Where Technology Meets Security: Key Storage and Data Separation for System-on-Chips

Georg Sigl* †, Mathieu Gross*, and Michael Pehl*
*Department of Electrical and Computer Engineering, Technical University of Munich
Email: sigl@tum.de, mathieu.gross@tum.de, m.pehl@tum.de
†Fraunhofer Institute for Applied and Integrated Security, AISEC

***Abstract*—This article investigates the dependency between advances in chip technology, architectures, and security. Two major properties of secure systems are analyzed in this context: data separation of different applications and secure storage of cryptographic keys. We discuss first examples for compromising data separation, e.g. the Rowhammer attack on modern DRAMs, enabled by the sensitivity of shrinked DRAM cells for crosstalk effects, or Meltdown and Spectre attacks using cache side channels. These attacks show the dependency between data separation and advances in technology and architecture. Even more powerful attacks exploiting bus and network-on-chip traffic are possible. Another area where technology meets security is the storage of cryptographic keys. New technologies offer new ways to realize non-volatile memory (NVM) for secret data storage and to implement physical unclonable functions (PUFs), which generate the key during system start and do not store it permanently in NVM. To enable good PUFs, technology and security people should work together as early as possible in the development phase, since PUFs must be characterized carefully. Ideally a PUF module is provided as a characterized and reliable security primitive in the design library. If we manage to take security already into account in early technology development phases and during architecture definition, we will get more secure systems-on-chip in the future.**

I. Introduction

Advances in silicon production technology over the last 50 years have driven the complexity of modern silicon chips towards incredible size. Multi-Processor Systems on Chip (MP-SoC) allow complex computations even on a single chip integrated in an embedded system. Memory hierarchies comprising of multiple levels of caches enable high performance memory accesses in order to feed all the processors with enough data, i.e. program code and input data. In many applications the compute power is however not really needed. This drove another trend: combination of multiple applications implemented on single processor micro-controllers into one MP-SoC using a multitasking environment managed by an operating system or a hypervisor. This increased the complexity even further and created the problem of separating the data of these different applications. There is a long lasting development of solutions for separating different tasks on a logic level by operating systems. The kernel is responsible for task separation and manages access rights. Due to the high complexity of operating systems with millions lines of code, many implementation weaknesses, e.g. the prominent buffer overflow vulnerabilities [5], in the software have been used to break separation. One approach to overcome this was the invention of microkernels, which are formally proven to be correct [18]. While the problem of separation through software is still not fully solved another trend came up in recent years: Technology advances created new “side channels” which compromised separation. Examples discussed in the next section are the rise of Rowhammer attacks [27], [17] enabled by DRAM shrinking, and the recently published combined cache/prefetch attacks Meltdown and Spectre [23], [19]. Another not yet fully exploited attach path are new Network-on-Chip (NoC) communication structures. More on these new threats against separation will be discussed in section II.

Another trend in modern systems is ubiquitous connectivity. Nowadays, all MP-SoCs and even small micro-controllers are connected to networks and often to the internet. Connectivity raises the need for secure communication usually implemented using cryptography with secret keys. The secure storage and use of a key in an MP-SoC requires strong separation as discussed in the previous paragraph. Additionally keys need to be protected against readout through side channel attacks and tampering through reverse engineering. Against the latter advances in technology usually help to protect keys, but advances in silicon analysis methods still pose a serious threat. There are two ways to provide keys on a MP-SoC: (i) The key can be stored in secure non-volatile memory and (ii) the key may be derived from a physical property using a physical unclonable function (PUF), e.g. [14], [6]. Secure storage is difficult to realize. Usually fuse or anti-fuse technology is used to store a unique device secret. If the location of those fuse elements is detected, the stored information can be read. PUFs do not store the key on a chip. They generate the key during start-up of the chip by evaluating physical structures which are depending on unique manufacturing variations. For both secure storage and PUFs advances in silicon technology can be used to build new more secure solutions. This topic will be elaborated in section III.

