Secure Processors Part II: Intel SGX Security Analysis and MIT Sanctum Architecture

Victor Costan, Ilia Lebedev and Srinivas Devadas
victor@costan.us, ilebedev@mit.edu and devadas@mit.edu
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Electronic Design Automation

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

V. Costan, I. Lebedev and S. Devadas. *Secure Processors Part II: Intel SGX Security Analysis and MIT Sanctum Architecture*. Foundations and Trends® in Electronic Design Automation, vol. 11, no. 3, pp. 249–361, 2017.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-302-7

Foundations and Trends® in Electronic Design Automation

Volume 11, Issue 3, 2017

Editorial Board

Full text available at: http://dx.doi.org/10.1561/1000000052

Editorial Scope

Topics

Foundations and Trends® in Electronic Design Automation publishes survey and tutorial articles in the following topics:

- System level design
- Behavioral synthesis
- Logic design
- Verification
- Test
- Physical design
- Circuit level design
- Reconfigurable systems
- Analog design
- Embedded software and parallel programming
- Multicore, GPU, FPGA, and heterogeneous systems
- Distributed, networked embedded systems
- Real-time and cyberphysical systems

Information for Librarians

Foundations and Trends® in Electronic Design Automation, 2017, Volume 11, 4 issues. ISSN paper version 1551-3939. ISSN online version 1551-3947. Also available as a combined paper and online subscription.

Foundations and Trends® in Electronic Design Automation
Vol. 11, No. 3 (2017) 249–361

DOI: 10.1561/1000000052

now
the essence of knowledge

Secure Processors Part II: Intel SGX Security Analysis and MIT Sanctum Architecture

Victor Costan, Ilia Lebedev and Srinivas Devadas
victor@costan.us, ilebedev@mit.edu and devadas@mit.edu
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Contents

Abstract

This manuscript is the second in a two part survey and analysis of the state of the art in secure processor systems, with a specific focus on remote software attestation and software isolation. The first part established the taxonomy and prerequisite concepts relevant to an examination of the state of the art in trusted remote computation: attested software isolation containers (enclaves). This second part extends Part I's description of Intel's Software Guard Extensions (SGX), an available and documented enclave-capable system, with a rigorous security analysis of SGX as a system for trusted remote computation. This part documents the authors' concerns over the shortcomings of SGX as a secure system and introduces the MIT Sanctum processor developed by the authors: a system designed to offer stronger security guarantees, lend itself better to analysis and formal verification, and offer a more straightforward and complete threat model than the Intel system, all with an equivalent programming model.

This two part work advocates a principled, transparent, and well-scrutinized approach to system design, and argues that practical guarantees of privacy and integrity for remote computation are achievable at a reasonable design cost and performance overhead.

V. Costan, I. Lebedev and S. Devadas. *Secure Processors Part II: Intel SGX Security Analysis and MIT Sanctum Architecture*. Foundations and Trends® in Electronic Design Automation, vol. 11, no. 3, pp. 249–361, 2017.
DOI: 10.1561/1000000052.

1

Introduction

Between the Snowden revelations and the seemingly unending series of high-profile hacks of the past few years, the public's confidence in software systems has decreased considerably. At the same time, key initiatives such as cloud computing and the IoT (Internet of Things) are gaining popularity but require users to place much trust in the systems providing these services. We must therefore develop capabilities to build software systems with compelling security, and gain back our users' trust.

This manuscript is the second in a two part survey of the state of the art in secure processor systems, with a specific focus on remote software attestation and software isolation. Part I [Costan et al., 2017] established relevant background in computer system design (§ I.2) and security primitives (§ I.3), and surveyed relevant prior work (§ I.4). The same work discussed the attested software isolation container (enclave): a modern primitive for modular secure software and trusted remote computation, as exemplified by Intel's Software Guard Extensions (§ I.5).

