AFRL-RI-RS-TR-2022-047

SIDE-CHANNEL LEAKAGE IN CRYPTOGRAPHIC PROTOCOLS ON COMPLEX DEVICES SOFTWARE

REGENTS OF THE UNIVERSITY OF MICHIGAN

MARCH 2022

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2022-047 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
CARL E. THOMAS
Work Unit Manager

/ S /
GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing and Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
MARCH 2022	FINAL TECHNICAL REPORT	SEPTEMBER 2019	SEPTEMBER 2021

4. TITLE AND SUBTITLE

SIDE-CHANNEL LEAKAGE IN CRYPTOGRAPHIC PROTOCOLS ON COMPLEX DEVICES SOFTWARE

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8750-19-C-0531	N/A	61101E

5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER
		R2W1

6. AUTHOR(S)

Daniel Genkin, Ayush Agarwal, Ingab Kang, Andrew Kwong, Marina Minkin, Stephan van Schaik

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Regents of the University of Michigan, Office of Research and Sponsored Projects
503 Thompson St
Ann Arbor MI 48109-1340

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

AFRL-RI-RS-TR-2022-047

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this project, we study the presence of microarchitectural attacks on complex devices, as well as the impact of these attacks on cryptographic protocols. We and several new side channel leakage sources on computer based systems. We then proceed to empirically evaluate the leakage available using these channels, as well as discussion attack scenarios. Finally, we quantify the impact of our newly discovered leakage sources on cryptographic primitives and protocols, demonstrating secret key recovery.

15. SUBJECT TERMS

Side Channel Attack, Intel, Microprocessor, Leakage, Cryptography

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	26

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
CARL E. THOMAS	315-330-2600

PREVIOUS EDITION IS OBSOLETE.

Contents

Table of Figures

1 SUMMARY

Since their introduction over a decade ago, microarchitectural side channel attacks have been proven to be a severe threat to the security of numerous computer systems. These attacks, which exploit contention on microarchitectural resources, have been shown to break cryptography [1, 2, 23, 32, 35, 37, 50, 51], bypass ASLR [13, 21], and more recently completely bypass hardware security boundaries [18, 19, 28, 31], including Intel SGX [42].

Recognizing the apparent danger, developers of cryptographic libraries have attempted to deploy countermeasures against side channel attacks. Examples of such countermeasures include various blinding techniques to prevent timing attack [8], attempts to prevent microarchitectural attacks by refactoring old cryptographic code to be constant time, formally verifying cryptographic code for the presence of side channel leakage [7, 52], or even highly-regular bottom up designs taking microarchitectural attacks in mind [4, 30]. While side channel defense strategies differ, one thread remains the same. A defense is only as good as the mental leakage model implicitly assumed by its designer. Unfortunately, as proven by the latest revelations in side channel research [28, 31, 42, 46], much remains to be done in order to understand what information can be leaked by a side channel adversary, and howto systematically protect systems against side channel attacks.

Project Methodology. In this project we aimed at obtaining a better understanding of the risk posed by side channel attacks to basic cryptographic primitives and the protocols built on top of them. Tothat aim, we first tackled the need of tedious manual analysis of cryptographic code, creating a comprehensive tool set for mounting side channel attacks. Next, we applied these tools to explore novel side channels and leakage sources, as well as unexpected implications of existing channels. Finally, we empirically tested existing side channel countermeasures present in cryptographic libraries, as well as designed new countermeasures to protect cryptographic code against the latest generation of side channel attacks.

Project Accomplishments. In addition to creating a systematic tool kit for mounting side channel attacks on cryptographic primitives, during the course of this project we have discovered multiple novel side channel issues with CPUs made by multiple vendors. These included new speculative and transient execution attacks on Intel and AMD CPUs, as well as the first microarchitectural side channel attack on Apple's newly released M1 CPU. We have also discovered side channel attacks on advanced cryptographic primitives, such as random number generators and cryptographic primitives standardized by Asian countries. Next, moving away from CPU-related issues, we have discovered integrity issues present in memory sticks, leading to new techniques for mounting Rowhammer attacks. Finally, going beyond the microarchitecture, we have discovered several physical side channel attacks on commodity phones and smart speaker devices.

Impact. Over its two year span, this project has directly funded 4 PhD students. It has resultedin ten publication in top tier security venues, garnering over 400 citations and multiple followup works. Countermeasures designed by hardware vendors in collaboration with the PI has been deployed on millions on machines, OS installations and numerous cryptographic libraries. Finally, in addition to being integrated into the security curricula in many top universities, this work has obtained a large amount of visibility in technical and general public media (e.g., The New York Times, CNN, BBC, etc.).

2 INTRODUCTION

Data isolation is a basic security guarantee underlying nearly all computer systems: an operating system must be isolated from user processes, while processes must not be able to access each other's memory. Similarly, virtual machines running on third-party clouds must be separated and protected from one another to prevent da6ta leakage. Recently, Intel Secure Guard Extensions (SGX) further allows developers to isolate their code and memory inside enclaves which are not accessible to even the (untrusted) operating system.

However, isolation is a mere allusion, as all software runs on common and often leaky hardware. Indeed, in 2018 we have seen the fragility of this abstraction, with Spectre and Meltdown [28, 31] showing that attackers can bypass nearly every possible isolation domain. Making things worse, the discovery of Spectre and Meltdown was not an isolated incident, but instead have opened a Pandora's box of CPU vulnerabilities. Indeed, since the introduction of transient execution attacks, it has been shown that side channels can bypass numerous hardware-enforced security boundaries, including user-from-kernel isolation [28, 31], guest-from-host and guest-from-guest virtual machine isolation [46], Intel's SGX technology [42], as well as others [6, 10, 12, 20, 24, 27, 28, 29, 33, 34]. Most worrisome, despite transient execution countermeasures being deployed both in software and in hardware [3, 16, 22, 36, 41, 47], new transient execution attacks are being repeatedly discovered [9, 39, 43, 45], with each attack requiring its own dedicated and delicate countermeasure design.