To summarize this introduction, the main security requirements for modern complex MP-SoCs are:

- Separation of tasks to protect one task from interference with another potentially compromised one.
- Secure secret key storage in order to keep information confidential.

Both topics will be discussed in the next two sections by presenting attacks and solutions. The goal of this contribution is to motivate people from architecture and technology to work together with security experts in designing more secure MP-SoCs in the future.

II. Separation

There have been many attacks against and solutions for the separation problem in the software community. Already the first mainframe computers had the challenge to separate different programs and provide an isolated platform for each software application. This lead to concepts like user/kernel space separation in operating systems and virtualization with hypervisors and microkernels. Other solutions like ARM TrustZone [22] or Intel SGX [2] try to define security islands which allow protected processing of confidential information. While these techniques have been improved over time new attack paths were uncovered recently: Caches and the DRAM memory. Despite the targeted separation caches and memory modules are still shared between security island and normal platform. This lack of separation in hardware has been used to mount a cache attack on SGX [7] and on TrustZone from the normal world [42].

The Rowhammer attack on DRAM disproved one basic assumption required for separation of data in memory, i.e. information stored in a memory cell cannot be changed through accessing a different memory cell. The basic mechanism of this attack is to access a target line as often as possible. Through these accesses to the target neighboring victim cells are affected through crosstalk. The storage capacitor of the victim cell is discharged with every access to the target. After a while the discharge is sufficient to flip the information stored in the victim cell. This effect was well known to the DRAM developers and prevented through timely refresh of all DRAM contents. Unfortunately the refresh time was calculated under the assumption of normal usage behavior and did not take attackers into account. The lessons learned from this attack must be that even abnormal behavior has to be taken into account if new memory technologies and architectures are developed by technology and circuit design people usually not thinking about security.

Another event which shocked the software community were Meltdown and Spectre. Historically processor designers had to solve the problem of relatively slow memory accesses compared to fast running CPUs requiring huge amounts of code and data. In order to overcome this limitation very complex memory hierarchies with many levels of caches have been introduced. It was argued very soon after the discovery of side channel attacks [20] that cache behavior [16], [31] can be used as a side channel to recover secret information. To explain the basic principle of a cache attack let's take a simple example: Assume a task A uses an array covering 256 different cache lines of data in a MP-SoC with direct mapped cache. The first access to each array entry belonging to a different cache line takes longer since data are not yet in the cache, while the second and all following accesses are faster. If another task B wants to know which data have been accessed by task A, it could do the following: After task A has been suspended from execution task B starts to access all cache lines by reading the same data as task A. By measuring the time for reading the data, task B can exactly determine which cache line was accessed by task A. Afterwards task B can overwrite the data in the cache by loading its own data in the 256 cache lines used by task A. A typical example where this kind of attack can be used are software libraries implementing symmetric ciphers containing substitution boxes usually realized as memory arrays. By accessing (task A) the array with addresses related to secret keys, the attacker (task B) can reveal this address information by measuring which cache lines have been used by task A as described above.

