Machine Learning for Embedded Security:
Identification and Mitigation of Side-Channel Attacks within the Internet of Things.

Graham Claffey, Student Member*
18296661
University of Limerick, Co. Limerick, Ireland
Email: 18296661@student.ul.ie

Abstract – The goal of this article is to raise awareness related to security vulnerabilities known as side-channel attacks (SCAs) and bring to light the methods used to detect and prevent these vulnerabilities. The most infamous of these discovered in 2017 known as Spectre and Meltdown, led to multiple variants of each being identified after initial discovery. Recently in 2019 a new type of side-channel attack has surfaced, dubbed Zombieload. These type of vulnerabilities exploit hardware design flaws present on almost all modern CPUs and microprocessors (Intel, AMD, and ARM etc.). These vulnerabilities are not limited to Server, desktop or laptop devices, even embedded devices are also vulnerable. With the Internet of Things (IoT) on the raise, more devices are connected to the Internet every year which increases the attack surface for anyone with malicious intent. We will see the problems that are presented with SCAs extent not just to the application layer but can effect Operating Systems (OS), Real-Time OS (RTOS) and all the way down to the hardware level.

How do we prevent these security issues from affecting smart embedded devices? The answer might be with the use of Machine Learning (ML) to identify and mitigate these issues as they arise.

I. INTRODUCTION

With the steady progression of system on chip (SoC) technologies, we have seen an increase in the number of transistors that can fit onto a single chip or chiplet. This has led to faster, more power efficient integrated circuits (ICs) within all aspects of the technology industry. The embedded world is no exception to this, with embedded devices leveraging these advancements. It is not uncommon with embedded devices to come equipped with multi-core processors. Due to the rise of the Internet of Things (IoT), the demands for cyber-physical systems has also increased dramatically within a multitude of industries, including agriculture, manufacturing, automobile, avionics, medical etc. IoT has also become a large part of the everyday persons' life (e.g. Smart homes, phones, watches). The increased demand of more powerful ICs' and the need for internet connectivity in the IoT domain, we also need to ensure security can meet this expansion. To keep up to date with IoT security[1], [12] we first need to know what kind of potential threats we need to combat in order to keep our embedded systems safe, while there are many internet-bound attacks like denial-of-service (DOS), distributed-denial-of-service (DDOS), man-in-the-middle(MitM), cross-site scripting(XSS), SQL injection etc. These types of attack work by exploiting software applications (e.g. XSS exploiting a web browser to perform unwanted/malicious actions) or networking protocol (e.g. DDOS using Transmission Control Protocol (TCP) SYN flood attack to overwhelm the target system). Although these attacks should be taken seriously, we are not going to take an in-depth look at them. This paper will be focused on Side-Channel Attacks (SCAs), with a particular focus on a certain type known as Cache-based Side-Channel Attacks (CSCA).

II. HARDWARE DESIGN VULNERABILITIES

Before taking a look at how SCAs work we need to take a look at the hardware design architecture of modern-day processors [16], [18], which is designed maximise performance benefits with the use of special techniques i.e. memory sharing, memory access, speculative execution, branch prediction, speculative evaluation, de-duplication. These techniques, while being able to maximise the performance of the processors are also the attack avenues for SCAs. In the case of Spectre, it is possible to gain access to private data using timing attacks on built-in features of most modern processors known as speculative execution, which results from a branch misprediction. This happens when the CPU uses hardware prefetcher to bring data to the cache before it is ready to be accessed. We can also see this design philosophy translated into the embedded world. With the need for more powerful embedded devices within the IoT, these techniques are leveraged due to embedded devices now having multi core processors (*e.g.* The Raspberry Pi 4 coming standard with an ARM based processor, available from 2 - 4 cores). With the extra processing power comes some of the

same vulnerabilities we see in desktop, server and cloud hardware. [17]

III. AN OVERVIEW OF SIDE-CHANNEL ATTACKS

SCAs are cryptanalysis techniques are used to break cryptographic algorithms by analysing the cryptographic operations to retrieve unauthorized private information. SCAs achieve this by exploiting cryptographic algorithm not based on the weakness of the algorithm itself but the information gathered from the computer system, be it a software or hardware implementation. Researcher have demonstrated SCAs breaking many types of encryption algorithms including Rivest-Shamir-Adleman (RSA) [27], Advanced Encryption Standard (AES) [28] and some Elliptic Curve Cryptography (ECC) schemes like Elliptic-Curve Scalar Multiplication (ECSM) [24]. The reason this works is because SCAs are used to gain access to a secret/private key, rather than trying to break the entire encrypted private data. To obtain a secret key the attacker needs to monitor the target system using one of the many techniques as follows:

