = a/256 \qquad \text{eq6}$$

a is the Hamming distance between the 256-bit long keys generated by the responses, and the keys generated by the challenges of the PUF stored by the server; ***D*** is the average PUF error rate. The final model assumes a normal distribution of the Hamming distance ***a*** among the ***k*** sub-keys. Shown in Fig.1 is an estimate of the average latencies of the RBC, using the model, as a function of the level of key fragmentation, and PUF error rates. Based on this model, an RBC with fragmentation by 8 is appropriate to handle PUFs with error rates in the 2 to 10% range, which is enough for most PUFs.

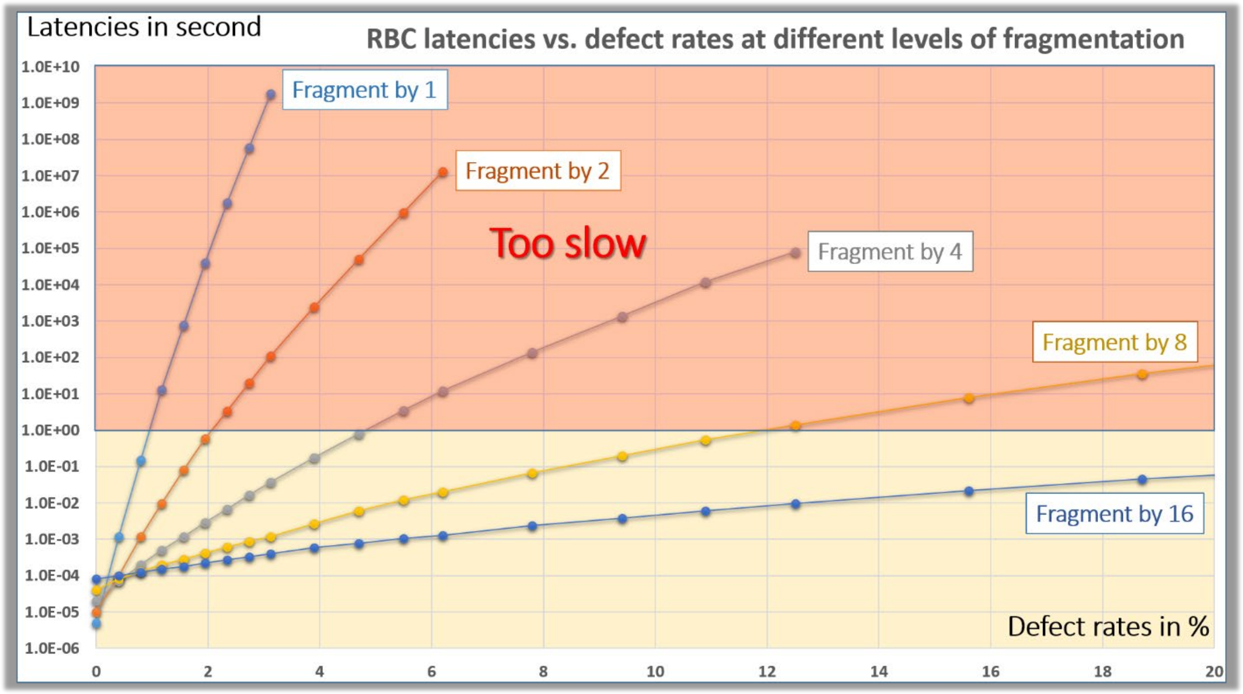


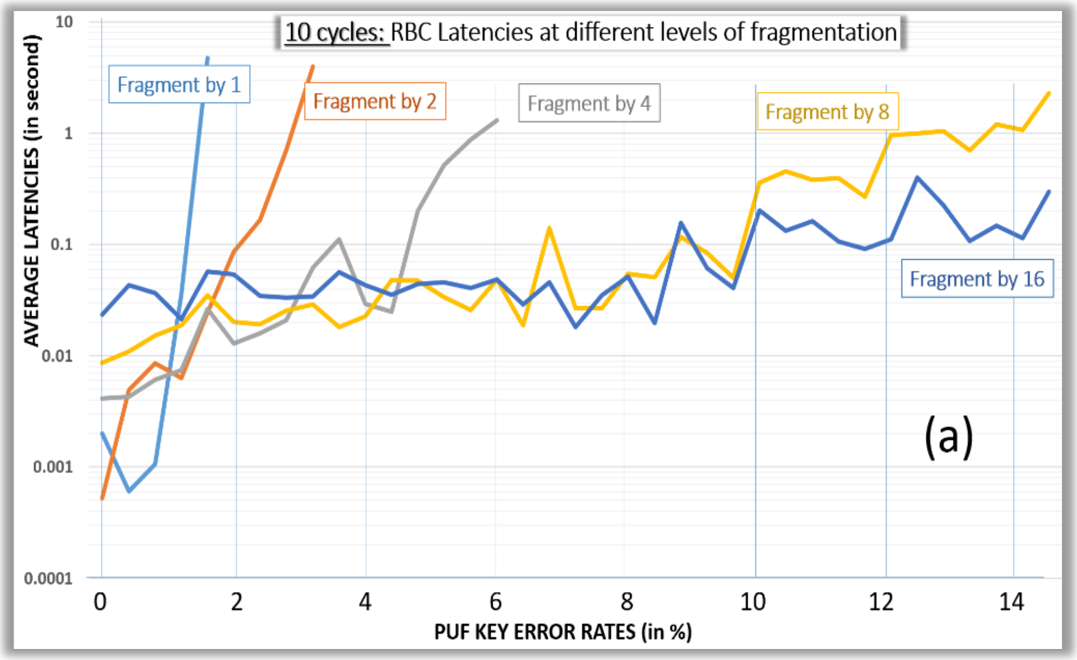
Figure 1: Modelling of RBC latencies

3- Experimental validation

To validate experimentally the effectiveness of the RBC with fragmentation, we used commercially available 32-Kbyte SRAM devices from Cypress semiconductor as PUFs. During enrollment, the SRAMs were submitted to power-off-power-on cycles, and the resulting patterns were stored in look up tables. About 50% of the flip-flop of the cells are mainly responding to such cycles as a "0" state, about 50% as a "1" state. The typical error rates of such PUFs were observed in the 2 to 10% range. The error rates can be reduced to 10^{-5} by cycling the SRAM multiple times during enrollment, and by eliminating the unstable cells. The error rates can also be increased to 20% by leveraging natural physical variations.

WiFire development boards from Microchip were used to drive the SRAM PUFs. These boards contain 200 MHz 32-bit RISC processors from ARM, ADC/DAC converters, 2MB flash, and 512KB RAM. The embedded software and cryptographic protocols to extract 256-bit keys from the PUFs were written in C, with software implementation of AES-256, and SHA-256. On the server side, Window 10 PCs powered by Intel I7 quad core processors were used, they can process an AES cycle in about