ADVERSARIAL LEARNING INSPIRED EMERGING SIDE-CHANNEL ATTACKS AND DEFENSES

Abhijitt Dhavlle
Electrical and Computer Engineering
George Mason University
Fairfax, USA.
adhavlle@gmu.edu

April 12, 2021

ABSTRACT

Evolving attacks on the vulnerabilities of the computing systems demand novel defense strategies to keep pace with newer attacks. This report discusses previous works on side-channel attacks (SCAs) and defenses for cache-targeted and physical proximity attacks. We then discuss the proposed Entropy-Shield as a defense against timing SCAs, and explain how we can extend the same to hardware-based implementations of crypto applications as “Entropy-Shield for FPGA”. We then discuss why we want to build newer attacks with the hope of coming up with better defense strategies.

Keywords Side-Channel Attacks (SCAs), Defenses, Adversarial Learning.

1 Introduction

The hardware security domain in recent years has experienced a plethora of threats Side-Channel Attacks [1,2], Malware attacks [3–9], Hardware Trojan attacks [10], reverse engineering threats [11–13] and so on. Among multiple threats, the side-channel attacks (SCAs) is one of the pivotal threats due to it’s capability to exploit the design despite being introduced in the market post-validation. These SCAs function by exploiting the side-channels, which invariably leak important data during an application’s execution. The information leaked through side-channels are inherent characteristics of the system and are often unintentional. The side-channel information can be device’s microarchitectural or physical information such as power consumption, thermal maps, the timing of the operations, acoustics, and cache access traces. Some of the attacks are reported in [14–16] with vulnerabilities in Linux been reported in [17]. The work in [18] lists and reports the common vulnerabilities in the computing systems discovered so far, while some newer threats to a CPU have been reported and described in [19]. It is quite evident by looking at the mentioned reports that the side-channel attacks have proven a major threat to system security, and developing novel mechanisms to mitigate them becomes inevitable.

Snooping on co-located programs’ cache trace is performed by cache-based SCAs and is prominently explored in the domain of SCAs. A part of my thesis work focuses on timing-based cache attacks, which exploit the time required to flush/reload an instruction from the cache subsystem as a covert channel [20,21]. Some of the popular cache targeted SCAs are the Flush+Reload [1] and Flush+Flush attack [2]. Intercepting secret information based on the study of power signature is a subdivision of SCAs where power consumption information serves as a covert channel leaking crucial information about the executed operations. Such physical and cache-based SCAs are known to be a significant threat to cryptosystems such as AES (Advanced Encryption Standard) and RSA and can reveal the encryption key efficiently. Yet these attacks are not limited to the cryptographic algorithms, and they can be extended to snoop on other applications as well. One can argue to have a solution that can simply shut down covert channels and hence prevent such threats in the first place. It is not as straightforward as it seems to be because shutting down covert channels that originated due to vulnerability is not feasible due to the cost and performance constraints. Hence, in this report, we introduce the reader to primarily cache-targeted and physical SCAs, discuss the proposed solution, existing works, and how it compares with our solution, along with current ongoing and future projects.
The rest of this report is organized as follows: Section 2 enumerates and describes the existing works on cache-targeted attacks and defenses, and our proposed defense; Section 3 mentions previous works on physical SCAs and the ongoing research in the domain; Section 5 concludes this report with the anticipated results and contributions considering the current ongoing research.

2 Cache-Targeted Side-Channel Attacks/Defenses

Cache-targeted attacks primarily rely on the timing information to successfully probe the intended process/data. The idea is to co-locate with a victim and snoop on its private data over the covert channel. Intel CPUs use `cflush` instruction to evict a specified cache line from the cache subsystem. Intel uses inclusive caches, meaning a line evicted from the cache using the `cflush` command does evict it from all the levels of cache. The block diagram of the cache subsystem is shown in Figure 1(b). The attackers time the `cflush` instruction after evicting a cache line belonging to the victim. Depending on the access time, the attacker deduces if the victim accessed the data. Flush+Flush [2] and Flush+Reload [1] are some of the most popular cache-targeted SCAs exploiting the timing information to steal private data.