1) **Timing SCAs:** An attack based on how much time passes between each computation. Each computation on a system takes certain time to complete and by monitoring the time delay between each computation a cryptographic the attacker can infer what kinds of operations are being executed on a system at any given time e.g. a square operation will take longer to execute than an addition operation, or more correct digits in password will take longer to compare than a password where all digits are incorrect.

2) **Differential Power Analysis:** The attacker is able to monitor the power consumption or a system (e.g. through an oscilloscope). When the system is executing cryptographic operations the power consumption jump in intensity, over the course of multiple cryptographic operations the attacker can infer what cryptographic computation have been executed on the system through statistical analysis of the data collected. This can be useful in conjunction with the timing attack method because the attacker can see how long the power was elevated during each operation which would yield additional information.

3) **Electromagnetic (EM) SCAs:** Every device has some electromagnetic leakage emitting from it in the form of EM radiation. This attack is performed by measuring the levels of electromagnetic radiation emitted from a device. The consequence of every electronic device leaks are certain level of EM radiation means the closer the attacker is to the device while monitoring the more accurate the measurements due to the noise of the surrounding devices. There are methods the attacker can employ to isolate frequencies in a certain range to retrieve whatever frequencies encryption is occurring on.

4) **Software-Initiated fault attacks**: In the case of Rowhammer[25] where the dynamic random-access memory (DRAM) has unintended side effects, causing the memory blocks to leak charges and interact electrically with other blocks. This was achieved by rapidly activating (hammering) adjacent blocks in memory. The side effects caused privileged escalation within software and allowed the unintended retrieval of private information.

5) **Schedule Based Attacks** [2]: As the name suggests this attack effects the schedulers of an OS, the main focus of research has been conducted on real-time systems with fixed-priority scheduling. This attack is achieved by interrupting normal scheduler tasks, by capturing the busy intervals of a scheduler execution and then estimating that busy interval set, the attacker can determine the number of tasks that are involved in each busy interval. From this information the arrival windows of each task can be determined, the arrival window can then be converted to a specific arrival time point, now the exact arrival windows for busy intervals the attacker can generate the start times of jobs, so they have enough information to reconstruct the entire scheduler and perform further attacks.

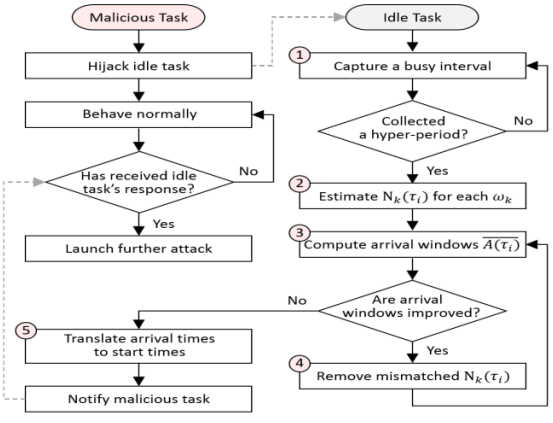

Figure 1: *Attack flow [2] of the proposed scheme between malicious task and idle task.*

Most of these techniques are used in conjunction with each other and are not limited to the sole

method of information gathering and attack strategy. They all have a common goal when it comes to breaking encryption, monitor until you get enough data to deduce the private key and bypass the encryption to access the private data.

These SCA techniques discussed here are not exclusive to just a few types of machines, they affect most types of modern devices. That includes IoT devices since the architecture and design philosophy are largely the same practices reduced into a smaller form-factor with less power consumption.

IV. CACHE-BASED SIDE-CHANNEL ATTACKS

There exists' many varieties of CSCAs, a large number have come to light in recent years. CSCAs effect platform from Server to Virtual Machines (VMs), PCs, laptops and IoT devices. The general idea of how these types of attacks work is, the attacker monitors the cache within a system (usually this caches is shared with other users or system operations etc.) while the CPU is executing instructions it will frequently send request to the cache for data it needs to continue an operation. If the cache has the data, it will send it to the CPU, this is known as a cache hit. If the data is not in the cache, the cache will have to access it from main memory and send it to the CPU, this is known as a cache miss.