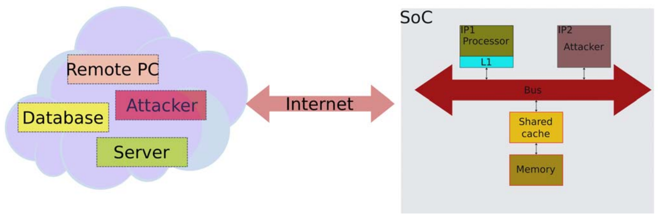

Fig. 1. Remote attack on MPSoC

Meltdown and Spectre use this cache leakage behavior in a different way. They combine it with another mechanism introduced to overcome the problem of slow memory: prefetching and speculative code execution. Modern CPUs try to predict a program's behavior in order to fetch data into the cache before they are first time used. This requires a prediction of the future program flow, e.g. which branches are taken or not. During speculative execution the program accesses code and data and fetches them into the cache. If the prediction turns out to be wrong, the changes in the CPU resources are reverted but not the changes in the complex cache hierarchy. The Meltdown attack uses this effect. An attacker writes an attack program that performs data accesses in a loop, which causes the predictor to assume that this loop is usually followed. After a while the data access pointer is modified in a way that it accesses data far out of the loop index range anywhere in the memory. The predictor will perform this access. Afterwards the attack program uses this data as an index for an array access and stops execution. After a while the CPU will recognize that the speculation was wrong and the attack program performed an invalid memory access. It will revert all changes but not in the cache and stop the attack program. Then the attacker can start another attack program which measures the access times to the memory array. One access will give significantly lower access time. The index of this access corresponds to the data read by the attack program. With this mechanism the complete memory content can be read by the attack program thus rendering useless all separation mechanisms.

This kind of attacks will become more and more elaborated in the next years. One example of a future trend is the analysis of bus or network-on-chip traffic as illustrated in the attack

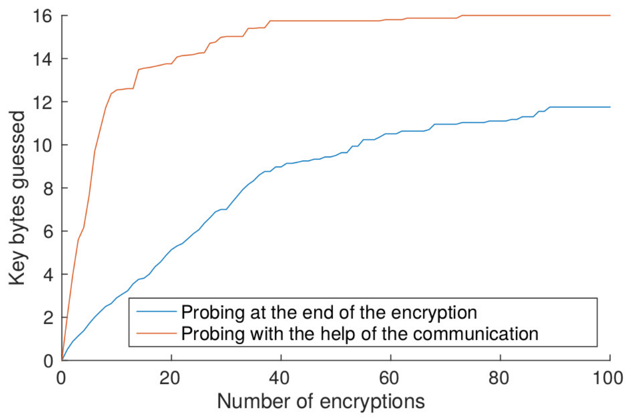

Fig. 2. Comparison of the classical and improved attack

setting of [32]. In this setting, it is assumed that the MP-SoC is connected to the internet (Figure 1).

After the download of malicious software from the internet, the attacker can infect one application running on one of the cores (IP2 in Figure 1). This enables the attacker to exploit the leakage of shared resources like buses, routers and caches in a side channel attack scenario. In certain scenarios, exploiting the resource contention of buses and NoC routers can lead to a significant improvement of cache attacks. This is for instance the case for a cache attack targeting a speed optimized implementation of AES. This implementation relies on the use of T-Tables for computing AES rounds. A well known cache attack [36] on such an implementation targets the first round of AES. In that round the indices x_i used for accessing the T-Tables are key dependent: $x_i = p_i \oplus k_i$ where P=$\{p_0$ ... $p_{15}\}$ represents the plaintext and K=$\{k_0$ ... $k_{15}\}$ the key. The technique used for observing the accessed cache lines is similar to the one described in previous section. Ideally the attack on the first round can observe the cache already after the first round is finished. However the remote attacker can usually not detect it and sees cache accesses in other rounds as additional noise. Our work [32] enhances the approach in [36] by using NoC congestion in order to determine the end of the first round. After triggering an encryption on the victim core, the attacker can continuously read a specific cache line and observe variations in the completion time of the previous read request. Increase in the completion time of that request occur due to contention on the interconnect. After observing enough victim accesses, the attacker is sure that the first round of AES is completed and can after that observe the cache access pattern of the victim, with the technique described in previous section. The improved attack contributes to reduce the number of traces required for the attack as shown on Figure 2.

This enhanced attack is particularly interesting for implementations using the SBOX for the first, second and last round, while keeping the T-Tables for the intermediate rounds. The SBOX implementations are less sensitive against cache attacks, because in comparison to the T-Tables, one SBOX only occupies 4 cache lines (assuming a cache line size of 64 Bytes). Therefore the observable leakage is very low with the attack technique of [36]. If the attacker is probing the cache at the end of the encryption, it is very likely that all SBOX specific cache lines are accessed, which renders the attack impossible. However, by probing the cache earlier with the enhanced attack, cache traces become exploitable.