Next, side channels themselves are unfortunately not limited to just the CPU. More specifically, Rowhammer [14, 26, 40] is a fault attack, in which the attacker uses a specific sequence of memory accesses that results in bit flips, i.e., changes in bit values, in memory locations other than those accessed. Because the attacker does not directly access the changed memory locations, the changes are not visible to the processor or the operating system, and are not subject to any permission checks. Thus far, this ability to reliably flip bits across security boundaries has been exploited for sandbox escapes [14, 40], privilege escalation attacks on operating systems and hypervisors [14, 17, 38, 40, 44, 48], denial-of-service attacks [17, 25], and even for fault injection in cryptographic protocols [5, 38].

Recognizing the apparent danger resulting from the arbitrary bypasses of isolation boundaries, developers of cryptographic libraries have attempted to deploy countermeasures against side channel attacks. Examples of such countermeasures include various blinding techniques to prevent timing attack [8], attempts to prevent microarchitectural attacks by refactoring old cryptographic code to be constant time, formally verifying cryptographic code for the presence of side channel leakage [7, 52], highly-regular bottom up designs taking microarchitectural attacks in mind [4, 30], or speculative load hardening aimed at preventing Spectre attacks [11].

Given the myriad of defense approaches, in this project we study the following main question.

Are existing side channel defenses sufficient to protect cryptographic primitives against side channel attacks? What would it take for a side channel attacker to violate the security of these primitives and how can such attacks be mitigated?

3 METHODS, ASSUMPTIONS AND PROCEDURES

In this project we aim to further understand the risk posed by side channel attacks to basic cryptographic primitives and the protocols built on top of them. Currently, side channel attacks heavily rely on tedious manual analysis of cryptographic code as well as prototyping using unstable graduate-level attack code. Thus, our first step in this project aimed at alleviating both issues by creating a comprehensive tool set for mounting side channel attacks as well as automaticallydetecting key-dependent leakage produced by cryptographic primitives.

3.1 Mastik

Mastik [49] is a side-channel toolkit that is commonly used for mounting (non-transient) cache-based side-channel attacks. Mastik currently supports several side-channel leakage channels, such as Flush+Reload [50], L1-D Prime+Probe [35, 37], L1-I Prime+Probe [1, 2], LLC Prime+Probe [23, 32], Flush+Flush [15] and CacheBleed [51]. In addition, Mastik includes a GUI for side channel signal visualization and leakage analysis.

We have advertised Mastik in two summer schools (FICHSA in Ramat Gan, Israel, and SILM in Rennes, France). Mastik seems to be gaining traction, and has attracted 31 citations as well as being used by Intel. We're now in the process of preparing Mastik as an open-source project. Feedback from the community is encouraging and we hope for increased community support for the project, once released.

3.2 CacheFX

Over the last two decades, the danger of sharing resources between programs has been repeatedly highlighted with the advent of multiple cache-based side channel attacks. In response, the research community has proposed multiple cache designs that aim at curbing these information leaks.

With multiple competing designs, there is a need for assessing the level of security against side-channel attacks that each design offers. Tackling this issue, we have built a flexible framework for assessing and evaluating the resilience of cache designs to side-channel attacks. Our framework allows the evaluator to implement various cache designs, victims, and attackers, as well as to exercise them for assessing the leakage of information via the cache.

More specifically, we implement nine cache designs, including traditional fully-associative and set-associative caches, PLCache, Newcache, PhantomCache, ScatterCache, way-partitionedcaches, and the two variants of CEASER. For caches that do not stipulate a replacement policy, we support four replacement algorithms: random replacement, least recently used (LRU), and two variants of pseudo-LRU. We then use our framework to evaluate these caches using several metrics.

Not surprisingly, we find that different metrics highlight different aspects of cache design. In particular, we find that building eviction sets is faster in skewed caches such as ScatterCache and CEASERS, than in randomized caches, such as fully associative caches or PhantomCache. Faster eviction-set construction reduces the effort required for mounting the attack. At the same time, our experiments show that, with the right parameters, skewed caches are not less secured than randomized cache when it comes to cryptographic attacks. More specifically, we find that the security against cryptographic attacker depends not only on the design, but also on other parameters, such as the replacement algorithm used and the cache associativity. We also show that all non-partitioned

caches are vulnerable to both eviction-set and occupancy attacks, often providing significantly lower security guarantees compared to their design goals. Finally, we show that partitioned caches are not necessarily a solution for cache-based side channel attacks, because they can only support a limited number of concurrent processes and time-sharing them raises a potential for further leakage.

4 RESULTS

4.1 Attacking Advanced Cryptographic Primitives

Modern cryptography requires the ability to securely generate pseudorandom numbers. However, despite decades of work on side channel attacks, there is little discussion of their application to pseudorandom number generators (PRGs). In this work we set out to address this gap, empirically evaluating the side channel resistance of common PRG implementations.

We find that hard-learned lessons about side channel leakage from encryption primitives have not been applied to PRGs, at all levels of abstraction. At the design level, the NIST- recommended CTR DRBG design does not have forward security if an attacker is able to compromise the state via a side-channel attack. At the primitive level, popular implementations of CTR DRBG such as OpenSSL's FIPS module and NetBSD's kernel use leaky T-table AES as their underlying block cipher, enabling cache side-channel attacks. Finally, we find that many implementations make parameter choices that enable an attacker to fully exploit the side-channel attack in a realistic scenario and recover secret keys from TLS connections.

4.2 Fallout: Leaking Data on Meltdown-Resistant CPUs

We have discovered a new speculative attack technique called Fallout. Fallout allows reading information from the CPU's store buffer, which is used every time a CPU pipeline needs to store any data. We show that a performance optimization, well documented in Intel's patents, causes the CPU to serve data from incorrect addresses during the resolution of faulty loads. This allows an attacker to break Kernel Address Space Layout Randomization (KASLR), as well as to leak sensitive data written to memory by the operating system kernel. Perhaps most ironically, Fallout is most effective against the 9th generation of Intel processors, which were designed to be resistant to Meltdown-like attacks.