In this section, we discuss the existing cache-targeted SCAs, defenses, and our proposed Entropy-Shield as a defense against cache-targeted attacks.

2.1 Existing Attacks Targeting Cache

Flush+Flush: Flush+Flush [2] is a passive type of an attack unlike its sibling, Flush+Reload [1], in the sense that it does not reload the data again to check a hit/miss. This attack depends on the time it takes for the `cflush` instruction to complete. Because the system has to flush all the copies of the flushed data, it requires some time to do so, and hence, flushing a data that is present in the cache takes more time as against missing data. Hence, Flush+Flush can silently attack a victim and steal its secret data.

Prime+Probe: The Prime+Probe [22] attack targets the LLC. The attacker ‘primes’ or prepares its cache by loading a chunk of data. It then waits for the victim to execute. If the victim’s data is mapped to the same area as the attacker’s data, the attacker data is evicted given the replacement policy. When the attacker reaccesses its data, it deduces if the victim’s data was present, depending on whether the ‘probe’ was a hit or a miss.

Flush+Reload: A quick background on the Flush+Reload attack is discussed here. Figure 1(a) demonstrates the attack process:

1. The attacker flushes the victim’s cache line, which corresponds to a line/function in victim’s code.
2. The attacker waits for the victim to access its data.
3. The attacker again reloads the same data and times this access.
4. If the time happens to be greater than the threshold, it means the victim did not access the data. This event corresponds to a cache miss.
5. If the access time is less than the threshold, the victim accessed the data. This corresponds to a cache hit.

2.2 Existing Defenses for Cache-Targeted SCAs

The existing defenses against cache-targeted SCAs can be broadly categorized into five categories, as discussed below.

2.2.1 Randomization based

Cases where modifying the hardware is not feasible, randomizing the memory accesses has been proposed in [23]. The work in [24] proposed the use of random memory-to-cache mappings to make the attack difficult. A permutation table for each process causes dynamic memory address to cache set maps. The attacker relies on evicting a specific cache line to perform the attack, dynamic memory address mapping prevents the attacker from evicting the targeted cache line. This comes for a price though, as maintaining the table during runtime affects performance. A dynamic random control flow by exploiting software diversity is introduced in [25]. The main idea is to transform the program trace of each application into a unique trace. The technique dynamically randomizes the control flow of the program during runtime. Replicas of functions within the code are created, and the dynamic program flow chooses one of them in runtime to randomize the overall control flow.
2.2.2 Partition based

DAWG by Kiriansky [26] proposes a Dynamically Allocated Way Guard (DAWG) to protect way partitions in a set-associative cache. Secure isolation is provided by giving a notion of protected domains within the cache. DAWG is capable of isolating cache miss, hit, and other metadata across the protected domains. DAWG can protect against attacks that depend on timing behaviors. The defense mechanism can be extended to branch history tables or the TLBs. However, DAWG would need supplemental techniques to block covert channels, and this defense leads to performance overhead. Another partitioning based technique is proposed in [24], which promises to protect sensitive data from attack by reserving dedicated cache sets. The sensitive process, in this way, would map to the protected cache area, thereby reducing the interference caused by other programs. But, this method requires the intervention of the operating system to separate cache into a reserved and non-reserved area, all of which must be non-overlapping. This brings overhead and inefficient cache utilization, along with security.

2.2.3 Intel Cache Allocation Technology (CAT)

Work in [27] utilized Intel’s CAT technology to protect SCAs on shared last-level cache. CAT is a way partitioning mechanism that ensures efficient utilization of the cache space/occupancy. Another work in [28] proposes to use CAT in addition to the mechanism that can partition the LLC to thwart SCAs. CAT cannot by itself securely partition the cache. The attack launches on a compromised operating system. Hence, by preventing cache line sharing with applications or OS, the authors propose to defend against SCAs.