III. Secure Key Storage

The previous section focuses on separation and presents security threats while the MP-SoC is operating. Another issue is to protect the MP-SoC while it is not operating against changes and information extraction. For these security goals we need to store cryptographic keys securely. There are two possibilities for providing keys in embedded systems. One is to program keys in a protected environment into NVM. Alternatively, keys can be derived from physical structures which vary due to manufacturing variations but provide enough stable information to derive a key. These physical structures are called Physical Unclonable Functions (PUFs). As long as NVM is available, storing keys there seems to be easy. But NVM has drawbacks when used for storing keys: The most severe one is probably the threat of reading out the complete memory with chip analysis tools. ROM – which suffers from the additional problem that a change in the key also requires modifications in the masks while manufacturing the chip – can be read out by etching techniques and optical inspection, e.g. [21]. Other memory technologies like fuses can be read out by a laser scan and modified using focused ion beams (FIBs) [9]. It can be expected, that anti-fuses can be attacked in a similar way. Also, memory solutions like EEPROM are vulnerable against attacks, e.g. using scanning electron microscopy [3]. The inherent problem in all these technologies is that NVM is non-volatile. I.e. the secret is available while the chip is powered down and no protection mechanism like sensors, which can be used to detect an attempt to attack a chip, can hinder an attacker from tampering with the device. Given the possible attacks, it can be concluded that the best way to prevent the attacker from reading out a key from NVM is to permanently keep at least the countermeasures against invasive attacks powered and to destroy all security relevant content, as soon as an attack is detected. But keeping the chip powered is expensive and especially not feasible in low cost devices.

In our opinion, therefore, the best solution for the problem of key storage is to use PUFs. PUFs have in general the great benefit, that they do not store the secret permanently. They rather measure chip-individual properties, which can be seen as a chip's fingerprint, on demand. After measuring, the PUF response is a secret derive from the otherwise not observable variations. Therefore, the secret is not available if the chip is not powered. However, PUFs also offer additional advantages. Different from most NVM technologies, PUFs are build from standard components, which are compatible with a standard manufacturing process and are therefore easy and cheap to integrate. Also, the secret provided by a PUF is inherently connected to the hardware, since it is derived from chip-individual features. Note, that this enables the use of PUFs – especially of PUFs which are configurable by a challenge like [33], [4], [10] – in lightweight authentication protocols, e.g. [26], [41], [40], even if this fact is not further considered in this work. Obviously, a PUF can be made most secure, if it is especially designed for the purpose of being a PUF. But even if there is no dedicated PUF provided in a chip, PUFs offer a way to add a certain level of security, since available standard components like SRAM cells can be used as PUFs to derive and store a secret key, e.g. [38].

However, PUFs also have some not yet completely solved problems. One main point in terms of key storage is noise. Since PUFs measure process variations on demand, the measurement result, and thus the derived secret, varies from measurement to measurement. But for a secret to be used as a key, no variations, i.e. errors, are acceptable. Fortunately, the last years brought some progress in the field of error correction for PUFs, like e.g. in [13], [12], [11].

The second important issue of PUFs is ensuring their quality. By quality of a PUF we refer to mainly two aspects: i) The reliability of PUFs which requires exploration of the effect of noise and environmental conditions on PUFs as well as aging effects. Here, security experts require the support of technology and circuit design experts to improve the PUF quality. ii) The unpredictability, i.e. the security level of a PUF. An example, how the unpredictability of a PUF can be affected is visible in the analysis of the ring oscillators (ROs) from [24] shown in Fig. 3 which was presented in [39]. The ROs are placed on an FPGA in 16 times 32 slices with one RO per slice. To build an RO PUF out of such an array, two RO frequencies are compared and depending on which frequency is faster, a logical 1 or 0 is output as the PUF response. To analyze the data, a covariance matrix of the frequencies can be computed. It describes statistical dependencies (covariances) between each two RO frequencies, which can impact the response quality. PCA orthogonalizes the covariance matrix and orders the orthogonal components with respect to the contribution to the variance in the analyzed data set. Fig. 3 shows a plot of the first eight principal component vectors plotted according to the spatial layout of the ROs in the PUF.