five microseconds. The cryptographic protocol is randomly pointing at 256 cells in the PUF, every 2 seconds; the RBC search engine operate with various levels of fragmentation, and PUF defect rates. The results of the experimental work are summarized in Fig.2: in the first graph (a), the latencies are averaged over 10 queries at different levels of PUF defect rates; in the second graph (b), the latencies are averaged over 100 queries.

At very low defect densities, the quad-core processor of the PC faces delays due to the initialization cycles, and the management of the multi-tasking operations of the PC. However, on relative terms, the latencies at very low defect densities are proportional to the level of fragmentation: about 100ms without fragmentation, and respectively 200ms, 400ms, 800ms, 1600ms with a fragmentation of 2, 4, 8, and 16. It is always desirable to minimize the level of fragmentation to reduce computing power at the client level, so fragmentation algorithms are not needed with error rates below 1%.



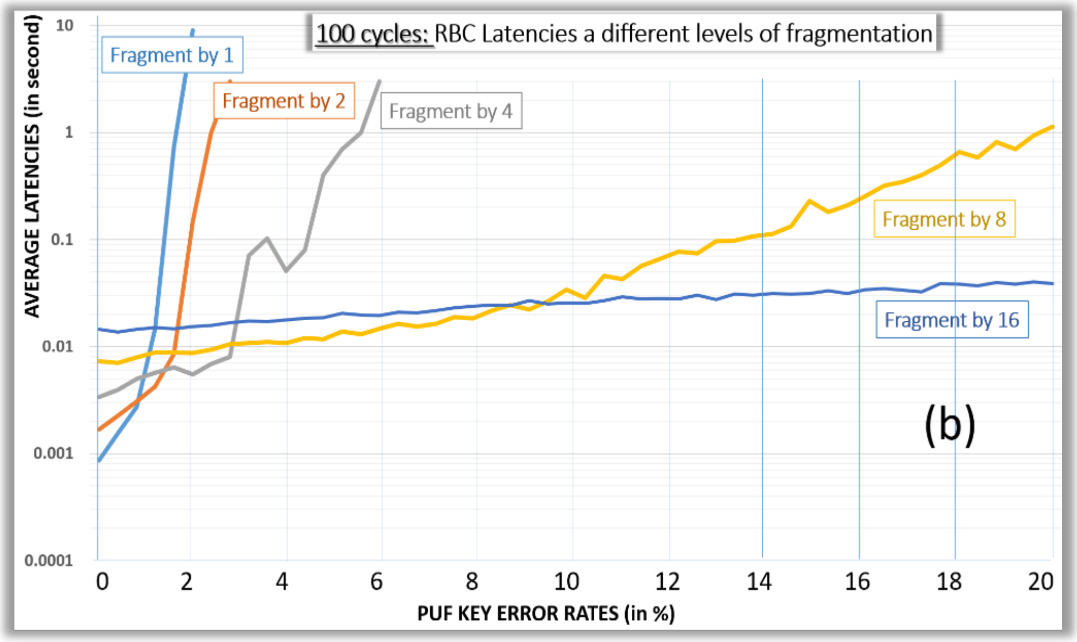


Figure 2: Measurement of RBC latencies – (a) 10 cycles ; (b) 100 cycles

Results also show that above 1% PUF error rate, both modelling and experimental data are consistent, and demonstrate the effectiveness of RBC with fragmentation. Shown in Table 1 are the back-to-back comparisons of the level of acceptable errors when the latencies of the PC are kept below one second. A fragmentation by eight sub-keys is optimum for the SRAM-based PUFs analyzed in this work. We are noticing that the experimental results with 100 cycles, and a fragmentation by 8, are more favorable than the modelling and the experimental results using only 10 cycles. This suggests that at higher defect rates, the effectiveness of parallelization of the quad core, which is hard to simulate, is about 50% higher than what we assumed in the model. However, the results are consistent at lower levels of fragmentation.