4.3 Light Commands: Laser-Based Audio Injection Attacks on Voice-Controllable Systems

We have identified a semantic gap between the physics and specifications of MEMS (microelectro-mechanical systems) microphones, where such microphones unintentionally respond to light
as if it was sound. Exploiting this effect, we can inject sound into microphones by simply modulating the amplitude of a laser light. Next, we investigated the vulnerability of popular

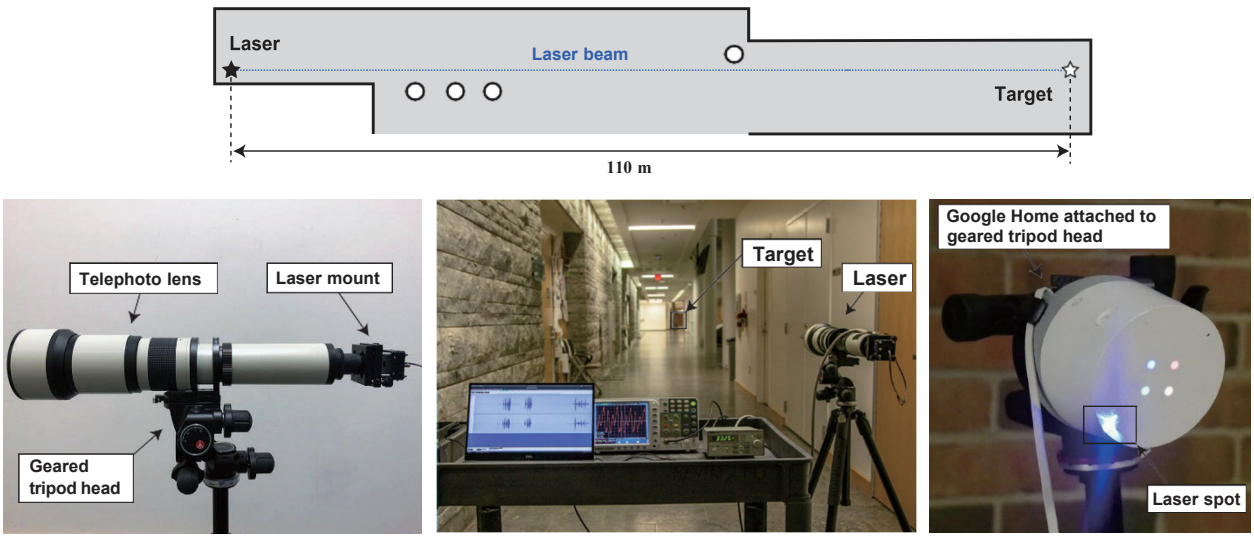

Figure 1: Experimental setup for exploring attack range. (Top) Floor plan of the 110 m long corridor. (Left) Laser with telephoto lens mounted on geared tripod head for aiming. (Center) Laser aiming at the target across the 110 m corridor. (Right) Laser spot on the target device mounted on tripod.

VC systems (such as Alexa, Siri, Portal, and Google Assistant) to light-based audio injection attacks. We find that 5 mW of laser power (the equivalent of a laser pointer) is sufficient to obtain full control over many popular Alexa and Google smart home devices, while about 60 mW is sufficient for gaining control over phones and tablets. See Figure 1.

Following the practice of responsible disclosure we have notified Apple, Amazon and Google regarding our finding. We have also published our findings at https://lightcommands.com/ as well as in Usenix Security 2020.

4.4 Load Value Injection

Recent research on transient-execution attacks has been characterized by a sharp split between on the one hand Spectre-type misspeculation attacks and on the other hand, Meltdown-type data extraction attacks. The first category, Spectre-type attacks, exploits the CPU's prediction machinery to trick a victim protection domain into speculatively executing instructions out of its intended execution path by "poisoning" the shared branch predictor. Spectre-type attacks require careful identification of "confused deputy" code patterns, dubbed gadgets, that when unintentionally transiently executed expose victim secrets through the microarchitectural state. The second category, Meltdown-type attacks, targets architecturally inaccessible data by exploiting illegal data flow from faulting or assisted instructions. The results of unauthorized loads are forwarded to subsequent transient operations which may encode the data before an exception is eventually raised. Over the past year, delayed exception handling and microcode assists have been shown to transiently expose data from various microarchitectural elements and across many security domains.

We set out to explore a new class of transient execution attacks, called Load Value Injection (LVI) attacks. We observed that under certain adversarial conditions, unintended microarchitectural leakage can also be inverted to *inject* incorrect data into the victim's

transient control/data flow. Being essentially a "reverse Meltdown"-type attack, we were able to abuse faulting or assisting load instructions executed within a victim domain to transiently forward incorrect values or attacker controlled data from various microarchitectural buffers. This data can be subsequently leaked from the victim program or used in order to corrupt the victim's control flow, causing it to execute instructions outside of its intended control flow.

The LVI paper has been published at IEEE Security and Privacy 2020, and has received a significant amount of media attention during public release. Given the impact on SGX enclaves, Intel needed to harden enclave code by removing LVI gadgets. This has resulted in a dedicated compiler mitigation and the addition of fencing instructions in critical places in order to limited speculation.[1] Finally, similar techniques were applied to Singal's SGX enclave in an attempt to harden Signal's private contact discovery service against LVI attacks.

4.5 Attacking Pilsung, the North Korean Block Cipher

Over the past two decades, cache attacks have been identified as a threat to the security of cipher implementations. While much is known about attacks on widely used ciphers such as AES, some nations have elected to use other AES variants which differ substantially from AES'original design.

In this thrust, we investigated an implementation of the North Korean cipher Pilsung, as reverse-engineered by Kryptos Logic. Like AES, Pilsung is a permutation-substitution cipher, but unlike AES, both the substitution and the permutation steps in Pilsung depend on the key, and are not known to the attacker. While cache attacks have been applied to ciphers that have fixed state transformations, it is unknown whether using secret, key- dependent, transformations enhances the cipher's security against side channels. Analyzing Pilsung, improve the state side channel techniques by developing techniques for reversing secret-dependent transformation and recovering information from such transformations using side channels. Our attack, which requires an average of eight minutes on a typical laptop computer, demonstrates that secret transformations do not necessarily protect ciphers against side channels. Finally, we show that Pilsung's secret-dependent structure makes obtaining a constant time software implementation harder, thus further facilitating side channel leakage.

4.6 CacheOut: Leaking Data on Intel CPUs via Cache Evictions

Recent speculative execution attacks, such as RIDL, Fallout, and ZombieLoad, demonstrated that attackers can leak information while it transits through various microarchitectural buffers. Named Microarchitectural Data Sampling (MDS) by Intel, these attacks are likened to "drinking from the firehose", as the attacker has little control over what data is observed and from what origin. Unable to prevent these buffers from leaking, Intel issued countermeasures via microcode updates that overwrite the buffers when the CPU changes security domains.