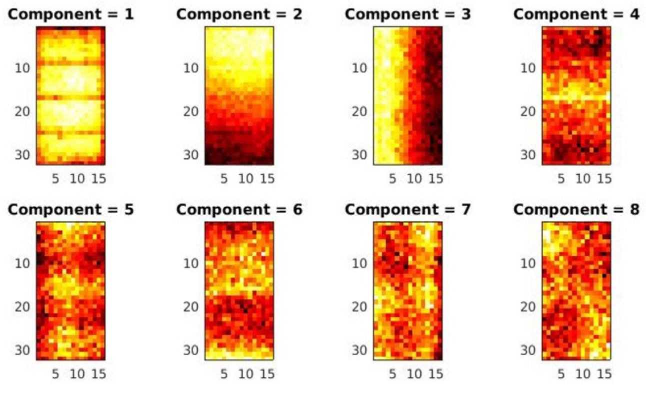

Fig. 3. First principal components derived from the data in [24].

The structure in the first component and both the gradients in the second and third component are clearly visible. While the structure in the first component is well explained by the placement and the environment on the FPGA, the gradients in the second and third component are best explained by a speed gradient in the chip caused by the manufacturing process. Pattern in PCA components are alarm indicators for PUF quality weaknesses which could arise through regular structures in the design or through techniques used for manufacturing. How it can effect the quality is shown in the next paragraph for an RO PUF which compares always two neighbor ROs in horizontal direction. It will be shown, that the speed gradient visible in this direction directly affects the generated secret.

The patterns found in the principal components indicate general patterns in all the devices, but do not state how strong the influence is on a specific device. Furthermore, it is not visible if, e.g., the gradient causes a speed increase from the left to the right or vice verse. This information is however available in coefficients computed during the PCA for each device. The correlation of these coefficients per device with the corresponding number of PUF responses evaluated to logical 1 is a measure on how much an attacker can learn about the secret by knowing the principal components. The significant correlations between the coefficients per device and the number of 1s in the responses for our example are plotted in Fig. 4.

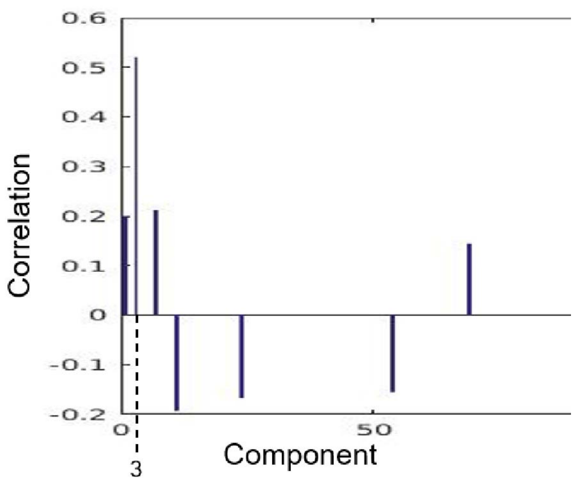

Fig. 4. Correlations found between the number of responses with logical value 1 and the influence of the principal components on the original data.

The third component (gradient in horizontal direction) gives the largest correlation in our case while the second component is not significantly correlated. I.e. it can be concluded that the number of logical 1s derived from the PUF strongly depends on the horizontal gradient. This is not too surprising: The responses from the RO PUF are derived from comparing horizontally adjacent ROs and the result is evaluated to logical 1 if the right RO is faster. This way, a gradient which causes an increase of the frequencies from the left to the right/right to the left causes a higher/lower probability for a logical 1 in the response. But the example shows that an attacker who knows about the gradient on a certain device, e.g. by observing some non security critical parts of the circuit, can have a significant advantage in predicting the PUF response and demonstrates the urgent need for analyses of the unpredictability of a PUF.

