A Survey on SGX Enclave Cache-miss Based Side-Channel Attacks

AMIN FALLAHI, Syracuse University, USA

Side-channel attacks targeting programs which are executed on Intel SGX processors are of common vulnerabilities in this platform. The security of code and data in the cloud is provided by execution of the program in the SGX enclave, but it is subject to side-channel attacks which target the enclave and retrieve data by controlling page-faults, cache-misses, TLB flushes, etc. In this report, we review and discuss common attacks on SGX enclaves, the protection which is needed and currently trending in research. In our study, we have observed and examined several researches done in this area and compared them with each other. Another importance in this area, is the performance loss caused by protection techniques which we also discuss in this report.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Security and privacy** → **Hardware-based security protocols**; **Side-channel analysis and countermeasures**; *Distributed systems security*; • **Computer systems organization** → *Processors and memory architectures*;

Additional Key Words and Phrases: Side-Channel Attack, Software Guard Extension, Page-Monitoring Attack

1 INTRODUCTION

One of the present challenges in cybersecurity systems is to outsource programs to be executed in the cloud and at the same time maintaining the privacy of the owner and the security of data. The program code has to be protected alongside the data which is being used inside it. Even in a securely encrypted code, the low-level instructions are visible when they are being executed by the CPU and a malicious operating system kernel can tamper with them. Several attacks are possible by reverse engineering the instructions, including code reuse, data injection, copyright violation, and recovery of classified data. Intel SGX (Software Guard eXtensions) [3] is a novel architecture introduced with Intel Skylake micro-architecture [6] that protects the code and the data inside an encrypted enclave that is a portion of memory which is protected by the support of hardware. The enclave is secure against direct eavesdropping by the adversary, but there are several successful attacking methods using side-channel information which is visible to the adversary [11]. The problem with this hardware extension is that the memory layout is still visible to the operating system kernel and a malicious OS in the cloud can recover useful information by observing the occurrences of page faults. As a result, this side-channel attack can be successful on recovering the data by clearing the page table and causing page faults and observing them for different inputs. In the same scenario, cache-misses are a point for side-channel attacks. The pattern of cache-miss occurrences can be used by a malicious operating system to attack the enclave program.

Several theoretical and practical attacks based on SGX side channels discussed and experimented to be practical and effective. The points of interest during the execution of a program inside enclave for the adversary are cache and whatever is missed and retrieved from the memory, page-faults, and page-table which is attacked by making use of accessed and modified bits.

Author's address: Amin Fallahi, Syracuse University, 111 College Pl, Syracuse, NY, 13210, USA, afallahi@syr.edu.

In response to the development of attacks, several defense mechanisms are designed which have been effective in some cases and conditions. These defense methods usually sacrifice performance or accuracy while improving the security against side-channel attacks.

In this report, we study the possible attacks which are possible using SGX side-channel information and their points of success. Then, we discuss possible defenses which have been proposed by the researchers and study their methods and compare the defenses and their effectiveness under different conditions and scenarios.

In summary, the main points of study in this work are:

- Examining SGX enclave points of attacks.
- Discussing the past work which have been done on defending the SGX enclave against side-channel attacks.
- Comparing and analyzing the defense methods and their effectiveness.

Outline In the remainder of this report, we discuss basic background knowledge required for studying this work in section 2. Then, we review and analyze the attacks which are possible to be launched against SGX enclave using side-channel information in section 3. In section 4, we introduce several defenses which have been proposed by past researches and in section 5 we analyze and compare the effectiveness of these defense methods in terms of security and performance. At last, we conclude the research in section 6.

2 BACKGROUND

In this section, we provide detailed background on side-channel attacks targeting enclaved applications, tools, and terms we use in this report.

2.1 Intel SGX

Protecting the data inside a shield has been a major interest for improving security and privacy in distributed systems and several software-based and hardware-based solutions have been proposed for this purpose.