This manuscript extends the discussion of enclaves and SGX by surveying the implementation and security properties of SGX (§ 2),

and documents the authors' concerns with its vulnerabilities to several classes of software attacks. Informed by the successes and shortcomings of SGX, this manuscript also discusses the MIT Sanctum processor (§ 3): a secure processor that offers an equivalent programming model with strong security guarantees against an insidious software threat model including cache timing and memory access pattern attacks. With this work, we hope to enable a shift in discourse in secure hardware architecture away from plugging specific security holes to a principled approach to eliminating attack surfaces.

1.1 The Case for Hardware Isolation

The best known practical method for securing a software system amounts to modularizing the system's code in a way that minimizes code in the modules responsible for the system's security. Formal verification techniques are then applied to these modules, which make up the system's trusted codebase (TCB). The method assumes that software modules are isolated, so the TCB must also include the mechanism providing the isolation guarantees.

Today's systems rely on an operating system kernel, or a hypervisor (such as Linux or Xen, respectively) for software isolation. However **each** of the last three years (2012-2014) witnessed over 100 new security vulnerabilities in Linux [cve, 2014a, Chen et al., 2011], and over 40 in Xen [cve, 2014b].

One may hope that formal verification methods can produce a secure kernel or hypervisor. Unfortunately, these codebases are far outside our verification capabilities: Linux and Xen have over *17 million* [Anthony, 2014] and 150,000 [xen, 2015] lines of code, respectively. In stark contrast, the seL4 formal verification effort [Klein et al., 2009] spent *20 man-years* to cover 9,000 lines of code.

Given Linux and Xen's history of vulnerabilities and uncertain prospects for formal verification, a prudent system designer cannot include either in a TCB (trusted computing base), and must look elsewhere for a software isolation mechanism.

Fortunately, Intel's Software Guard Extensions (SGX) [McKeen et al., 2013, Anati et al., 2013] has brought attention to the alternative of providing software isolation primitives in the CPU's hardware. This avenue is appealing because the CPU is an unavoidable TCB component, and processor manufacturers have strong economic incentives to build correct hardware.

1.2 Intel SGX is Not the Answer

Unfortunately, although the SGX design includes a vast array of defenses against a variety of software and physical attacks, it fails to offer meaningful software isolation guarantees. The SGX threat model protects against all direct attacks, but excludes "side-channel attacks", even if they can be performed via software alone.

Alarmingly, cache timing attacks require only unprivileged software running on the victim's host computer, and do not rely on any physical access to the machine. This is particularly concerning in a cloud computing scenario, where gaining software access to the victim's computer only requires a credit card [Ristenpart et al., 2009], whereas physical access is harder, requiring trespass, coercion, or social engineering on the cloud provider's employees.

Similarly, in many Internet of Things (IoT) scenarios, the processing units have some amount of physical security, but they run outdated software stacks that have known security vulnerabilities. For example, an attacker may exploit a vulnerability in an IoT lock's Bluetooth stack and obtain software execution privileges, then mount a cache timing attack on its access-granting process, and obtain the cryptographic key that opens the lock.

Furthermore, the analysis of SGX documentation as described in Part I of this work reveals that it is impossible for anyone but Intel to reason about SGX's security properties, because significant implementation details are not covered by the publicly available documentation. This is a concern, as the myriad of security vulnerabilities [Wojtczuk and Rutkowska, 2011, 2009b, Wojtczuk et al., 2009, Duflot et al., 2006, Rutkowska and Wojtczuk, 2008, Wojtczuk and Rutkowska,

2009a, Wecherowski, 2009, Embleton et al., 2010] in TXT [Grawrock, 2009], Intel's previous attempt at securing remote computation, show that securing the machinery underlying Intel's processors is incredibly challenging, even in the presence of strong economic incentives.