The unpredictability analyses can only be computed using statistical analysis methods. However, the methods need to be adapted to the needs of PUFs. This is required since, first, there are many different dimensions like the PUF behavior over challenges, over the position on a chip, over many chips at the same position, etc. to be considered; these dimension must also be combined in metrics for PUF quality in a reasonable way. Second, to get evaluations with a certain confidence, many methods would require thousands of PUF instances to be analyzed. So special approaches need to be developed or adapted for the needs in the PUF context. Currently, mainly methods which consider bias in certain bits of a PUF response are used [25], [15]. However, also correlations between PUF bits can be an issue. First metrics were already presented to consider also correlations, i.e. dependencies, between PUF bits, e.g. [37]. The availability of metrics can help designers to counteract possible issues in a PUF which might lead to predictable secrets, while security experts can help to interpret the results of the metrics and to assess the critically of problems found.

For new technologies, currently a major problem is that frequently no complete chips or large numbers of cells are build. Instead, models are build from single cells and are used to demonstrate the behavior of complete structures, neglecting correlation effects as well as the influence of placement and routing. For PUFs this is a severe problem, since it hinders a sufficient analysis of new PUF structures in new technologies. Still, new technologies, like memristor based PUFs, e.g. [1], provide promising properties: Many of these elements seem to offer unpredictable but reliable behavior that can be exploited. But to evaluate the true quality of PUFs in such new technologies, it is required to analyze the primitives in hardware. E.g., since memristors are NVM it has to be ensured that an attacker cannot read out the memristors in the power down state. But it is currently not clear, if a read out is feasible, or if countermeasures, like programming the memristor to a certain value after legitimately reading the key and reproducing the state which reveals the key only on demand by a certain strategy, are required and possible.

The urgency of such analyses can be easily seen when considering attacks on PUFs. While there is currently no attack known which is able to read out the secret provided by a reasonable good PUF as long as it is not powered, attacks from the last years have shown that it is possible to attack a PUF while the secret is derived: It was demonstrated, that the photon emission caused by switching activities of transistors when the PUF is operating, can be used to extract the secret, e.g. [8], [34]. For ring oscillator PUFs the frequencies can be measured and the secret can be extracted under certain circumstances [28], [29]. Since the PUF response is noisy, also the error correction is required, which can also be attacked [30], [35]. In general it can be assumed that an attacker might find for every PUF a point, which can be attacked. However, if attacks are known, they can be prevented by appropriate countermeasures.

Especially for new technologies and PUF designs it is an urgent need to consider this insight that known attacks can be prevented early in the development of such new structures. To analyze the strength of PUFs in new technologies against hardware attacks, and to develop solutions which counteract currently known attacks on PUFs and outperform currently known PUF solutions, experts from security, circuit design, and technology need to work hand in hand.

IV. Conclusion

We have shown that a mix of progress in technology and architecture can create significant vulnerabilities for two important security targets: separation of applications and secure key storage. Separation can be violated through memories, which change contents of not accessed neighbor areas as shown in the Rowhammer attack. An example for an architectural weakness creating a separation problem are Meltdown and Spectre, which use the cache as a side channel to transport information. We forecast that other shared resources will be used as well unless carefully protected during the design phase already.

For secure key storage PUFs provide a nice opportunity to generate a key only at start of operation, i.e. preventing key extraction through tampering. However designing high quality PUFs requires good interaction between security and technology experts. Proper metrics need to be defined to assess the quality of PUF designs and the impact of systematic technology variations on the key quality. The design of a PUF has to consider both these metrics and potential technology or manufacturing weaknesses. Again this requires early interaction between design, technology, and security experts.