Intel introduced a new hardware-based technique in its new Skylake micro-architecture [6] line of processors called Software Guard eXtensions (SGX) [3]. The main function of it is assigning a portion of main memory to a program and protect it from being read and written by other processes. The effectiveness of this technique scales up to preventing the operating system kernel itself from accessing the shielded memory which is called enclave. Intel SGX API [4] uses two main kinds of specialized function calls to work with enclave. ECALLs are functions which run inside the enclave and can be called from outside and OCALLs are useful for using system functions and other operations which are possible to be done outside enclave [2].

One of the major problems with running applications inside the cloud is the leakage of data and code instructions. Even with encryption, the CPU can only execute unencrypted instructions which can be eavesdropped and reverse engineered to useful code. Only with hardware modifications and adding SGX support to the hardware, it is possible to improve security and privacy in this term.

2.2 Storage Structure

In the common hardware architectures used in cloud computing environments today, data is transferred through several buses using multiple protocols to be used. The encrypted data is stored on the hard drive and will be transferred into enclaved memory. The encrypted data in the main memory will be cached using multiple levels of cache for performance

optimization. Also, page mappings between virtual and physical addresses of data are kept in the main memory and a transaction look-aside buffer (TLB) is used to cache the mappings in the page-table for better performance.

Each data transfer can be a point of side-channel attack. While the data is encrypted, the patterns of reads and writes, cache-misses, page-faults, and other interrupts and exceptions can be used by the adversary for launching attacks and recovering useful information.

2.3 Side-Channel Attacks on the Enclave

Multiple effective attacks and defenses on the enclave have been proposed by past research. Most of the attacks make use of the side-channel information that can be leaked and accessed by a modified malicious kernel. In these attacks, the malicious kernel retrieves useful information about the code and the data by changing the behavior of program execution or analyzing the page-faults and other possible points of data leakage.

One of the major points of attacks is targeting exceptions caused by the program. Since the exceptions are first acknowledged by the kernel before the program, a malicious OS can use them to infer activities and the behavior of the program according to the inputs which leads to recovering the actual data and code of the program.

Another kind of attack that can be triggered by a malicious OS is an interrupt-based attack. The kernel can trigger interrupts to control the execution of program.

Other attacks such as Stealthy Page Monitoring (SPM) attacks [10, 11] can also be effective. For a typical page-fault attack, the attacker can make all the page table entries invalid and restrict enclave programs access to all the pages and by causing page-faults, view the pattern of memory reads and writes by the program. This action will cause a lot of page-faults which lead to a high overhead which leads to attack detection by the victim, so the attacker is interested to retrieve this information and view the pattern without causing unwanted page-faults by the victim. In page monitoring attacks, the attacker monitors the bits representing the validity and whether a page is accessed or modified. The attacker is also able to clear these bits and view them again frequently. So, the pattern of accesses can be visible to the attacker without causing page-faults [10, 11].

2.4 Intel TSX

One of the technologies which have been came into notice for making benefit in defense against side-channel attacks is Intel TSX [7]. This technology has been introduced by Intel since Haswell mirco-architecture [1] for critical sections management and monitoring serialization. Intel TSX provides two frameworks to add transaction support to the processor. Both frameworks intoduce new instructions for region specification, backward compatiblity, and flexiblity [7]. We use RTM (Restricted Transactional Memory) [5] for implementing our defenses in this work.

RTM has been used by previous researches like T-SGX [9] to isolate instructions and Cloak [8] to pin sensitive code and data to the caches which is not supported by current hardware instructions.

3 ATTACKS

Several side-channel attacks on the enclave have been proposed by the past research. Focusing on the thorough work done by Wang et al. [11] we focus on summarizing the attacks and the points of attacks in this section.

3.1 Attack Model

In our attack scenario, the enclave is protected by Intel SGX, so no one except the main program running inside enclave has the key to access the enclave. The operating system kernel which runs on top of a hypervisor can view and manage tasks queue, exceptions and interrupts. The hypervisor kernel is also capable of viewing data transfer between memories and caches while Intel SGX prevents it from accessing the enclave data. It is considered that other tenants executing on the same hypervisor are not able to control the resources assigned to the main machine by the hypervisor.

3.2 Points of Attack

Several weaknesses in SGX architecture have been proposed to be points for launching attacks against the enclave.