If a successor to SGX claimed to protect against cache timing attacks, substantiating such a claim would require an analysis of its hardware and microcode, and ensuring that no implementation detail is vulnerable to cache timing attacks. Barring a highly unlikely shift to open-source hardware from Intel, such analysis will never happen.

A concrete example: the SGX documentation [Int, 2013, 2014] does not state where SGX stores the EPCM (enclave page cache map). If the EPCM is stored in cacheable RAM, page translation verification is subject to cache timing attacks. Interestingly, this detail is unnecessary for analyzing the security of today's SGX implementation, as we know that SGX uses the operating system's page tables, and page translations are therefore vulnerable to cache timing attacks. The example does, however, demonstrate the fine nature of crucial details that are simply undocumented in today's hardware security implementations.

In summary, while the principles behind SGX have great potential, the SGX design does not offer meaningful isolation guarantees, and the SGX implementation is not open enough for independent researchers to be able to analyze its security properties.

1.3 MIT Sanctum Processor

The Sanctum processor's main contribution is a software isolation scheme that addresses the issues raised above: Sanctum's isolation provably defends against known software side-channel attacks, including cache timing attacks and passive address translation attacks. Sanctum is a co-design that combines *minimal* and *minimally invasive* hardware modifications with a trusted software security monitor that is amenable to rigorous analysis and does not perform cryptographic operations using keys.

Sanctum achieves minimality by reusing and lightly modifying existing, well-understood mechanisms. For example, Sanctum's per-

enclave page tables implementation uses the core's existing page walking circuit, and requires very little extra logic. Sanctum is minimally invasive because it does not require modifying any major CPU building block. It only adds hardware to the interfaces between blocks, and does not modify any block's input or output. The use of conventional building blocks limits the effort needed to validate a Sanctum implementation.

Sanctum demonstrates that memory access pattern attacks by malicious software can be foiled without incurring unreasonable overheads. Its hardware changes are small, small enough to present the added circuits, in their entirety, in Figures 3.9 and 3.10. Sanctum cores have the same clock speed as their insecure counterparts, as there are no modifications on the CPU core critical execution path. Using a straightforward page-coloring-based cache partitioning scheme with Sanctum adds a few percent of overhead in execution time, which is orders of magnitude lower than the overheads of the ORAM schemes [Goldreich, 1987, Stefanov et al., 2013] that are usually employed to conceal memory access patterns.

All layers of Sanctum's TCB are open-sourced [MIT, 2017], and unencumbered by patents, trade secrets, or other similar intellectual property concerns that would disincentivize security researchers from analyzing it. The Sanctum prototype targets the Rocket Chip [Lee et al., 2014], an open-sourced implementation of the RISC-V [Waterman et al., 2014, 2015] instruction set architecture, which is an open standard. Sanctum's software stack bears the MIT license.

To further encourage analysis, most of Sanctum's security monitor is written in portable C++ which, once rigorously analyzed, can be used across different CPU implementations. Furthermore, even the non-portable assembly code can be reused across different implementations of the same architecture. In comparison, SGX's microcode is CPU model-specific, so each micro-architectural revision would require a separate verification effort.

References

Linux kernel: CVE security vulnerabilities, versions and detailed reports. http://www.cvedetails.com/product/47/Linux-Linux-Kernel.html?vendor_id=33, 2014a. [Online; accessed 27-April-2015].

XEN: CVE security vulnerabilities, versions and detailed reports. http://www.cvedetails.com/product/23463/XEN-XEN.html?vendor_id=6276, 2014b. [Online; accessed 27-April-2015].

Xen project software overview. http://wiki.xen.org/wiki/Xen_Project_Software_Overview, 2015. [Online; accessed 27-April-2015].

Ittai Anati, Shay Gueron, Simon P. Johnson, and Vincent R. Scarlata. Innovative technology for CPU based attestation and sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, HASP*, volume 13, 2013.

Sebastian Anthony. Who actually develops Linux? the answer might surprise you. http://www.extremetech.com/computing/175919-who-actually-develops-linux, 2014. [Online; accessed 27-April-2015].