Figure 2: *Excerpt of the GDK Cache Template. [15] Dark cells indicate key-address-pairs with high cache-hit ratios.*

With this information the attacker can perform a number of different attacks in order to extract information from the cache. Examples of these attacks are as follows:

1) **Flush+Reload** [14], [19]: The attacker first flushes a cached shared memory location, while the victim accesses or does not access that same location. When the attacker re-accesses the memory location, if the attacker has a fast access time to that memory location, it means the victim has just accessed that part of the cache. If the access time for the attacker is slow it will tell them the victim did not access that memory location. Note the flush in the name refers to the *clflush* (Cache Line Flush) command available in the x86 instruction set. Thus F+R would not work as an attack method on an ARM based processor since the instruction set does not present a way to flush the cache.

2) **Evict+Reload** [14]: There is no flush instruction need for this attack, so other Instruction Set Architectures (ISA) can be vulnerable to this attack. The Attacker needs to evict data from the Last-Level Cache (LLC). With this attack the attacker can use huge shared memory page files. The set to occupy is selected by the Physical address. The attacker can now evict (fill) the set, the victim accesses or does not access a memory location in the set. When the attacker reloads, as with the F+R depending on the access time fast vs slow, the attacker will know if the victim has access a memory address but this time with the set. Note with the E+R method the attacker needs to replace the memory by loading something else instead of flushing it.

3) **Prime+Probe** [14], [19], [20]: Similar to E+R but instead of evicting data the attacker primes (fills) a large block of memory, big enough so that it will cover the entire cache with arbitrary data. When the victim executes a process that access some places in the memory it need to be loaded into cache. Now the attacker can probe (read) the same block of memory that was primed, depending on the access time fast vs slow, the attacker will know if the victim has access a memory address within the primed block.

4) **Last-Level Cache (LLC) Attack** [20]: This type of cache attack is more focused within a Virtual Machine environment, where the users are hosted on cloud based virtual machines within a server setting. Every user has their own virtual environment where they have access to their own private cache access. The problem arises because outside of that virtual environment a level of abreaction down from the user VMs is a shared cache connecting the virtual cache of each user to the CPU. This is where the attacker can use the methods of attack mentioned to gain access to the private data of the users.

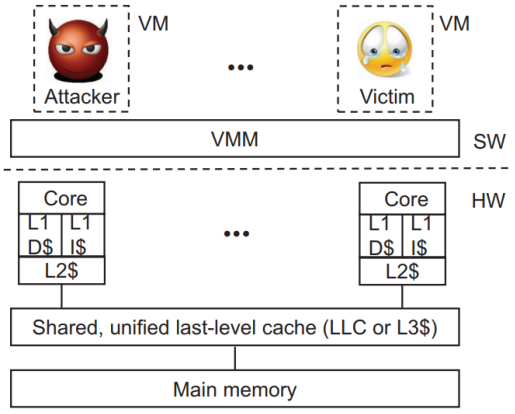

Figure 3: *LLCA [20] System model for multi-core processor.*

These are some examples of SCAs' that exploit the cache to gain access to unauthorised data. There are many more like: Flush+Flush (F+F) [23], Evict+Time (E+T) [15], Invalidate+Transfer (I+T) etc. While these attacks have deferent methods of execution they have a similar goal, which is to find the secret key used in the users cryptographic process [22]. The attacker can run these attacks multiple times to find out what areas of the cache are being accessed while the user applications are executing certain function. From the information of cache hits and misses, the attacker can work backwards to get the secret key, once the secret key is obtained the attacker can use this to decrypt the victims encrypted data. This has been replicated on RSA [27], AES [28], ECC [25] schemes and other cryptographic algorithms where research teams gained access to the secret private key within a few minutes.

V. MACHINE LEARNING FOR DETECTION AND MITTIGATION

How do we defend against these attacks? The answer may lay with Machine Learning [13] [14], which has been highly successful in other areas. Machine learning for security within the technology industry is a relativity new concept, researchers have made great progress in using machine learning that can identify and mitigate not only CSCA but many other types of SCA. The reason machine learning has been successful in the research phase is because most of these SCAs create a unique footprint while they are gathering information about the victim systems.