TLB, as an address translation cache, can be one point of attack. While TLB is shared between enclave and non-enclave programs when hyper-threading is enabled, it can create side-channels which can be used for retrieving useful information [11].

Also, during AEX (Asynchronous EXit), TLB and paging-structure caches will be flushed and let the adversary know the flushed entries at context switch [11].

Another point of attack is the use of page table entry flags. The accessed and dirty bits which are set and unset for page table entries upon accesses and modifications, are used for launching attacks. So, the code in non-enclave mode can observe page-table updates and the memory write pattern to recover useful data [10, 11].

Other than TLBs, CPU caches are shared between the enclave and non-enclave code resulting in the possiblity of cache side-channel attacks [11].

3.3 Flush+Reload

By making use of page sharing between enclave and non-enclave code, the attacker can find out if a certain page is removed from the cache. More precisely, the attacker flushes the target memory line from the cache and waits for the victim to access it. The wait time can be fixed to either a random value or a value that is experimented to be good for the attack success. Then, the attacker will request to access the page, and measures the time it takes for the page to be fetched. Judging based on the time, the attacker can say if the page is already cached or fetched from the memory. In the case that it takes longer than a certain amount of time for the page to be retrieved, the attacker can conclude that it has not been fetched or touched by the victim [12].

3.4 Sneaky Page-Monitoring Attacks

Wang et al. [11] introduce three kinds of attacks for monitoring page-table entries with the consideration of the performance slowdown caused by inducing intentional page-faults. In the case that a page-fault is induced for each memory access by the victim, the number of page-faults and performance slowdown can be a point for attack detection by the victim.

3.4.1 Accessed flags monitoring. Accessed and dirty flags of page table entries, as discussed earlier, can be used to trigger attacks without causing page-faults. In this attack, the attacker monitors the set-unset pattern of the flags for page table entries and recovers information about the program running inside the enclave and encrypted data which is being accessed in the protected memory. A problem with this attack is that after address translations are cached in TLB, the flags are no longer updated in the page-table, but are updated in the TLB. Thus, it is needed to force the victim to walk its page-tables in order for the flags to

be updated. In this attack, the attacker flushes the TLB by initiating an inter-processor interrupt which causes TLB shootdown. Thus, the flags will need to be updated in the page-table before being cached in the TLB [10, 11].

3.4.2 Timing enhancement. High number of accesses to a page-table entry for a certain input, can result in lots of interrupt requests in accessed flags monitoring attack which can be detected by monitoring interrupt requests by the victim. To prevent this, by finding out two pages which are accessed by the victim repetitively, and measuring the time between these two page accesses, the input can be recovered. Thus, instead of multiple interrupts, one interrupt after each cycle is enough for clearing the TLB and successful attack.

3.4.3 TLB flushing through Hyper-Threading. To further removing the interrupts which are needed by the two aforementioned attacks, it is possible to make use of the fact that the TLB is shared between two threads which are being executed on the same core. Thus, by running the attacker program in the same core with the victim program, it is possible to invalidate TLB entries outside the enclave and without initiating interrupts.

4 DEFENSES

5 MMSN PROTOCOL

5.1 Frequency Assignment

We propose a suboptimal distribution to be used by each node, which is easy to compute and does not depend on the number of competing nodes. A natural candidate is an increasing geometric sequence, in which

$$P(t) = \frac{b^{\frac{t+1}{T+1}} - b^{\frac{t}{T+1}}}{b-1}, \tag{1}$$

where $t = 0, \ldots, T$, and b is a number greater than 1.

In our algorithm, we use the suboptimal approach for simplicity and generality. We need to make the distribution of the selected back-off time slice at each node conform to what is shown in Equation (1). It is implemented as follows: First, a random variable α with a uniform distribution within the interval $(0, 1)$ is generated on each node, then time slice i is selected according to the following equation:

$$i = \lfloor (T+1) \log_b [\alpha(b-1)+1] \rfloor.$$

It can be easily proven that the distribution of i conforms to Equation (1).