Gorka Irazoqui Apecechea, Mehmet Sinan Inci, Thomas Eisenbarth, and Berk Sunar. Fine grain cross-VM attacks on Xen and VMware are possible! Cryptology ePrint Archive, Report 2014/248, 2014. http://eprint.iacr.org/.

Sebastian Banescu. Cache timing attacks. 2011. [Online; accessed 26-January-2014].

Joseph Bonneau and Ilya Mironov. Cache-collision timing attacks against AES. In *Cryptographic Hardware and Embedded Systems-CHES 2006*, pages 201–215. Springer, 2006.

E. Brickell and J. Li. Hardening inter-device secure communication using physically unclonable functions, 2014. US Patent App. 13/844,559.

Ernie Brickell and Jiangtao Li. Enhanced privacy ID from bilinear pairing. *IACR Cryptology ePrint Archive*, 2009.

Billy Bob Brumley and Nicola Tuveri. Remote timing attacks are still practical. In *Computer Security–ESORICS 2011*, pages 355–371. Springer, 2011.

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

J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. In *Proceedings of the 9th annual ACM Symposium on Theory of Computing*, pages 106–112. ACM, 1977.

Haogang Chen, Yandong Mao, Xi Wang, Dong Zhou, Nickolai Zeldovich, and M. Frans Kaashoek. Linux kernel vulnerabilities: State-of-the-art defenses and open problems. In *Proceedings of the Second Asia-Pacific Workshop on Systems*, page 5. ACM, 2011.

Victor Costan and Srinivas Devadas. Security challenges and opportunities in adaptive and reconfigurable hardware. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 1–5. IEEE, 2011.

Victor Costan and Srinivas Devadas. Intel SGX explained. Cryptology ePrint Archive, Report 2016/086, Feb 2016.

Victor Costan, Ilia Lebedev, and Srinivas Devadas. Secure processors part I: Background, taxonomy for secure enclaves and Intel SGX architecture. In *FnTEDA*, 2017.

Shaun Davenport. SGX: the good, the bad and the downright ugly. *Virus Bulletin*, 2014.

Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

Leonid Domnitser, Aamer Jaleel, Jason Loew, Nael Abu-Ghazaleh, and Dmitry Ponomarev. Non-monopolizable caches: Low-complexity mitigation of cache side channel attacks. *ACM Transactions on Architecture and Code Optimization (TACO)*, 8(4):35, 2012.

Loïc Duflot, Daniel Etiemble, and Olivier Grumelard. Using CPU system management mode to circumvent operating system security functions. *CanSecWest/core06*, 2006.

Alan Dunn, Owen Hofmann, Brent Waters, and Emmett Witchel. Cloaking malware with the trusted platform module. In *USENIX Security Symposium*, 2011.

Shawn Embleton, Sherri Sparks, and Cliff C. Zou. SMM rootkit: a new breed of OS independent malware. *Security and Communication Networks*, 2010.

Dmitry Evtyushkin, Jesse Elwell, Meltem Ozsoy, Dmitry Ponomarev, Nael Abu Ghazaleh, and Ryan Riley. Iso-X: A flexible architecture for hardware-managed isolated execution. In *Microarchitecture (MICRO), 2014 47th annual IEEE/ACM International Symposium on*, pages 190–202. IEEE, 2014.

Christopher W. Fletcher, Marten van Dijk, and Srinivas Devadas. A secure processor architecture for encrypted computation on untrusted programs. In *Proceedings of the Seventh ACM Workshop on Scalable Trusted Computing*, pages 3–8. ACM, 2012.

Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 148–160. ACM, 2002.

Oded Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *Proceedings of the 19th annual ACM symposium on Theory of Computing*, pages 182–194. ACM, 1987.