For instance using machine learning to in conjunction with Hardware Performance Counters (HPCs) [4] which are special-purpose registers built into most modern processors and microprocessors to store counts of hardware-related activities within a computer system. HPCs are useful here because they can disclose behavioural information by counting software micro-events at run-time, specific to detecting CSCAs is the information on cache hits, cache misses, cache references, CPU cycles, retired instructions, branch misspredictions etc. Machine learning can be used with HPC and has proven successful for detecting F+R and F+F attacks have been proven with a detection accuracy of up 99.51% and 99.97% respectively [4]. Machine learning has also shown great results in detection and mitigation of the well-known Spectre vulnerabilities with researcher achieving an accuracy of over 99% [5], both of these examples have provided the results within a test environment, the conclusion for ML being successful in a normal environment are positive based on the results.
Contextual results have also been tested at run-time by recerchers et al M. Mushtaq [3].

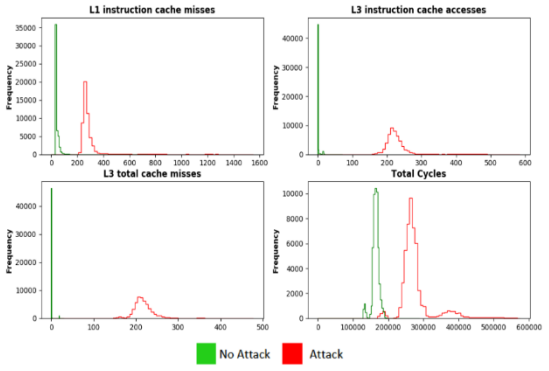

Figure 4: *Selected hardware events under Zero Load Conditions for RSA encryption algorithm: [3] with & without F+R Attack.*

A good example of how this can be implemented on a wide scale of devices is from IBM who have developed their own ML model called WATSON, which is an Artificial Intelligence (AI) system [11] industry customers purchase to monitor customer inquiries, automated machines etc. In the case of monitoring machine, WASTON can detect if there is a

performance reduction over time from any individual machine linked to the system. This is achieved by feeding the model a data set of normal running operations, if there is a sudden deviation from the normal parameters the system will flag the machine having the issues.

Are there any downside to using machine learning as a detection system? It may seem like a complex detection system using ML would come with unsustainable performance overhead, especially in terms of IoT were real-time performance matters, but research from M. Mushtaq et al [3] shows the performance overhead is quite insignificant at a <2% maximum, within 1% completion of a single RSA encryption round. Although, this research was conducted on more powerful Intel processors used mainly for desktops and laptops, how does performance look for IoT device running lower power hardware? This leads to researchers T. Qin et al [10] who developed an Intelligent Maintenance and Lightweight Anomaly Detection System (IMLADS) that has proven very efficient for security management of the IoT. These research teams have shown an accurate ML anomaly detection system can be developed with limited impact on performance.

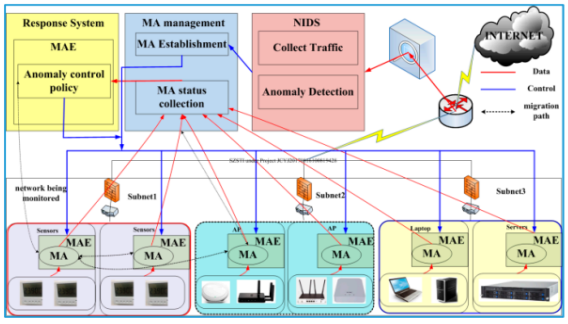

Figure 5: *Introdunction to Intelligent Maintenance and Lightweight Anomaly Detection System (IMLADS) [10]: framwork of IMLADS.*

VI. METHODS TO CORRECT THE ISSUES OF SIDE-CHANNEL ATTACKS

The current solutions to dealing with SCAs on multi-core processors is to disable multi-threading or hyper-threading for most common desktop and laptop based processors. The problem with this is method is the reduction in overall performance for that chip, [29].

Researchers and companies are working on different design implementations that can mitigate most of the known threats from SCAs[5], [6], [7], [8], [9], [10] The issues with this approach is different types of SCAs can surface which would end up being a costly effort to redesign the chip architecture and manufacturing process.