So protocols [**? ? ? ? ? ?**] that use RTS/CTS controls[1] for frequency negotiation and reservation are not suitable for WSN applications, even though they exhibit good performance in general wireless ad-hoc networks.

5.1.1 Exclusive Frequency Assignment. In exclusive frequency assignment, nodes first exchange their IDs among two communication hops so that each node knows its two-hop neighbors' IDs. In the second broadcast, each node beacons all neighbors' IDs it has collected during the first broadcast period.

Eavesdropping. Even though the even selection scheme leads to even sharing of available frequencies among any two-hop neighborhood, it involves a number of two-hop broadcasts. To reduce the communication cost, we propose a lightweight eavesdropping scheme.

[1]RTS/CTS controls are required to be implemented by 802.11-compliant devices. They can be used as an optional mechanism to avoid Hidden Terminal Problems in the 802.11 standard and protocols based on those similar to [**?**] and [**?**].

ALGORITHM 1: Frequency Number Computation

Input: Node α's ID (ID_α), and node α's neighbors' IDs within two communication hops.
Output: The frequency number ($FreNum_\alpha$) node α gets assigned.

```
index = 0; FreNum_α = -1;
repeat
    Rnd_α = Random(ID_α, index);
    Found = TRUE;
    for each node β in α's two communication hops do
        Rnd_β = Random(ID_β, index);
        if (Rnd_α < Rnd_β) or (Rnd_α == Rnd_β and ID_α < ID_β);
         then
            Found = FALSE; break;
        end
    end
    if Found then
        FreNum_α = index;
    else
        index ++;
    end
until FreNum_α > −1;
```

5.2 Basic Notations

As Algorithm 1 states, for each frequency number, each node calculates a random number (Rnd_α) for itself and a random number (Rnd_β) for each of its two-hop neighbors with the same pseudorandom number generator.

Bus masters are divided into two disjoint sets, $\mathcal{M}_{RT}$ and $\mathcal{M}_{NRT}$.

RT Masters $\mathcal{M}_{RT} = \{\vec{m}_1, \ldots, \vec{m}_n\}$ denotes the n RT masters issuing real-time constrained requests. To model the current request issued by an $\vec{m}_i$ in $\mathcal{M}_{RT}$, three parameters—the recurrence time (r_i), the service cycle (c_i), and the relative deadline (d_i)—are used, with their relationships.

NRT Masters $\mathcal{M}_{NRT} = \{\vec{m}_{n+1}, \ldots, \vec{m}_{n+m}\}$ is a set of m masters issuing nonreal-time constrained requests. In our model, each $\vec{m}_j$ in $\mathcal{M}_{NRT}$ needs only one parameter, the service cycle, to model the current request it issues.

Here, a question may arise, since each node has a global ID. Why don't we just map nodes' IDs within two hops into a group of frequency numbers and assign those numbers to all nodes within two hops?

6 SIMULATOR

If the model checker requests successors of a state which are not created yet, the state space uses the simulator to create the successors on-the-fly. To create successor states the simulator conducts the following steps.

(1) Load state into microcontroller model.
(2) Determine assignments needed for resolving nondeterminism.
(3) For each assignment.
  (a) either call interrupt handler or simulate effect of next instruction, or
  (b) evaluate truth values of atomic propositions.

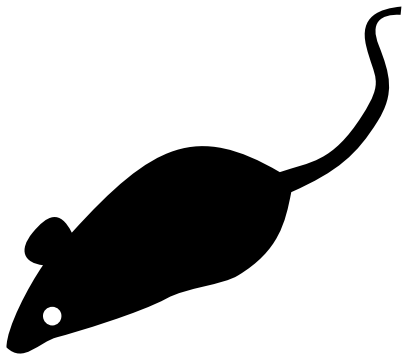

Fig. 1. Code before preprocessing.

(4) Return resulting states.

Figure 1 shows a typical microcontroller C program that controls an automotive power window lift. The program is one of the programs used in the case study described in Section 6. At first sight, the programs looks like an ANSI C program. It contains function calls, assignments, if clauses, and while loops.