In this research thrust we investigated the feasibility of using a flushing-based approach in order to stop MDS attacks. Unfortunately our results indicate that the ad-hoc buffer

[1] https://software.intel.com/security-software-guidance/insights/deep-dive-load-value-injection

overwrite countermeasures are insufficient, allowing the attacker to nonetheless recover data from the leaky buffers and even select which cache sets to read from the CPU's L1 Data cache. Finally, we show that CacheOut is applicable to nearly all hardware-backed security domains, including process to process and process to kernel isolation and virtual machine boundaries.

4.7 CacheOut Attacks on SGX

Intel's Software Guard Extensions (SGX) promises an isolated execution environment, protected from all software running on the machine. However, a significant limitation of SGX is its lack of protection against side-channel attacks. In particular, recent works such as CacheOut and Foreshadow have shown that transient-execution attacks can leak arbitrary data from SGX, breaching SGX's confidentiality. However, less work has been done on the implications of such attacks on the SGX ecosystems.

Indeed, going beyond attacks on SGX's confidentiality properties, we extended CacheOut to also breach SGX enclaves' integrity. Specifically, we target the SGX sealing mechanism, which provides long-term storage of data, by securely generating an encryption key used to seal data by encrypting it before forwarding it to the operating system for long-term storage. We use CacheOut to extract the sealing key from a victim enclave. The main challenge is that the SGX SDK implementation of the sealing API wipes out the sealing key immediately after using it, leaving a short window in which the key can be retrieved. After recovering the sealing key, we use it to unseal and read the sealed information, then modify and reseal it. As SGX provides no integrity mechanism to detect such a change, the victim enclave now operates on data corrupted by the attacker.

Next, we turned our attention to SGX's remote attestation protocol, which allows an enclave to prove to a remote party that it has been initialized correctly and is executing on a genuine (presumably secure) Intel processor. To attack attestation, we first used our attack on SGX's sealing mechanism and retrieve the sealing key of the SGX Quoting Enclave. Notably, unlike previous works from 2019 and 2020 which build and sign their own quoting enclave, we attack a genuine, Intel-signed, production enclave, which employs all of Intel's countermeasures for transient-execution attacks, including the recent hardening for the LVI attack. We then use the retrieved sealing key to unseal the persistent storage of the Quoting Enclave, which contains the private attestation key. We note that this attestation key is the only differentiating factor between a fully secure SGX enclave running on genuine Intel hardware and a malicious SGX simulator offering no security guarantees. Thus, we can build a malicious SGX simulator that passes Intel's entire remote attestation process. Performing remote attestation with the provider's server is usually the first task of any enclave running on the user's machine, resulting in a secure connection between the user's enclave and provider's servers. With the machine's production attestation keys compromised, any secrets provided by server are immediately readable by the client's untrusted host application, while all outputs allegedly produced by enclaves running on the client cannot be trusted for correctness. Finally, our ability to read enclave contest and fully pass remote attestation also erodes trust in SGX based secure remote computation protocols as users cannot trust that their data will be properly protected by a genuine SGX enclave.

4.8 EM Attacks on Constant Time Code

Recognizing the danger stemming from information leakage via side channels, the cryptographic community attempted to protect code against non-speculative side channels by establishing the notion of constant-time code. At a high level, the aim is to avoid secret dependent control flow and memory access patterns, thereby decoupling the program's secrets from effects which are easily observable through physical and microarchitectural leakage channels.

Due to this carefully imposed regularity and secret-independence, and aided by the complexity and high processing speed of the GHz-scale CPUs used in computer and mobile type of devices, it is believed that constant-time code offers some protection against simple power and EM analysis. However, given the empirical nature of this belief, we set out to explore the resilience to physical side-channel attacks afforded by deployed constant-time elliptic curve implementations running on mobile platforms.

We show that, unfortunately, constant-time elliptic curve implementations running on full fledged mobile platforms can be vulnerable to simple EM analysis. More specifically, we evaluated four constant-time implementations of elliptic curve digital signatures (Libgcrypt, OpenSSL, HACL* and Curve25519-donna) all of which use practices of constant-time coding for side channel protection. We demonstrate that even when the attacker only has access to coarse and low bandwidth information due to the complexity and speed of the targeted device, accurate key recovery attacksare still possible.

More specifically, using only a handful of profiling traces taken from a device similar to the target device, our attacks are able to extract the secret key using the EM leakage recorded during only a single digital signature. Moreover, our attacks only rely on inexpensive hardware (e.g., an $800 software defined radio) and use only 40 MHz of analog bandwidth to extract a key from Android phones operating multiple cores at the 1.1—1.4 GHz scale (see Figure 2). Despite having only about 0.03 samples per cycle, no hardware-based cycle-accurate artificial triggering, and noise resulting from normal Android background activity, our attack is capable of extracting about 90%of the key bits within seconds. Finally, we note that our attack is applicable to Curve22519 which uses the constant time swap operation standardized as part RFC 7748. We benchmark our attack on the Curve25519-donna reference implementation, on the wildly deployed OpenSSL implementation, as well as on the implementation of HACL* which is formally verified to be constant-time.

4.9 Evaluating Side Channel Countermeasures in Browsers

The "eternal war in cache" has reached browsers, with multiple cache-based side-channel attacks and countermeasures being suggested. A common approach for countermeasures is to disable or restrict JavaScript features deemed essential for carrying out attacks. A recent proposal following this approach is Chrome Zero, a browser add-on that intercepts and changes the semantics of some JavaScript features, while claiming minimal impact on user experience.

We demonstrate the fallacy of this approach. We show that controlling and disabling JavaScript features may attenuate but does not completely prevent side-channel attacks. We follow a line of research that performs website fingerprinting attacks. We develop a sequence:

Figure 2: Experimental setup for capturing EM emanations from ZTE ZFIVE (left) and Alcatel Ideal (right) phones. In each setup, our custom probe (the flat, beige, circular object at the end of the silver-colored cable) is positioned close to but not touching the phone. The cable is held in position by a mechanical arm, which is visible in the photo on the right. An Ettus B200-mini SDR (white box) digitizes the signal and sends it through a USB cable to a personal computer (not shown) for analysis.

of attacks with progressively decreasing dependency on JavaScript features, culminating in the first browser-based side-channel attack which is constructed entirely from Cascading Style Sheets (CSS), and therefore works even when script execution is completely blocked. Finally, we show that our attacks are effective against security hardened browsers, such as Tor and DeterFox.