On the software front as more is learned about these kinds of attacks, developers are able to design cache leakage free software. This kind of software design can be designed in a number of ways from, random power cycling of the CPU and cache during cryptographic computation to throw off differential power analysis, secret independent instruction and data access instead of having dependencies, data scrambling [26].

VII. SIMILAR TOPIC FROM ANOTHER SOURCE

Another similar area of research that has come up in January 2019 et al D. Gruss [21]. In the paper they present a new hardware-agnostic side-channel attack known as a Page Cache Attack that targets on of the most fundamental software caches in modern computer systems.

VIII. DISCUSSION AND RESEARCH OPPORTUNITIES

Although there has been a lot of great research within this area, I feel like we are only scratching the surface of what is known about these type of exploits. For instance it is quite possible that there are still some undiscovered vulnerabilities to be uncovered. ML as it in most areas it is applied seems to be quite successful at detecting these kinds of attacks, given the right training data set. I would like to see this area continue to develop at a rapid pace to keep on top of these types of vulnerabilities as they are uncovered. Maybe there is the possibility to expand on current system wide research and implement a multi-side-channel detection system driven by ML in the future.

IX. CONCLUSION

The level of sophistication of SCAs demonstrated on a wide variety of devices is an important reminder that we can never overlook the value of security within IoT. I believe with machine learning implemented in every area of security, it can lead to a cost effective method to identify and mitigate similar threats that could arise from SCAs. Rather than solutions of microprocessor overhauls or large performance impacts. The aim of this paper is to bring to light the issues that IoT and general computational area face. Research into using machine learning on IoT device is still in its infancy the verity in the approaches to solving these issues has made great progress, the research seems quite promising and is a great step needed to secure future IoT platforms.

REFERENCES

[1]C.-Y. Chen, M. Hasan and S. Mohan "Securing Real-Time Internet-of-Things", arXiv:1705.08489v2 [cs.NI] 10 Dec 2018

[2]C.-Y. Chen, R. B. Bobba, and S. Mohan "Schedule-based side-channel attack in fixed-priority real-time systems," University of Illinois, http://hdl.handle.net/2142/88344, Tech. Rip., 2015, [Online].

[3]M. Mushtaq, A. Akram, M. K. Bhatti, M. Chaudhry, M. Yousaf, U. Farooq, V. Lapotre and G. Gogniat "Machine Learning for Security: The Case of Side-Channel Attack Detection at Run-time". ICECS-2018, Dec 2018, Bordeaux, France. hal-01876792

[4]M. Mushtaq, A. Akram, M. K. Bhatti, V. Laptore and G. Gogniat "Cache-Based Side-Channel Intrusion Detection using Hardware Performance Counters" CryptArchi 2018 – 16th International Workshops on Cryptographic architectures embedded in logic devcices, Jun 2018, Lorient, France. cel-01824512

[5]J. Depoix and P. Altmeyer "Detecting Spectre Attacks by identifying Cache Side-Channel Attacks using Machine Learning", WAMOS2018, August 2018, Wiesbaden.

[6]A. Zankl, H. Seuschek, G. Irazoqui and B. Gülmezoğlu "Side-Channel Attacks in the Internet of Things: Threats and Challenges". 10.4018/978-1-5225-2845-6.ch013 (2017)

[7]Hemavati, Dr. Aparna R, "A Survey on Intrusion Detection System using Machine Learning and Deep Learning", International Journal of Scientific Research in Computer Science, Engineering and Information Technolgy (IJSRCSEIT), ISSN : 2456-3307, Volume 5 Issue 2, pp. 264-270, March-April 2019.

[8]T. Alves, R. Das and T. Morris "Embedding Encryption and Machine Learning Intrusion Prevention Systems on Programmable Logic Controllers" in IEEE Embedded Sytesm Letters, vol. 10, no. 3, pp. 99-102, Sept. 2018.

[9]K.-K. R. Choo, M. M. Kermani, R. Azarderakhsh and M. Govindarasu "Emerging Embedded and Cyber Physical System Security Challenges and Innovations", IEEE Transactions on Dependable and Secure Computing, Vol. 14, No. 3, pp. 235-236, May/June 2017

[10]T, Qin, B. Wang, R. Chen, Z. Qin and L. Wang "IMLADS: Intelligent Maintenance and Lightweight Anomaly Detection System for Internet of Things", Sensors 2019, 19, 958; doi:10.3390/s19040958

[11]A. B. T. Hopkins, P. Sartian, K. D. McDonald-Maier and W. G. J. Howells "Towards Embedded Artificial Intelligence Based Security for Computer Systems" 2008 Bio-inspired, Learning and Intelligent Systems for Security, Edinburgh, 2008, pp. 81-86.