6.1 Problem Formulation

The objective of variable coalescence-based offset assignment is to find both the coalescence scheme and the MWPC on the coalesced graph. We start with a few definitions and lemmas for variable coalescence.

Definition 6.1 (Coalesced Node (C-Node)). A C-node is a set of live ranges (webs) in the AG or IG that are coalesced. Nodes within the same C-node cannot interfere with each other on the IG. Before any coalescing is done, each live range is a C-node by itself.

Definition 6.2 (C-AG (Coalesced Access Graph)). The C-AG is the access graph after node coalescence, which is composed of all C-nodes and C-edges.

Lemma 6.3. *The C-MWPC problem is NP-complete.*

Proof. C-MWPC can be easily reduced to the MWPC problem assuming a coalescence graph without any edge or a fully connected interference graph. Therefore, each C-node is an uncoalesced live range after value separation and C-PC is equivalent to PC. A fully connected interference graph is made possible when all live ranges interfere with each other. Thus, the C-MWPC problem is NP-complete. □

Lemma 6.4 (Lemma Subhead). *The solution to the C-MWPC problem is no worse than the solution to the MWPC.*

Proof. Simply, any solution to the MWPC is also a solution to the C-MWPC. But some solutions to C-MWPC may not apply to the MWPC (if any coalescing were made). □

7 PERFORMANCE EVALUATION

During all the experiments, the Geographic Forwarding (GF) by Akyildiz et al. [**?**] routing protocol is used. GF exploits geographic information of nodes and conducts local data-forwarding to achieve end-to-end routing. Our simulation is configured according to the

Table 1. Simulation Configuration

TERRAIN[a]	(200m×200m) Square
Node Number	289
Node Placement	Uniform
Application	Many-to-Many/Gossip CBR Streams
Payload Size	32 bytes
Routing Layer	GF
MAC Layer	CSMA/MMSN
Radio Layer	RADIO-ACCNOISE
Radio Bandwidth	250Kbps
Radio Range	20m–45m

Source: This is a table sourcenote. This is a table sourcenote. This is a table sourcenote.
Note: This is a table footnote.

[a]This is a table footnote. This is a table footnote. This is a table footnote.

settings in Table 1. Each run lasts for 2 minutes and repeated 100 times. For each data value we present in the results, we also give its 90% confidence interval.

8 CONCLUSIONS

In this article, we develop the first multifrequency MAC protocol for WSN applications in which each device adopts a single radio transceiver. The different MAC design requirements for WSNs and general wireless ad-hoc networks are compared, and a complete WSN multifrequency MAC design (MMSN) is put forth. During the MMSN design, we analyze and evaluate different choices for frequency assignments and also discuss the nonuniform back-off algorithms for the slotted media access design.

9 TYPICAL REFERENCES IN NEW ACM REFERENCE FORMAT

A paginated journal article [**?**], an enumerated journal article [**?**], a reference to an entire issue [**?**], a monograph (whole book) [**?**], a monograph/whole book in a series (see 2a in spec. document) [**?**], a divisible-book such as an anthology or compilation [**?**] followed by the same example, however we only output the series if the volume number is given [**?**] (so Editor00a's series should NOT be present since it has no vol. no.), a chapter in a divisible book [**?**], a chapter in a divisible book in a series [**?**], a multi-volume work as book [**?**], an article in a proceedings (of a conference, symposium, workshop for example) (paginated proceedings article) [**?**], a proceedings article with all possible elements [**?**], an example of an enumerated proceedings article [**?**], an informally published work [**?**], a doctoral dissertation [**?**], a master's thesis: [**?**], an online document / world wide web resource [**? ? ?**], a video game (Case 1) [**?**] and (Case 2) [**?**] and [**?**] and (Case 3) a patent [**?**], work accepted for publication [**?**], 'YYYYb'-test for prolific author [**?**] and [**?**]. Other cites might contain 'duplicate' DOI and URLs (some SIAM articles) [**?**]. Boris / Barbara Beeton: multi-volume works as books [**?**] and [**?**].

A couple of citations with DOIs: [**? ?**].

Online citations: [**? ? ?**].