K. C. Gotze, G. M. Iovino, and J. Li. Secure provisioning of secret keys during integrated circuit manufacturing, 2014a. US Patent App. 13/631,512.

K. C. Gotze, J. Li, and G. M. Iovino. Fuse attestation to secure the provisioning of secret keys during integrated circuit manufacturing, 2014b. US Patent 8,885,819.

David Grawrock. *Dynamics of a Trusted Platform: A building block approach.* Intel Press, 2009.

Shay Gueron. Quick verification of RSA signatures. In *8th International Conference on Information Technology: New Generations (ITNG)*, pages 382–386. IEEE, 2011.

Shay Gueron. A memory encryption engine suitable for general purpose processors. Cryptology ePrint Archive, Report 2016/204, 2016.

Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Vinay Phegade, and Juan Del Cuvillo. Using innovative instructions to create trustworthy software solutions. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, HASP*, volume 13, 2013.

Mehmet Sinan Inci, Berk Gulmezoglu, Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar. Seriously, get off my cloud! cross-VM RSA key recovery in a public cloud. Cryptology ePrint Archive, Report 2015/898, 2015.

Software Guard Extensions Programming Reference. Intel Corporation, 2013. Reference no. 329298-001US.

Software Guard Extensions Programming Reference. Intel Corporation, 2014. Reference no. 329298-002US.

Intel® Software Guard Extensions (Intel® SGX). Intel Corporation, Jun 2015a. Reference no. 332680-002.

Intel® 64 and IA-32 Architectures Software Developer's Manual. Intel Corporation, Sep 2015b. Reference no. 325462-056US.

Simon Johnson, Vinnie Scarlata, Carlos Rozas, Ernie Brickell, and Frank Mckeen. Intel® software guard extensions: EPID provisioning and attestation services. `https://software.intel.com/en-us/blogs/2016/03/09/intel-sgx-epid-provisioning-and-attestation-services`, Mar 2016. [Online; accessed 21-Mar-2016].

Simon P. Johnson, Uday R. Savagaonkar, Vincent R. Scarlata, Francis X. McKeen, and Carlos V. Rozas. Technique for supporting multiple secure enclaves, Dec 2010. US Patent 8,972,746.

Richard E. Kessler and Mark D. Hill. Page placement algorithms for large real-indexed caches. *ACM Transactions on Computer Systems (TOCS)*, 10 (4):338–359, 1992.

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 *Proceeding of the 41st annual International Symposium on Computer Architecuture*, pages 361–372. IEEE Press, 2014.

Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, et al. seL4: Formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 207–220. ACM, 2009.

Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology–CRYPTO'96*, pages 104–113. Springer, 1996.

Jingfei Kong, Onur Aciicmez, Jean-Pierre Seifert, and Huiyang Zhou. Deconstructing new cache designs for thwarting software cache-based side channel attacks. In *Proceedings of the 2nd ACM workshop on Computer security architectures*, pages 25–34. ACM, 2008.

Tsvika Kurts, Guillermo Savransky, Jason Ratner, Eilon Hazan, Daniel Skaba, Sharon Elmosnino, and Geeyarpuram N. Santhanakrishnan. Generic debug eXternal connection (GDXC) for high integration integrated circuits, 2011. US Patent 8,074,131.

Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. Inferring fine-grained control flow inside SGX enclaves with branch shadowing. *CoRR*, abs/1611.06952, 2016. URL `http://arxiv.org/abs/1611.06952`.

Yunsup Lee, Andrew Waterman, Rimas Avizienis, Henry Cook, Chen Sun, Vladimir Stojanovic, and Krste Asanovic. A 45nm 1.3 GHz 16.7 double-precision GFLOPS/w RISC-V processor with vector accelerators. In *European Solid State Circuits Conference (ESSCIRC), ESSCIRC 2014-40th*, pages 199–202. IEEE, 2014.