[12]J. Lizarraga, R. Uribeetxeberria, U. Zurutuza and M. Fernáandez "Security in Embedded Systems", https://www.researchgate.net/publication/249810205_SECURITY_IN_EMBEDDED_SYSTEMS IADIS International Conference Applied Computing 2006.

[13]L. Lerman, G. Bontempi and O. Markowitch "Side channel attack: an approach based on machine learning", https://www.researchgate.net/publication/269463480_Side_channel_attack_An_approach_based_on_machine_learning [online].

[14]G. Irazoqui and X. Guo "Cache Side Channel Attack: Exploitability and Countermeausres", blackhat asia 2017

[15]D. Gruss, R. Spreitzer and S. Mangard "Cache template Attacks: Automating Attacks on Inclusive Last-Level Caches", 24th USENIX Security Symposium, August 12-14, 2015 Washington, D.C.

[18]F. Abdi, C.-Y Chen, M. Hasan, S. Liu, S. Mohan and m. Caccamo "Guaranteed Physical Security with Restart-Based Design for Cyber-Physical Systems" in ACM/IEEE ICCPS, 2018.

[16]A. Fogh "Cache side channel attacks: CPU Design as a security problem", HITBSecConf Amsterdam//Malaysia, May 27, 2016 https://conference.hitb.org/hitbsec-conf2016ams/sessions/cache-side-channel-attacks-cpu-design-as-a-security-problem/ [online].

[17]D. Dinu and I. Kizhvatov (2018) "EM Analysis in the IoT Context: lessons Learned from an Attack on Thread" IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018(1), pp. 73-97.

[19]Z. He and R. B. Lee. 2017 "How secure is your cache against side-channel attacks?" In Proceedings of MICRO-50, Cambridge, MA, USA. October 14-18, 2017.

[20]F. Liu, Y. Yarom, Q. Ge, G. Heiser and R. B. Lee "Last-Level Cache Side-Channel Attacks are Practical" in 2015 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 2015 pp. 605-622.

[21]D. Gruss, E. Kraft, T. Tiwari, M. Schwarz, A. Trachtenberg, J. Hennessey, A. Ionescu and A. Fogh "Page Cache Attacks" arXiv:1901.01161v1 [cs.CR] 4 Jan 2019.

[22]Y. Zhang, A. Jeuls, M. K. Reiter and T. Ristenpart "Cross-Tenant Side-Channel Attacks in PaaS Clouds". ACM Conference on Computer and Communications Security (2014). pp. 990-1003.

[23]D. Gruss, C. Maurice, K. Wagner and S. Mangard "Flush+Flush: A Fast and Stealthy Cache Attack" arXiv:1511.04594v3 [cs.CR] 5 Apr 2016.

[24]X. Lou, F. Zhang, Z. L. Chua, Z. Liang, Y. Cheng and Y. Zhou "Understanding Rowhammer Attacks through the Lens of a Unified Reference Framework" arXiv:1901.03538v1 [cs.CR] 11 Jan 2016.

[25]E. Nascimento, L. Chemielewski, D. Oswald and P. Schwabe "Attacking embedded ECC implementations through comv side channels" SAC 2016 (Lecture Notes in Computer Science).

[26]M. Neagu and L. Miclea "Protecting Cache Memories through Data Scrambling Technique" 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj Napoca, 2014, pp. 297-303.

[27]S. Sarkar and S. Maitra "Side Channel Attack to Actual Cryptanalysis: Breaking CRT-RSA with Low Weight Decryption Exponents" Lecture Notes in Computer Science, pp. 476-493, 2012.

[28]M. Neve and K. Tiri "On the complexity of side-channel attacks on AES-256 – methodology and quantitative results on cache attacks –" IACR Cryptology ePrint Archive, pp. 318, 2007.

[29]G. Chen, W. Wang, T. Chen, S. Chen, Y. Zhang, X. Wang, T.-H. Laui and D. Lin "Racing in Hyperspace: Closing Hyper-Threading Side Channels on SGX with Contrived Data Races" 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, 2018, pp. 178-194.