A SWITCHING TIMES

In this appendix, we measure the channel switching time of Micaz [**?**] sensor devices. In our experiments, one mote alternatingly switches between Channels 11 and 12. Every time after the node switches to a channel, it sends out a packet immediately and then changes to a new channel as soon as the transmission is finished. We measure the number of packets the test mote can send in 10 seconds, denoted as N_1. In contrast, we also measure the same value of the test mote without switching channels, denoted as N_2. We calculate the channel-switching time s as

$$s = \frac{10}{N_1} - \frac{10}{N_2}.$$

By repeating the experiments 100 times, we get the average channel-switching time of Micaz motes: 24.3 μs.

B SUPPLEMENTARY MATERIALS

B.1 This is an Example of Appendix Subsection Head

Channel-switching time is measured as the time length it takes for motes to successfully switch from one channel to another. This parameter impacts the maximum network throughput, because motes cannot receive or send any packet during this period of time, and it also affects the efficiency of toggle snooping in MMSN, where motes need to sense through channels rapidly.

By repeating experiments 100 times, we get the average channel-switching time of Micaz motes: 24.3 μs. We then conduct the same experiments with different Micaz motes, as well as experiments with the transmitter switching from Channel 11 to other channels. In both scenarios, the channel-switching time does not have obvious changes. (In our experiments, all values are in the range of 23.6 μs to 24.9 μs.)

B.2 Appendix Subsection Head

The primary consumer of energy in WSNs is idle listening. The key to reduce idle listening is executing low duty-cycle on nodes. Two primary approaches are considered in controlling duty-cycles in the MAC layer.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Maura Turolla of Telecom Italia for providing specifications about the application scenario.

The work is supported by the National Natural Science Foundation of China under Grant No.: 61273304_a and Young Scientists' Support Program (http://www.nnsf.cn/youngscientists).

REFERENCES

[1] Intel Corporation. [n. d.]. 4th Generation Intel Core Processor. Retrieved April 12, 2018 from https://software.intel.com/en-us/node/524022

[2] Intel Corporation. [n. d.]. ECALL/OCALL Functions. Retrieved April 12, 2018 from https://software.intel.com/en-us/node/709001

[3] Intel Corporation. [n. d.]. Intel Software Guard Extensions (Intel SGX). Retrieved April 12, 2018 from https://software.intel.com/en-us/sgx

[4] Intel Corporation. [n. d.]. Intel Software Guard Extensions (Intel SGX) SDK. Retrieved April 12, 2018 from https://software.intel.com/en-us/sgx-sdk

[5] Intel Corporation. [n. d.]. Restricted Transactional Memory Overview. Retrieved April 12, 2018 from https://software.intel.com/en-us/node/524022

[6] Intel Corporation. 2017. 6th Generation Intel Processor Family. Retrieved April 12, 2018 from https://www.intel.com/content/www/us/en/processors/core/desktop-6th-gen-core-family-spec-update.html

[7] Intel Corporation. 2017. Intel Transactional Synchronization Extensions (Intel TSX) Overview. Retrieved April 12, 2018 from https://software.intel.com/en-us/node/524022

[8] Daniel Gruss, Julian Lettner, Felix Schuster, Olya Ohrimenko, Istvan Haller, and Manuel Costa. 2017. Strong and efficient cache side-channel protection using hardware transactional memory. In *USENIX Security Symposium*.

[9] Ming-Wei Shih, Sangho Lee, Taesoo Kim, and Marcus Peinado. 2017. T-SGX: Eradicating controlled-channel attacks against enclave programs. In *Proceedings of the 2017 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA*.

[10] Jo Van Bulck, Nico Weichbrodt, Rüdiger Kapitza, Frank Piessens, and Raoul Strackx. 2017. Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution. In *Proceedings of the 26th USENIX Security Symposium*. USENIX Association.

[11] Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, Xiaofeng Wang, Vincent Bindschaedler, Haixu Tang, and Carl A. Gunter. 2017. Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX. 2421–2434.

[12] Yuval Yarom and Katrina Falkner. 2014. FLUSH+ RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack.. In *USENIX Security Symposium*. 719–732.