Table 1: Levels of PUF error rates requiring a latency of one second

Fragmentation		1	2	4	8	16
Levels of acceptable errors for 1 second latency (Modelling)		0.8%	2.1%	4.7%	12%	>25%
Levels of acceptable errors for 1 second latency (Experimental)	10 cycles	0.8-1.7%	1.8-2.5%	4.5-6%	11.5-13%	>25%
	100 cycles	1.5-2.0%	2.2-3.0%	5-6%	18-22%	>25%

4- Conclusion

The suggested method to fragment PUF-generated keys, has the potential to enhance the effectiveness of RBC algorithms, eliminate the need to use error-correcting methods, helper data, and thereby simplifies PUF-based cryptographic protocols, and enhances security. Below 1% error rates, a simple RBC algorithm without fragmentation is powerful enough, while a fragmentation by 8 can handle error rates in the 10 to 12% range.

We see the need to complete this research work with additional crypto-analysis to highlight possible weaknesses of the protocol under attack. We also see the need to validate the RBC algorithm with various PUFs, to perform additional experimental work to refine the models and comprehend initial latencies, and to predict and optimize parallel computing.

Finally, the deployment of the technology to industry will require the design of a custom secure microcontroller, protecting the client devices from side channel analysis.

Acknowledgements

The authors are thanking the students from NAU's cybersecurity lab who contributed to this work, in particular Vince Rodriguez, and Ian Burke.

References:

1) Holcomb, D. E., W. P. Burleson, and K. Fu. 2008. "Power-up SRAM state as an Identifying Fingerprint and Source of TRN". *IEEE Transaction on Computing, vol 57, No 11.*
2) Chen, T. I. B., F. M. Willems, R. Maes, E. v. d. Sluis, and G. Selimis. 2017. "A Robust SRAM-PUF Key Generation Scheme Based on Polar Codes". *In arXiv:1701.07320 [cs.IT].*
3) Prabhu, P., A. Akel, L. M. Grupp, W-K S. Yu, G. E. Suh, E. Kan, and S. Swanson. 2011. "Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations". *In 4th international conference on Trust and trustworthy computing.*
4) Cambou, B., and M. Orlowski. 2016. "Design of Physical Unclonable Functions with ReRAM and ternary states". *Cyber and Information Security Research Conference, CISR-2016, Oak Ridge, TN, USA.*
5) Korenda, A., F. Afghah and B. Cambou. 2018. "A Secret Key Generation Scheme for Internet of Things using Ternary-States ReRAM-based Physical Unclonable Functions". *In International Wireless Communications and Mobile Computing Conference (IWCMC 2018).*
6) Gao, Y., D. Ranasinghe, S. Al-Sarawi, O. Kavehei, and D. Abbott. 2016. "Emerging Physical Unclonable Functions with nanotechnologies". *IEEE, DOI:10.1109/ACCESS.2015.2503432.*
7) Maes, R., P. Tuyls and I. Verbauwhede. 2009. "A Soft Decision Helper Data Algorithm for SRAM PUFs". *In 2009 IEEE International Symposium on Information Theory.*
8) Boehm, H. M. 2010. "Error correction coding for physical unclonable functions". *In Austrochip-2010, Workshop in Microelectronics.*
9) Taniguchi, M., M. Shiozaki, H. Kubo and T. Fujino. 2013. "A stable key generation from PUF responses with a Fuzzy Extractor for cryptographic authentications". *In IEEE 2nd Global Conference on Consumer Electronics (GCCE), Tokyo, Japan.*
10) Delvaux, J., D. Gu, D. Schellekens and I. Verbauwhede. 2015. "Helper Data Algorithms for PUF- Kang, H., Y. Hori, T. Katashita, M. Hagiwara and K. Iwamura. 2014. "Cryptographie key generation from PUF data using efficient fuzzy extractors". *In 16th International Conference on Advanced Communication Technology, Pyeongchang, Korea.*
11) Becker, G. T., A. Wild and T. Güneysu. 2015. "Security analysis of index-based syndrome coding for PUF-based key generation". *In 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Washington, DC*.
12) Cambou, B., C. Philabaum, D. Duane Booher, and D. Telesca. "Response-based Cryptography Methods with Physical Unclonable Functions". *Future of Information and Communication Conference, FICC-2019.*
13) Cambou, B., "Unequally Powered Cryptography with PUFs for network of IoT", Spring Simulation Conference, SpringSim2019.