2.2.4 Detection based defenses

Cloudradar [29] proposes to detect SCAs using hardware performance counters (HPCs). Specialized cores are employed to detect SCAs based on either signature detection technique or anomaly-based detection established on HPCs. The system constantly monitors the HPCs. These traces are then compared with pre-stored attack signatures to detect the presence of an attack. Another technique used is the anomaly detection technique, which alerts the system based on the drift of the program flow from its normal flow. Stealthmem [30] provides a system-level protection against cache SCAs. It protects the cache from unauthorized accesses by providing a set of locked cache lines per core. These cache lines are never evicted. Authors in [31] develop a technique, titled as dynamic cache coloring. This technique cautions the virtual machine when a security-sensitive operation is being processed. This swaps the data to a safe, isolated cache line that has limited access, thus preventing the attack.
2.2.5 SGX Enclave based

Vayrs [32] proposes to protect the system from timing based and page-table attacks. The main idea is to prevent shared resources. During the execution of secure-sensitive operations in the enclave, strict reservation of physical cores is enforced. These previous works have been summarized in Table 1.

| Title/Reference | Technique |
|-----------------|-----------|
| New Cache [23]  | Randomize memory maps. |
| "How secure is your cache?" [24] | Use permutation tables to randomize cache mapping to make it harder to evict specific targeted cache line. |
| Vayrs [32]      | Protects programs in SGX enclaves from cache attacks. Mitigates reservation of physical cores to threads. |
| Cloudradar [29] | Cores (processors) equipped with specialized signature detection. Detects SCAs based on the hardware performance counters (HPCs). |
| Stealthmem [30] | Protects cache from unauthorized access by creating set of locked cache lines per core. Lines are never evicted from the cache. |
| Dynamic Cache Coloring [31] | Dellers VM when application executing secure-sensitive operations. Swaps associated data to safe and isolated cache line, limiting its access. |
| Partition Locked (PL) Cache [32] | Sensitive process assigned reserved ways in each cache set. Disallow attacker from occupying whole cache set. |

2.3 Proposed Entropy-Shield

In this subsection, we discuss the vulnerability in the GnuPG RSA encryption algorithm and propose a defense mechanism to thwart attacks.

2.3.1 Vulnerability in RSA

The vulnerabilities in the existing RSA algorithms is described in [1] and also demonstrates how Flush+Reload can successfully retrieve all the secret key bits. Referring to Table 2, bit ‘0’ in a secret key would translate to RSA executing a square operation followed by a modulo operation. Likewise, square-modulo-multiply-modulo operations are called for a bit ‘1’. Table 3 shows the sequence of operations corresponding to different key bit combinations. By observing the functions calls to the square, reduce or modulo lines of code, the attacker can deduce the secret key bits. Not all the bits need to be retrieved at once, but the attacker has to execute the attack for several hundreds of thousands iteratively to guess the correct key.

2.3.2 Entropy-Shield

Earlier, we discussed different attacks and defense strategies. With the defenses discussed, it is evident that some of them require software level changes to the operating system or the hypervisor, while others require hardware modifications, causing inefficient utilization of the cache or severe overheads. Hence, we propose Entropy-Shield as a defense against timing-based SCAs, shown in Figure 2. In Figure 2(a), we can see that the attacker can sniff the covert channel and hence, retrieve secure-sensitive data. With Entropy-Shield the attacker is misled to an incorrect key. This is achieved using fake/dummy calls to the square, reduce, or multiply functions. We call this as adding perturbations to the signal/channel observed by the attacker. For example, if a bit ‘0’ is processed by the victim, the Entropy-Shield calls dummy Multiply-Modulo operations following the original Square-Modulo operations, thus giving the notion of bit ‘1’ processed by the victim when bit ‘0’ was processed. Refer to Table 4 for the fake operations corresponding to key bits. Referring to Figure 2(b), with Entropy-Shield dummy operations are called to mislead the attacker. The Entropy-Shield offers Uniform and Deceptive mode of operation. Uniform mode adds maximum noise to the channel by flipping all 0’s to 1’s, while Deceptive mode cognitively perturbs the sequence by randomly calling dummy operations, the position of the bits perturbed change every run. The results of the Entropy-Shield from [20] is presented in in Tables 5 and 6.