4.10 Software-Visible Physical Side Channels.

Unlike microarchitectural attacks which rely on minute timing variations inside the CPU in order to extract secret information, physical side channel attacks extract information via leveraging externally-observable physical effects such as power consumption and electromagnetic radiation. However, as such attacks require proximity of the target device to an attacker-controlled external measurement equipment, they are traditionally considered to be "outside of the threat model".

Distinguishing Operations Using Soundcards. Exploring the feasibility of distinguishing different CPU operations using soundcard interfaces, Figure 3 shows a spectogram of a recording done using the built-in soundcard interface of an Acer S3 laptop, while the laptop was executing 1 second loop of various CPU operations. As can be seen, every operation has its own, unique and stable spectral signature, allowing for an easy identification.

Observing ECDSA Signatures. Next, we have attempted to observe the signal corresponding to ECDSA signatures performed by the CPU running GnuPG (a popular cryptographic library) as the soundcard was recording. Indeed, Figure 4 (left) is a spectrogram

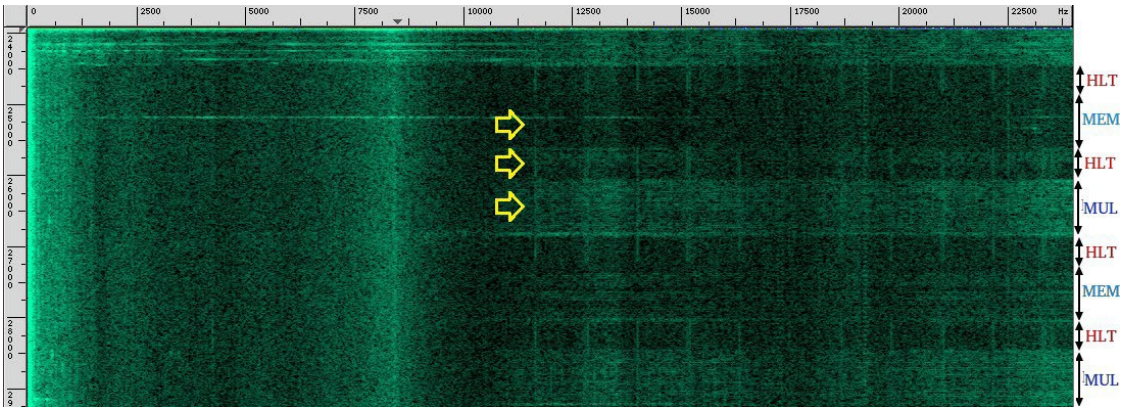

Figure 3: Spectrogram of a recording of 1 second loops of various CPU operations using the built in soundcard interface. Notice that the differences between HLT, MEM and MUL are clearly visible (marked by the Yellow arrows).

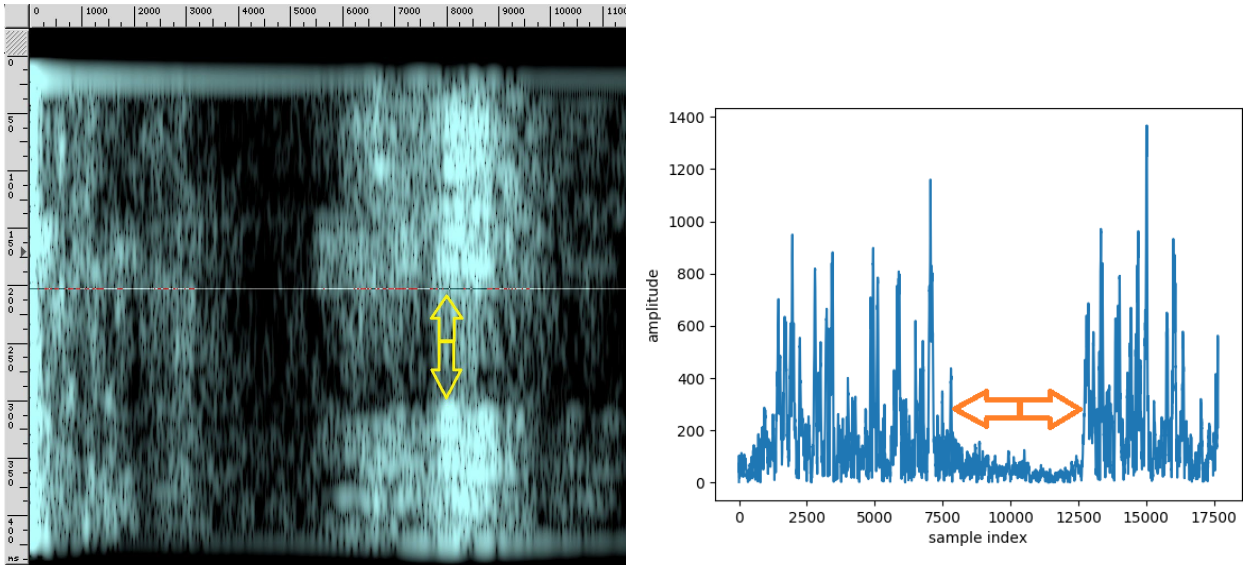

Figure 4: (Left) Spectrogram of a recording of an ECDSA operation done via the soundcard interface. Notice the gap in the 8 kHz signal (marked by the yellow arrow). (Right) The same signal plotted in time domain after signal analysis. Notice the gap marked by the orange arrows corresponding to the ECDSA signature.

of the obtained recording. Notice that the ECDSA operation interrupts the 8 kHz carrier which is present while the laptop is idle. Next, further analyzing the signal in time domain by demodulating the 8 kHz carrier (Figure 4 (right)), the ECDSA signature manifested as significant drop in signal intensity, thereby revealing the length of the ECDSA signing operation.

Attack Scenarios. Empirically demonstrating software-based physical attacks, we conduct physical side channel attacks on computation by remote and purely passive analysis of commonly-shared channels. These attacks require neither physical proximity (which could be mitigated by distance

and shielding), nor the ability to run code on the target or configure its hardware. We analyze the computation-dependent leakage captured by internal microphones, and empirically demonstrate its efficacy for attacks. In one scenario, an attacker steals the secret ECDSA signing keys of the counter-party in a voice call. In another, the attacker detects what web page their counterparty is loading. In the third scenario, a player in the Counter-Strike online multiplayer game can detect a hidden opponent waiting in ambush, by analyzing how the 3D rendering done by the opponent's computer induces faint but detectable signals into the opponent's audio feed.