Jiang Lin, Qingda Lu, Xiaoning Ding, Zhao Zhang, Xiaodong Zhang, and P. Sadayappan. Gaining insights into multicore cache partitioning: Bridging the gap between simulation and real systems. In *14th International IEEE Symposium on High Performance Computer Architecture (HPCA)*, pages 367–378. IEEE, 2008.

Fangfei Liu and Ruby B. Lee. Random fill cache architecture. In *Microarchitecture (MICRO), 2014 47th annual IEEE/ACM International Symposium on*, pages 203–215. IEEE, 2014.

Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B. Lee. Last-level cache side-channel attacks are practical. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 143–158. IEEE, 2015.

Fangfei Liu, Qian Ge, Yuval Yarom, Frank Mckeen, Carlos Rozas, Gernot Heiser, and Ruby B. Lee. CATalyst: Defeating last-level cache side channel attacks in cloud computing. In *HPCA*, Mar 2016.

R. Maes, P. Tuyls, and I. Verbauwhede. Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 332–347, 2009.

Clémentine Maurice, Nicolas Le Scouarnec, Christoph Neumann, Olivier Heen, and Aurélien Francillon. Reverse engineering Intel last-level cache complex addressing using performance counters. In *Proceedings of the 18th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2015.

Francis X. McKeen, Carlos V. Rozas, Uday R. Savagaonkar, Simon P. Johnson, Vincent Scarlata, Michael A. Goldsmith, Ernie Brickell, Jiang Tao Li, Howard C. Herbert, Prashant Dewan, Stephen J. Tolopka, Gilbert Neiger, David Durham, Gary Graunke, Bernard Lint, Don A. Van Dyke, Joseph Cihula, Stalinselvaraj Jeyasingh, Stephen R. Van Doren, Dion Rodgers, John Garney, and Asher Altman. Method and apparatus to provide secure application execution, Dec 2009. US Patent 9,087,200.

Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. Innovative instructions and software model for isolated execution. *HASP*, 13:10, 2013.

MIT. Reference implementation of a Sanctum security monitor. `https://github.com/pwnall/sanctum`, 2017. [Online; accessed 1-Jan-2017].

Yossef Oren, Vasileios P. Kemerlis, Simha Sethumadhavan, and Angelos D. Keromytis. The spy in the sandbox – practical cache attacks in JavaScript. *arXiv preprint arXiv:1502.07373*, 2015.

Peter Pessl, Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Reverse engineering intel DRAM addressing and exploitation. *CoRR*, abs/1511.08756, 2015.

Stefan M. Petters and Georg Farber. Making worst case execution time analysis for hard real-time tasks on state of the art processors feasible. In *Sixth International Conference on Real-Time Computing Systems and Applications*, pages 442–449. IEEE, 1999.

Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 199–212. ACM, 2009.

Xiaoyu Ruan. *Platform Embedded Security Technology Revealed*. Apress, 2014. ISBN 978-1-4302-6571-9.

Joanna Rutkowska. Thoughts on Intel's upcoming software guard extensions (part 2). *Invisible Things Lab*, 2013.

Joanna Rutkowska and Rafał Wojtczuk. Preventing and detecting Xen hypervisor subversions. *Blackhat Briefings USA*, 2008.

Daniel Sanchez and Christos Kozyrakis. The ZCache: Decoupling ways and associativity. In *Microarchitecture (MICRO), 2010 43rd annual IEEE/ACM International Symposium on*, pages 187–198. IEEE, 2010.

Daniel Sanchez and Christos Kozyrakis. Vantage: scalable and efficient fine-grain cache partitioning. In *ACM SIGARCH Computer Architecture News*, volume 39, pages 57–68. ACM, 2011.

Mark Seaborn and Thomas Dullien. Exploiting the DRAM rowhammer bug to gain kernel privileges. `http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html`, Mar 2015. [Online; accessed 9-March-2015].