| Secret Key Bit | Corresponding Operations |
|----------------|--------------------------|
| Bit ‘0’        | Square-Modulo            |
| Bit ‘1’        | Square-Modulo-Multiply-Modulo |

| Bit Sequence | Corresponding Operations |
|--------------|--------------------------|
| “0010”       | SR-SR-SRMR-SR            |
| “1100”       | SRMR-SRMR-SR-SR          |
Table 4: Sequence of Key Bits to Corresponding Operations

| Secret Key Bit | Corresponding Fake Operations |
|----------------|------------------------------|
| Bit '0' | Square-Modulo |
| Bit '1' | Square-Modulo-Multiply-Modulo |

Table 5: Key as visible to the attacker and the victim with Entropy-Shield - Uniform mode of operation

| Attack Type | Encryption | Key | Original Key | Victim seen key | Key seen by the attacker |
|-------------|------------|-----|--------------|------------------|-------------------------|
| Flush+Reload | RSA-RSA | key_1 | 0FCFFF | 0FCFFF | FFFFFF |
| DSA-Elgamal | key_2 | 587BFA | 587BFA | FFFFFF |
| Flush+Flush | RSA-RSA | key_3 | 54FF0B | 54FF0B | FFFFFF |
| DSA-Elgamal | key_4 | 89DE00 | 89DE00 | FFFFFF |

Table 6: Key as visible to the attacker and the victim with Entropy-Shield - Deceptive mode of operation

| Attack Type | Encryption | Key | Original Key | Victim seen key | Key seen by the attacker |
|-------------|------------|-----|--------------|------------------|-------------------------|
| Flush+Reload | RSA-RSA | key_1 | 0FCFFF | 0FCFFF | 5FDFFF |
| DSA-Elgamal | key_2 | 587BFA | 587BFA | 59FBFB |
| Flush+Flush | RSA-RSA | key_3 | 54FF0B | 54FF0B | 75FF1B |
| DSA-Elgamal | key_4 | 89DE00 | 89DE00 | CBDFO2 |

3 Physical Side-Channel Attacks

Physical side-channel attacks are those that require proximity to a victim device - an FPGA or a CPU. The attack principle is to target data-dependent power traces/signatures of the victim device. Crypto algorithm like the AES demonstrates data-dependent power consumption which serves as a covert channel. The attacker attaches a probe to measure the power traces by executing the application iteratively. By applying methods such as Correlation Power Analysis (CPA), Differential Power Analysis (DPA), or the Simple Power Analysis (SPA), the attacker can reveal the entire secret key. This data-dependent behavior can be extended to other non-cryptographic applications as well to snoop on the private data. In this section, we discuss the physical side-channel attacks, those that require proximity to the target hardware, as well as the remote physical side-channel attacks. We review the existing attacks, defenses, and our ongoing work in this domain.
3.1 Existing Power-based Physical Side-Channel Attacks