4.11 Attacking Kalyna, the Ukrainian Block Cipher

Since the seminal work of Kocher, side-channel attacks have become a major threat to the security of virtually any cryptographic primitive. Here, most research attention have been given to western standards such as AES, RSA, or Elliptic Curve Cryptography, presumably due to their wide spread adoption, readily-available standardization documents, and acceptability of reference implementations. Much less attention in comparison has been given to national cipher standards of former Eastern Bloc countries, which often deploy their own cryptography standards, possibly due to their inherent distrust of the West and its intelligence community.

While modern symmetric Eastern Bloc designs are often based on AES, these often include variations in features such as round function and key expansion, due to different trade-offs between security and performance made by local standardization agencies while adapting the cipher to local use. As side-channel attacks are by their very nature implementation specific, it is unclear how such local adaptations impact the cipher's side-channel resistance.

Tackling this issue, in the final thrust of this report investigate the Ukrainian cipher Kalyna as a case study of an Eastern Bloc cipher. At a high level, Kalyna is modeled after AES, but has some important differences. First, Kalyna's key expansion algorithm is considered to be non- reversible. Thus, a side-channel adversary that wishes to attack Kalyna needs to retrieve all of theround keys. Moreover, Kalyna uses a non-linear arithmetic addition operation for pre- and post- whitening. This is known to hinder cryptanalysis. Despite these changes, we present the first cache attack against Kalyna, resulting in the attacker extracting the secret key from Kalyna- 128/128's reference implementation with a 97% success probability and an expected complexity of $2^{43.58}$. As aiming to recover the round keys would require a large number of traces, we instead focus on Kalyna's master key. We demonstrate that Kalyna's computation of the round keys from the master key propagates differences between internal values, in a way that allows using them for recovering the master key. To the best of our knowledge, this is the first demonstrated attack against the key expansion of any block cipher, and is the first attack that recovers the master key of Kalyna. We further demonstrate that the additive key whitening used in Kalyna is more vulnerable to cache attacks than the more common Boolean whitening. The main cause is that due to carry ripples during addition, an attacker can use a simple binary search to find the key. The attack requires only seven (chosen) plaintexts to completely recover the key.

4.12 Spook.js: Attacking Google Chrome's Strict Site Isolation via Speculative Execution and Type Confusion

As a final research thrust, we have investigated the feasibility of mounting Spectre attacks in browser environments, such as the Chrome browser. More specifically, following the discovery of Spectre, Chrome deployed a mitigation strategy called strict site isolation, which isolates every resource present on a loaded webpage to its own dedicated address space.

Unfortunately, in this thrust we present Spook.js, a new transient-execution attack capable of extracting sensitive information despite Chrome's strict site isolation architecture. Moreover, Spook.js can overcome Chrome's 32-bit sandboxing countermeasures, reading the entire address space of the rendering process. Additionally, we demonstrate a gap between Chrome's eTLD+1-based consolidation policy and the same-origin policy typically used for security in web context. The discrepancy can results in mutually distrusting security domain co-residing in the same address space, allowing one subdomain to attack another. Going beyond Chrome, we show that Chromium-based browsers such as Edge and Brave are also vulnerable to Spook.js. Finally, we show that Spectre adversely affect the security model of extensions in Chrome, demonstrating usernames and passwords leaks from the LastPass password manager.

4.13 Publications in Peer Reviewed Conferences

Over its two years, the project has supported the following publications.

1. Claudio Canella, Daniel Genkin, Lukas Giner, Daniel Gruss, Moritz Lipp, Marina Minkin, Daniel Moghimi, Frank Piessens, Michael Schwarz, Berk Sunar, Jo Van Bulck, Yuval Yarom:**Fallout: Leaking Data on Meltdown-resistant CPUs**, in CCS 2019.

2. Shaanan Cohney, Andrew Kwong, Shahar Paz, Daniel Genkin, Nadia Heninger, Eyal Ro- nen, Yuval Yarom: **Pseudorandom Black Swans: Cache Attacks on CTR_DRBG**, in IEEE Symposium on Security and Privacy 2020.

3. Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, Kevin Fu: **Light Commands: Laser-Based Audio Injection Attacks on Voice-Controllable Systems**, in USENIX Security Symposium 2020.

4. Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yuval Yarom, Berk Sunar, Daniel Gruss, Frank Piessens: **LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection**, in IEEE Symposium on Security and Privacy 2020.

5. Daniel Genkin, Romain Poussier, Rui Qi Sim, Yuval Yarom, Yuanjing Zhao: **Cache vs. Key-Dependency: Side Channeling an Implementation of Pilsung** in IACR Transactions in Cryptographic Hardware and Embedded Systems 2020.

6. Stephan van Schaik, Marina Minkin, Andrew Kwong, Daniel Genkin, Yuval Yarom: **CacheOut: Leaking Data on Intel CPUs via Cache Evictions**, in IEEE Symposium on Security and Privacy 2021.

7. Anatoly Shusterman, Ayush Agarwal, Sioli O'Connell, Daniel Genkin, Yossi Oren, Yu- val Yarom: **Prime+Probe 1, JavaScript 0: Overcoming Browser-based Side- Channel Defenses**, in USENIX Security Symposium 2021.

8. Alam Monjur, Baki Yilmaz, Frank Werner, Niels Samwel, Alenka Zajic, Daniel Genkin, Yuval Yarom, Milos Prvulovic: **Nonce@Once: A Single-Trace EM Side Channel Attack on Several Constant-Time Elliptic Curve Implementations in Mobile Platforms**, in IEEE European Symposium on Security and Privacy 2021.

9. Ayush Agarwal and Sioli O'Connell and Jason Kim and Shaked Yehezkel and Daniel Genkin and Eyal Ronen and Yuval Yarom: **Spook.js: Attacking Chrome Strict Site Isolation via Speculative Execution**, in EEE Symposium on Security and Privacy 2022.