Acknowledgment

This work was partly funded by the Fraunhofer High Performance Center for Secure Connected Systems Munich and by the German Research Foundation (DFG) under Grant SI 2064/1-1.

References

[1] D. Arum, S. Manich, R. Rodrguez-Montas, and M. Pehl. Rram based random bit generation for hardware security applications. In *2016 Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–6, Nov 2016.

[2] Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptology ePrint Archive*, 2016:86, 2016.

[3] Franck Courbon, Sergei Skorobogatov, and Christopher Woods. Reverse engineering flash EEPROM memories using scanning electron microscopy. In *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, pages 57–72, 2016.

[4] Meng day (Mandel) Yu. Recombination of physical unclonable functions. In *GOMACTech-10 Conference*, 2010. SUM-PUF original publikation.

[5] Common Weakness Enumeration. CWE-119: Improper restriction of operations within the bounds of a memory buffer. https://cwe.mitre.org/data/definitions/119.html, Page Last Updated: March 29, 2018.

[6] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Delay-based circuit authentication and applications. In *Symposium on Applied Computing (SAC)*, 2003.

[7] Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. Cache attacks on Intel SGX. In *Proceedings of the 10th European Workshop on Systems Security*, EuroSec'17, pages 2:1–2:6, New York, NY, USA, 2017. ACM.

[8] Clemens Helfmeier, Dmitry Nedospasov, Christian Boit, and Seifert Jean-Pierre. Cloning physically unclonable functions. In *Proceedings of the IEEE Int. Symposium of Hardware-Oriented Security and Trust*. IEEE, June 2013.

[9] Clemens Helfmeier, Dmitry Nedospasov, Christopher Tarnovsky, Jan Starbug Krissler, Christian Boit, and Jean-Pierre Seifert. Breaking and entering through the silicon. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, pages 733–744, New York, NY, USA, 2013. ACM.

[10] R. Hesselbarth and G. Sigl. Fast and reliable puf response evaluation from unsettled bistable rings. In *2016 Euromicro Conference on Digital System Design (DSD)*, pages 82–90, Aug 2016.

[11] M. Hiller, M. D. Yu, and G. Sigl. Cherry-picking reliable puf bits with differential sequence coding. *IEEE Transactions on Information Forensics and Security*, 11(9):2065–2076, Sept 2016.

[12] Matthias Hiller, Leandro Rodrigues Lima, and Georg Sigl. Seesaw: An area-optimized fpga viterbi decoder for pufs. In *Euromicro Conference on Digital System Design (DSD)*, pages 387–393. IEEE, 2014.

[13] Matthias Hiller, Meng-Day (Mandel) Yu, and Michael Pehl. Systematic low leakage coding for physical unclonable functions. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 155–166, 2015.

[14] D.E. Holcomb, W. Burleson, and K. Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *Proceedings of the Conference on RFID Security*, 2007.

[15] Yohei Hori, Takahiro Yoshida, Toshihiro Katashita, and Akashi Satoh. Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs. In *Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs*, RECONFIG '10, pages 298–303, Washington, DC, USA, 2010. IEEE Computer Society.

[16] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side channel cryptanalysis of product ciphers. In Jean-Jacques Quisquater, Yves Deswarte, Catherine Meadows, and Dieter Gollmann, editors, *Computer Security — ESORICS 98*, pages 97–110, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[17] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. *SIGARCH Comput. Archit. News*, 42(3):361–372, June 2014.

[18] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. sel4: Formal verification of an os kernel. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 207–220, New York, NY, USA, 2009. ACM.

[19] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. http://arxiv.org/abs/1801.01203, 2018.

[20] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113, London, UK, UK, 1996. Springer-Verlag.

[21] Oliver Kömmerling and Markus G. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, WOST'99, pages 2–2, Berkeley, CA, USA, 1999. USENIX Association.

[22] ARM Limited. Arm security technology building a secure system using trustzone technology. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf, 2009.

[23] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. http://arxiv.org/abs/1801.01207, 2018.