Work in [34] exploits a new vulnerability in the power-sharing within FPGA to snoop on other entity’s power consumption. The power rails are shared across all the components. The attack employs ring oscillator (RO) -based design to convert voltage drops to figures, representing power consumption of co-located applications on the same FPGA. The attack describes the successful recovery of the RSA key. Authors in [35] bring to light yet another type of vulnerability. The work discusses how an RO-based spy can read the relative delay in the wire, a delay caused by logic values on the adjacent victim wire. The crosstalk between two wires - transmitter (victim) and the receiver (spy) - is exploited in this type of attack. Hardware designs have a strict power budget, which causes one component to power throttle while another power-hungry component is operating. This power budget has been exploited as a covert channel in [36]. The source entity is assumed to have access to sensitive data while the sink has access to a third party application that reads the data. Using power viruses, the attack causes other known (spy) entity to power throttle, thus utilizing this reduced performance factor to correspond to logical values ‘0’ or ‘1’. Simple Power Analysis (SPA) [37, 38], Difference-of-Means Power Analysis (DPA) [39] and Correlation Power Analysis (CPA) [39] are analysis methodologies used to extract secret data from the observed power traces of an application. Simple Power Analysis is the most simple type of them. It requires manual observation of the power traces to extract the key. DPA requires the collection of a large number of traces. These traces are processed statistically to find the mean of differences observed in the traces, to derive the secret information. While in CPA, the correlation between the measured power traces and the guesses (from a hypothetical power model) is calculated to conclude the secret key.

3.2 Existing Defenses

Nele Mentens in [40] proposed to utilize hardware components of the FPGA to thwart attacks. The reconfigurations to prevent leakage are performed in runtime. The author discusses the addition of random noise, irregular clock cycles, and scrambling S-Box in the hardware to thwart attacks. Work in [41] proposed an intelligent place-and-route technique to perform symmetrical routing. This approach is explained to defense against power analysis SCAs. Authors in [42] explain the algorithmic countermeasures that minimize the correlation between power and processed data.

3.3 Ongoing work in Physical Side-Channel Attacks domain

The ongoing work in physical SCAs is to research on methods to secure FPGA implementation of crypto algorithms from attacks. We are working on CPA type attacks on power traces collected during the execution of the AES application. Once the attack is successful, securing the FPGA with approximate modules is what we plan to focus on in the future. Post power trace collection on the secure design, we would attack with CPA again to evaluate the resilience of the new security measure implemented.

4 Research Progress:

As described in the previous sections, I have been working and publishing papers on Malware Detection [6], Side-Channels Analysis [20, 21], Hardware-based Trojan Attack and Detection [10], and Survey-based papers [43, 44] published to conferences and journals. I intend to dive deeper into more of SCA based attacks and defenses in future and contribute my work to top tier conferences and journals.

5 Conclusion

Utilizing traditional randomization is not the best solution -given the overhead, performance, and design requiring significant hardware modifications. As a panacea, we discussed our proposed Entropy-Shield that would offer security with significantly less overhead. Our Entropy-Shield incurs less overhead (2×). With the positive results achieved so far, we would like to extend this work further in the hardware domain, thus enabling cognitive randomization based defense for FPGAs, which would secure them from side-channel attacks. Furthermore, working on approximate modules for AES, adversarial learning, and one-shot learning with neural networks, we want to contribute to the community by building attacks that would help us evaluate our defenses more accurately.

References

[1] Y. Yarom and K. Falkner, “Flush+reload: A high resolution, low noise, l3 cache side-channel attack,” in USENIX Conference on Security, 2014.
[2] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, “Flush+flush: A fast and stealthy cache attack,” in \textit{Int. Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment}, 2016.

[3] A. Dhavlle, S. Shukla, S. Rafatirad, H. Homayoun, and S. M. P. Dinakarrao, “Hmd-hardener: Adversarially robust and efficient hardware-assisted runtime malware detection,” in \textit{Design, Automation and Test in Europe (DATE)}, 2021.

[4] A. Dhavlle, R. Hassan, M. Mittapalli, and S. M. P. D, “Design of hardware trojans and its impact on cps systems: A comprehensive survey,” in \textit{International Symposium on Circuits and Systems (ISCAS)}, 2021.

[5] M. M. Ahmed, A. Dhavlle, N. Mansoor, S. M. P. D, K. Basu, and A. Ganguly, “What can a remote access hardware trojan do to a network-on-chip?” in \textit{International Symposium on Circuits and Systems (ISCAS)}, 2021.