5 CONCLUSION

As outlined in this research, microarchitectural side channel attacks have been proven to be a severe threat to the security of numerous computer systems. From speculative execution attacks to Rowhammer induced bit flips, it seems that nearly every performance enchaining feature can be used for mounting side channel attacks. Making things worse, software running on computer based systems seems to be extremely vulnerable to side channels, with targets ranging from cryptographic primitives to basic software isolation mechanisms. While this project aims to discover, evaluate and mitigate some of these issues, much remains to be done in order to understand side channel attacks and how to best mitigate them.

REFERENCES

[1] Onur Acıiçmez. Yet another microarchitectural attack: exploiting I-Cache. In *CSAW*, pages 11–18, 2007.

[2] Onur Acıiçmez and Werner Schindler. A vulnerability in RSA implementations due to instruction cache analysis and its demonstration on OpenSSL. In *CT-RSA*, pages 256–273, 2008.

[3] ARM. Arm v8.5-a cpu updates. https://developer.arm.com/support/ arm-security-updates/speculative-processor-vulnerability/downloads/ arm-v8-5-a-cpu-updates, June 2019.

[4] Daniel J. Bernstein. Curve25519: new Diffie-Hellman speed records. URL https: //cr.yp.to/ecdh.html.

[5] Sarani Bhattacharya and Debdeep Mukhopadhyay. Curious case of Rowhammer: Flipping secret exponent bits using timing analysis. In *CHES*, 2016.

[6] Atri Bhattacharyya, Alexandra Sandulescu, Matthias Neugschwandtner, Alessandro Sorniotti, Babak Falsafi, Mathias Payer, and Anil Kurmus. SMoTherSpectre: Exploiting speculative execution through port contention. In *CCS*, 2019.

[7] Barry Bond, Chris Hawblitzel, Manos Kapritsos, K. Rustan M. Leino, Jacob R. Lorch, Bryan Parno, Ashay Rane, Srinath T. V. Setty, and Laure Thompson. Vale: Verifying high-performance cryptographic assembly code. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017.*, pages 917–934. USENIX Association, 2017.

[8] David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[9] Claudio Canella, Daniel Genkin, Lukas Giner, Daniel Gruss, Moritz Lipp, Marina Minkin, Daniel Moghimi, Frank Piessens, Michael Schwarz, Berk Sunar, Jo Van Bulck, and Yuval Yarom. Fallout: Leaking Data on Meltdown-resistant CPUs. In *CCS*, 2019.

[10] Claudio Canella, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin Von Berg,Philipp Ortner, Frank Piessens, Dmitry Evtyushkin, and Daniel Gruss. A systematic evaluation of transient execution attacks and defenses. In *USENIX Security*, pages 249–266,2019.

[11] Chandler Carruth. Speculative load hardening, 2018. URL https://llvm.org/docs/ SpeculativeLoadHardening.html.

[12] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten- Hwang Lai. SgxPectre: Stealing Intel secrets from SGX enclaves via speculative execu- tion. In *Euro S&P*, pages 142–157, 2019.

[13] Dmitry Evtyushkin, Ryan Riley, Nael B. Abu-Ghazaleh, and Dmitry Ponomarev. BranchScope: A new side-channel attack on directional branch predictor. In *ASPLOS*, pages 693–707, 2018.

[14] Daniel Gruss, Cl´ementine Maurice, and Stefan Mangard. Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript. In *DIMVA*, 2016.

[15] Daniel Gruss, Cl´ementine Maurice, Klaus Wagner, and Stefan Mangard. Flush+Flush: A Fast and Stealthy Cache Attack. In *DIMVA*, 2016.

[16] Daniel Gruss, Moritz Lipp, Michael Schwarz, Richard Fellner, Cl´ementine Maurice, and Stefan Mangard. KASLR is dead: Long live KASLR. In *International Symposium on Engineering Secure Software and Systems*, pages 161–176, 2017.

[17] Daniel Gruss, Moritz Lipp, Michael Schwarz, Daniel Genkin, Jonas Juffinger, Sioli O'Connell, Wolfgang Schoechl, and Yuval Yarom. Another flip in the wall of rowhammer defenses. In *2018 IEEE Symposium on Security and Privacy, SP*, 2018.

[18] Jann Horn. Reading privileged memory with a side- channel. https://googleprojectzero.blogspot.com.au/2018/01/ reading-privileged-memory-with-side.html, 2018.

[19] Jann Horn. Speculative execution, variant 4: speculative store bypass. https://bugs. chromium.org/p/project-zero/issues/detail?id=1528, 2018.

[20] Jann Horn. Speculative execution, variant 4: Speculative store bypass. https://bugs. chromium.org/p/project-zero/issues/detail?id=1528, 2018.

[21] Ralf Hund, Carsten Willems, and Thorsten Holz. Practical timing side channel attacks against kernel space ASLR. In *NDSS*, 2013.

[22] Intel. Speculative execution side channel mitigations. https://software. intel.com/security-software-guidance/api-app/sites/default/files/ 336996-Speculative-Execution-Side-Channel-Mitigations.pdf, May 2018.

[23] Gorka Irazoqui Apecechea, Thomas Eisenbarth, and Berk Sunar. S$A: A shared cache attack that works across cores and defies VM sandboxing - and its application to AES. In *S&P*, pages 591–604, 2015.

[24] Saad Islam, Ahmad Moghimi, Ida Bruhns, Moritz Krebbel, Berk Gulmezoglu, Thomas Eisenbarth, and Berk Sunar. SPOILER: Speculative load hazards boost Rowhammer and cache attacks. In *USENIX Security*, pages 621–637, 2019.

[25] Yeongjin Jang, Jaehyuk Lee, Sangho Lee, and Taesoo Kim. SGX-Bomb: Locking down the processor via Rowhammer attack. In *SysTEX*, page 5, 2017.

[26] 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. In *International Symposium on Computer Architecture (ISCA)*, pages 361–372. IEEE Computer Society, 2014.

[27] Vladimir Kiriansky and Carl Waldspurger. Speculative buffer overflows: Attacks and defenses. *arXiv preprint arXiv:1807.03757*, 2018.

[28] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *S&P*, 2019.

[29] Esmaeil Mohammadian Koruyeh, Khaled N. Khasawneh, Chengyu Song, and Nael Abu-Ghazaleh. Spectre returns! speculation attacks using the return stack buffer. In *WOOT*, 2018.

[30] Adam Langley, Mike Hamburg, and Sean Turner. Elliptic Curves for Security. RFC 7748, January 2016. URL https://rfc-editor.org/rfc/rfc7748.txt.