[24] A. Maiti, J. Casarona, L. McHale, and P. Schaumont. A large scale characterization of ro-puf. In *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 94–99, June 2010.

[25] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A systematic method to evaluate and compare the performance of physical unclonable functions. Cryptology ePrint Archive, Report 2011/657, 2011. http://eprint.iacr.org/2011/657.

[26] Mehrdad Majzoobi, Masoud Rostami, Farinaz Koushanfar, Dan S. Wallach, and Srinivas Devadas. Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching. In *International Workshop on Trustworthy Embedded Devices (TrustED)*, pages 33–44, 2012.

[27] Thomas Dullien Mark Seaborn. Exploiting the dram rowhammer bug to gain kernel privileges. In *Black Hat 2015*, 2015.

[28] Dominik Merli, Johann Heyszl, Benedikt Heinz, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Localized electromagnetic analysis of RO PUFs. In *Proceedings of the IEEE Int. Symposium of Hardware-Oriented Security and Trust*. IEEE, June 2013.

[29] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Semi-invasive EM attack on FPGA RO PUFs and countermeasures. In *6th Workshop on Embedded Systems Security (WESS'2011)*, Taipei, Taiwan, October 2011. ACM.

[30] Dominik Merli, Frederic Stumpf, and Georg Sigl. Protecting PUF error correction by codeword masking. Cryptology ePrint Archive, Report 2013/334, 2013. http://eprint.iacr.org/2013/334.

[31] Dan Page. Theoretical use of cache memory as a cryptanalytic side-channel. In *IACR Cryptology ePrint Archive*, volume 169, 2002.

[32] J. Sepulveda, M. Gross, A. Zankl, and G. Sigl. Exploiting bus communication to improve cache attacks on systems-on-chips. In *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 284–289, July 2017.

[33] Gookwon Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *ACM/IEEE Design Automation Conference (DAC)*, pages 9–14, 2007.

[34] Shahin Tajik, Enrico Dietz, Sven Frohmann, Jean-Pierre Seifert, Dmitry Nedospasov, Clemens Helfmeier, Christian Boit, and Helmar Dittrich. Physical characterization of arbiter pufs. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, pages 493–509, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[35] Lars Tebelmann, Michael Pehl, and Georg Sigl. EM side-channel analysis of bch-based error correction for puf-based key generation. In *Proceedings of the 2017 Workshop on Attacks and Solutions in Hardware Security*, ASHES '17, pages 43–52, New York, NY, USA, 2017. ACM.

[36] Eran Tromer, Dag Arne Osvik, and Adi Shamir. Efficient cache attacks on aes, and countermeasures. *J. Cryptol.*, 23(1):37–71, January 2010.

[37] F. Wilde, B. M. Gammel, and M. Pehl. Spatial correlation analysis on physical unclonable functions. *IEEE Transactions on Information Forensics and Security*, 13(6):1468–1480, June 2018.

[38] Florian Wilde. Large scale characterization of sram on infineon xmc microcontrollers as puf. In *4th Workshop on Cryptography and Security in Computing Systems (CS2 2017) HIPEAC17*, Stockholm, Sweden, Jan 2017.

[39] Florian Wilde, Matthias Hiller, and Michael Pehl. Statistic-based security analysis of ring oscillator pufs. In *2014 International Symposium on Integrated Circuits (ISIC), Singapore, December 10-12, 2014*, pages 148–151, 2014.

[40] M. D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede. A lockdown technique to prevent machine learning on pufs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems*, 2(3):146–159, July 2016.

[41] M. D. Yu, D. M'Rahi, I. Verbauwhede, and S. Devadas. A noise bifurcation architecture for linear additive physical functions. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 124–129, May 2014.

[42] Ning Zhang, Kun Sun, Deborah Shands, Wenjing Lou, and Y. Thomas Hou. Truspy: Cache side-channel information leakage from the secure world on ARM devices. *IACR Cryptology ePrint Archive*, 2016:980, 2016.