[6] S. M. P. Dinakarrao, S. Amberkar, S. Bhat, A. Dhavlle, H. Sayadi, A. Sasan, H. Homayoun, and S. Rafatirad, “Adversarial attack on microarchitectural events based malware detectors,” in \textit{Design Automation Conference (DAC)}, 2019.

[7] S. Shukla, G. Kolhe, S. M. P. D, and S. Rafatirad, “Microarchitectural events and image processing-based hybrid approach for robust malware detection: Work-in-progress,” in \textit{Proceedings of the International Conference on Compliers, Architectures and Synthesis for Embedded Systems Companion}, 2019.

[8] S. Shukla, G. Kolhe, S. M. PD, and S. Rafatirad, “Rnn-based classifier to detect stealthy malware using localized features and complex symbolic sequence,” in \textit{2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)}, 2019, pp. 406–409.

[9] S. Shukla, G. Kolhe, S. M. P D, and S. Rafatirad, “Stealthy malware detection using rnn-based automated localized feature extraction and classifier,” in \textit{2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)}, 2019, pp. 590–597.

[10] M. Meraj Ahmed, A. Dhavlle, N. Mansoor, P. Sutradhar, S. M. Pudukotai Dinakarrao, K. Basu, and A. Ganguly, “Defense against on-chip trojans enabling traffic analysis attacks,” in \textit{2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)}, 2020, pp. 1–6.

[11] G. Kolhe, S. M. PD, S. Rafatirad, H. Mahmoodi, A. Sasan, and H. Homayoun, “On custom lut-based obfuscation,” in \textit{Proceedings of the 2019 on Great Lakes Symposium on VLSI}, 2019.

[12] G. Kolhe, H. M. Kamali, M. Naicker, T. D. Sheaves, H. Mahmoodi, P. D. Sai Manoj, H. Homayoun, S. Rafatirad, and A. Sasan, “Security and complexity analysis of lut-based obfuscation: From blueprint to reality,” in \textit{2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)}, 2019.

[13] R. Hassan, G. Kolhe, S. Rafatirad, H. Homayoun, and S. M. Dinakarrao, “Satconda: Sat to sat-hard clause translator,” in \textit{International Symposium on Quality Electronic Design (ISQED)}, 2020.

[14] \url{https://www.darkreading.com/vulnerabilities—threats/researchers-develop-new-side-channel-attacks-on-intel-cpus/dd/dd-id/1337287}.

[15] \url{https://threatpost.com/intel-zombieload-side-channel-attack-10-takeaways/144771/}.

[16] \url{https://threatpost.com/new-swapgs-side-channel-attack-bypasses-spectre-and-meltdown-defenses/147034/}.

[17] \url{https://access.redhat.com/articles/CPUSideChannelAttackIndex}.

[18] \url{https://www.cvedetails.com/}.

[19] \url{https://cpu.fail/}.

[20] A. Dhavlle, R. Mehta, S. Rafatirad, H. Homayoun, and S. M. P. Dinakarrao, “Entropy-shield: Side-channel entropy maximization for timing-based side-channel attacks,” in \textit{International Symposium on Quality Electronic Design (ISQED)}, 2020.

[21] A. Dhavlle, S. Rafatirad, K. Khasawneh, H. Homayoun, and S. M. P. Dinakarrao, “Imitating functional operations for mitigating side-channel leakage,” \textit{IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems}, pp. 1–1, 2021.

[22] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, “Last-level cache side-channel attacks are practical,” in \textit{2015 IEEE Symposium on Security and Privacy}, 2015.

[23] Z. Wang and R. B. Lee, “New cache designs for thwarting software cache-based side channel attacks,” in \textit{Proceedings of the ISCA}, 2007.