V. Shanbhogue, J. W. Brandt, and J. Wiedemeier. Protecting information processing system secrets from debug attacks, 2015. US Patent 8,955,144.

Emil Stefanov, Marten Van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: An extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 299–310. ACM, 2013.

G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*, pages 9–14. ACM, 2007.

G. Edward Suh, Dwaine Clarke, Blaise Gassend, Marten Van Dijk, and Srinivas Devadas. AEGIS: architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th annual international conference on Supercomputing*, pages 160–171. ACM, 2003.

G. Edward Suh, Charles W. O'Donnell, Ishan Sachdev, and Srinivas Devadas. Design and Implementation of the AEGIS Single-Chip Secure Processor Using Physical Random Functions. In *Proceedings of the* 32^{nd} *ISCA'05*. ACM, June 2005.

George Taylor, Peter Davies, and Michael Farmwald. The TLB slice - a low-cost high-speed address translation mechanism. *SIGARCH Computer Architecture News*, 18(2SI):355–363, 1990.

Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, XiaoFeng Wang, Vincent Bindschaedler, Haixu Tang, and Carl A. Gunter. Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX. *CoRR*, abs/1705.07289, 2017.

Zhenghong Wang and Ruby B. Lee. New cache designs for thwarting software cache-based side channel attacks. In *Proceedings of the 34th annual International Symposium on Computer Architecture*, ISCA '07, pages 494–505, 2007. ISBN 978-1-59593-706-3.

Andrew Waterman, Yunsup Lee, David A. Patterson, and Krste Asanovic. The RISC-V instruction set manual, volume I: User-level ISA, version 2.0. Technical Report UCB/EECS-2014-54, EECS Department, University of California, Berkeley, May 2014. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.html`.

Andrew Waterman, Yunsup Lee, Rimas Avizienis, David A. Patterson, and Krste Asanovic. The RISC-V instruction set manual volume II: Privileged architecture version 1.7. Technical Report UCB/EECS-2015-49, EECS Department, University of California, Berkeley, May 2015. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-49.html`.

Filip Wecherowski. A real SMM rootkit: Reversing and hooking BIOS SMI handlers. *Phrack Magazine*, 13(66), 2009.

Mark N. Wegman and J. Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.

Rafal Wojtczuk and Joanna Rutkowska. Attacking SMM memory via Intel CPU cache poisoning. *Invisible Things Lab*, 2009a.

Rafal Wojtczuk and Joanna Rutkowska. Attacking Intel trusted execution technology. *Black Hat DC*, 2009b.

Rafal Wojtczuk and Joanna Rutkowska. Attacking intel txt via sinit code execution hijacking, 2011.

Rafal Wojtczuk, Joanna Rutkowska, and Alexander Tereshkin. Another way to circumvent Intel® trusted execution technology. *Invisible Things Lab*, 2009.

Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *Proceedings of the 36th IEEE Symposium on Security and Privacy (Oakland)*. IEEE – Institute of Electrical and Electronics Engineers, May 2015.

Yuval Yarom and Katrina E. Falkner. Flush+Reload: a high resolution, low noise, L3 cache side-channel attack. *IACR Cryptology ePrint Archive*, 2013: 448, 2013.

Yuval Yarom, Qian Ge, Fangfei Liu, Ruby B. Lee, and Gernot Heiser. Mapping the Intel last-level cache. Cryptology ePrint Archive, Report 2015/905, 2015.

Bennet Yee, David Sehr, Gregory Dardyk, J. Bradley Chen, Robert Muth, Tavis Ormandy, Shiki Okasaka, Neha Narula, and Nicholas Fullagar. Native client: A sandbox for portable, untrusted x86 native code. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 79–93. IEEE, 2009.

Marcelo Yuffe, Ernest Knoll, Moty Mehalel, Joseph Shor, and Tsvika Kurts. A fully integrated multi-CPU, GPU and memory controller 32nm processor. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pages 264–266. IEEE, 2011.