[31] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading kernel memory from user space. In *USENIX Security Symposium*, 2018.

[32] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B. Lee. Last-level cache side-channel attacks are practical. In *S&P*, pages 605–622, 2015.

[33] Andrei Lutas and Dan Lutas. Security implications of speculatively executing segmenta- tion related instructions on Intel CPUs. https://businessresources.bitdefender. com/hubfs/noindex/Bitdefender-WhitePaper-INTEL-CPUs.pdf, Aug 2019.

[34] Giorgi Maisuradze and Christian Rossow. ret2spec: Speculative execution using return stack buffers. In *CCS*, pages 2109–2122, 2018.

[35] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of AES. In *CT-RSA*, pages 1–20, 2006.

[36] Andrew Pardoe. Spectre Mitigations in MSVC. https://blogs.msdn.microsoft.com/ vcblog/2018/01/15/spectre-mitigations-in-msvc/, Jan 2018.

[37] Colin Percival. Cache missing for fun and profit. In *Proceedings of BSDCan*, 2005. URL https://www.daemonology.net/papers/htt.pdf.

[38] Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. Flip feng shui: Hammering a needle in the software stack. In *USENIX Security Symposium*, 2016.

[39] Michael Schwarz, Moritz Lipp, Daniel Moghimi, Jo Van Bulck, Julian Stecklina, Thomas Prescher, and Daniel Gruss. ZombieLoad: Cross-privilege-boundary data sampling. In *CCS*, 2019.

[40] Mark Seaborn and Thomas Dullien. Exploiting the DRAM rowhammer bug to gain kernel privileges. In *Black Hat Briefings*, 2015.

[41] Paul Turner. Retpoline: a Software Construct for Preventing Branch Target Injection. https://support.google.com/faqs/answer/7625886, Jan 2018.

[42] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *USENIX Security Symposium*, pages 991–1008, 2018.

[43] Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yarom Yuval, Berk Sunar, Daniel Gruss, and Frank Piessens. LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection. In *41th IEEE Symposium on Security and Privacy (S&P'20)*, 2020.

[44] Victor van der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clemen- tine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. Drammer: Deterministic Rowhammer attacks on mobile platforms. In *CCS*, 2016.

[45] Stephan van Schaik, Marina Minkin, Andrew Kwong, Daniel Genkin, and Yuval Yarom. CacheOut: Leaking data on Intel CPUs via cache evictions. https://cacheoutattack. com/, 2020.

[46] Ofir Weisse, Jo Van Bulck, Marina Minkin, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Raoul Strackx, Thomas F. Wenisch, and Yuval Yarom. Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution. *Technical report*, 2018.

[47] David Woodhouse. x86/retpoline: Fill RSB on Context Switch for Affected CPUs. https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux. git/commit/?id=c995efd5a740d9cbafbf58bde4973e8b50b4d761, Jan 2018.

[48] Yuan Xiao, Xiaokuan Zhang, Yinqian Zhang, and Radu Teodorescu. One bit flips, one cloud flops: Cross-VM row hammer attacks and privilege escalation. In *USENIX Security*, 2016.

[49] Yuval Yarom. Mastik: A micro-architectural side-channel toolkit. http://cs. adelaide.edu.au/~yval/Mastik/Mastik.pdf, September 2016.

[50] Yuval Yarom and Katrina Falkner. Flush+Reload: A high resolution, low noise, L3 cache side-channel attack. In *USENIX Security Symposium*, pages 719–732, 2014.

[51] Yuval Yarom, Daniel Genkin, and Nadia Heninger. CacheBleed: a timing attack on OpenSSL constant-time RSA. *J. Cryptographic Engineering*, 7(2):99–112, 2017.

[52] Jean Karim Zinzindohou´e, Karthikeyan Bhargavan, Jonathan Protzenko, and Benjamin Beurdouche. Hacl*: A verified modern cryptographic library. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1789–1806. ACM, 2017.

LIST OF ACRONYMS

1. ACM --- Association for Computing Machinery
2. AES --- Advanced Encryption Standard
3. AMD --- Advanced Micro Devices
4. API --- Application Programming Interface
5. ARM --- Advanced Reduced Instruction Set Computer (RISC) Machines
6. ASLR --- Address Space Layout Randomization
7. ASPLOS --- International Conference on Architectural Support for Programming Languages and Operating Systems
8. CCS -- Conference on Computer and Communications Security
9. CEASER --- Cache operated on Encrypted Address-Space with Remapping
10. CHES --- Cryptographic Hardware and Embedded Systems
11. CPU – Central Processing Unit
12. CSS --- Cascading Style Sheets
13. DIMVA --- Conference on Detection of Intrusions and Malware & Vulnerability Assessment
14. DRAM --- Dynamic Random Access Memory
15. DRBG --- Deterministic Random Bit Generator
16. ECDSA --- Elliptic Curve Digital Signature Algorithm
17. EM --- Electromagnetic
18. FICHSA --- First Israeli Conference on Hardware and Side-Channel Attacks
19. FIPS --- Federal Information Processing Standard
20. GUI --- Graphical User Interface
21. IEEE --- Institute of Electrical and Electronics Engineers
22. KASLR --- Kernel Address Space Layout Randomization
23. LRU – Least Recently Used
24. LVI --- Load Value Injection
25. MDS --- Microarchitectural Data Sampling
26. MEMS --- Micro Electromechanical Systems
27. NDSS --- Network and Distributed System Security Symposium
28. NIST --- National Institute of Standards and Technology
29. OS --- Operating System
30. PI --- Principal Investigator
31. PLRU --- Pseudo LRU
32. PRG --- Pseudo Random Number Generators
33. RFC --- Request for Comments
34. RIDL --- Rogue In-Flight Data Load
35. RSA --- Rivest–Shamir–Adleman
36. SDK --- Software Development Kit
37. SDR --- Software Defined Radio
38. SGX --- Software Guard Extensions
39. SIGSAC --- Special Interest Group on Security, Audit and Control

40. SILM --- Security of Software / Hardware Interfaces
41. TLS --- Transport Layer Security
42. URL --- Uniform Resource Locator
43. USB --- Universal Serial Bus
44. VC --- Voice Command
45. VM --- Virtual Machine
46. WOOT --- Workshop on Offensive Technologies