[24] Z. He and R. B. Lee, “How secure is your cache against side-channel attacks?” in \textit{Proceedings of the IEEE/ACM}, 2017.
[25] S. Crane, A. Homescu, and et al., “Thwarting cache side-channel attacks through dynamic software diversity,” in *In Network and Distributed System Security Symposium*, 2015.
[26] V. Kiriansky, I. A. Lebedev, and et al., “DAWG: A defense against cache timing attacks in speculative execution processors,” in *MICRO*, 2018.
[27] F. Liu, Q. Ge, Y. Yarom, F. McKeen, C. Rozas, G. Heiser, and R. B. Lee, “Catalyst: Defeating last-level cache side channel attacks in cloud computing,” in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, March 2016.
[28] X. Dong, Z. Shen, and et al., “Shielding software from privileged side-channel attacks,” in *USENIX Security Symposium*, 2018.
[29] T. Zhang, Y. Zhang, and R. B. Lee, “Cloudradar: A real-time side-channel attack detection system in clouds,” in *RAID*, 2016.
[30] T. Kim, M. Peinado, and G. Mainar-Ruiz, “STEALTHMEM: System-level protection against cache-based side channel attacks in the cloud,” in *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, 2012.
[31] J. Shi, X. Song, H. Chen, and B. Zang, “Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring,” in *Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops*, 2011.
[32] O. Oleksenko, B. Trach, and et al., “Varys: Protecting SGX enclaves from practical side-channel attacks,” in *USENIX*, 2018.
[33] L. Domnitser, A. Jaleel, J. Loew, N. Abu-Ghazaleh, and D. Ponomarev, “Non-monopolizable caches: Low-complexity mitigation of cache side channel attacks,” *ACM Trans. Archit. Code Optim.*, vol. 8, no. 4, Jan. 2012.
[34] M. Zhao and G. E. Suh, “Fpga-based remote power side-channel attacks,” in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018.
[35] C. Ramesh, S. B. Patil, S. N. Dhanuskodi, G. Provelengios, S. Pillement, D. Holcomb, and R. Tessier, “FPGA Side Channel Attacks without Physical Access,” in *International Symposium on Field-Programmable Custom Computing Machines*, 2018.
[36] S. Khatamifard, L. Wang, A. Das, S. Kose, and U. Karpuzcu, “Powert channels: A novel class of covert communication exploiting power management vulnerabilities,” in *Proceedings - 25th IEEE International Symposium on High Performance Computer Architecture, HPCA 2019*, 2019.
[37] S. Mangard, M. Oswald, and T. Popp, *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007.
[38] S. Mangard, “A simple power-analysis (spa) attack on implementations of the aes key expansion,” in *Proceedings of the 5th International Conference on Information Security and Cryptology*, ser. ICISC’02, 2002.
[39] O. Lo, W. J. Buchanan, and D. Carson, “Power analysis attacks on the aes-128 s-box using differential power analysis (dpa) and correlation power analysis (cpa),” *Journal of Cyber Security Technology*, 2017.
[40] N. Mentens, “Hiding side-channel leakage through hardware randomization: A comprehensive overview,” in *2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS 2017*, Pythagorion, Greece, 2017.
[41] P. Yu and P. Schaumont, “Secure fpga circuits using controlled placement and routing,” in *IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, 2007.
[42] P. C. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO ’99, 1999.
[43] F. Brasset, L. Davi, A. Dhaville, T. Frassetto, S. M. P. Dinakarrao, S. Rafatirad, A. Sadeghi, A. Sasan, H. Sayadi, S. Zeitouni, and H. Homayoun, “Advances and throwbacks in hardware-assisted security: special session,” in *CASES*, 2018.
[44] A. Dhaville and S. M. Pudukotai Dinakarrao, “A comprehensive review of ml-based time-series and signal processing techniques and their hardware implementations,” in *International Green and Sustainable Computing Workshops (IGSC)*, 2020, pp. 1–8.