Full text available at: http://dx.doi.org/10.1561/1000000051

Secure Processors Part I: Background, Taxonomy for Secure Enclaves and Intel SGX 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

Full text available at: http://dx.doi.org/10.1561/1000000051

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 I: Background, Taxonomy for Secure Enclaves and Intel SGX Architecture.* Foundations and Trends® in Electronic Design Automation, vol. 11, no. 1-2, pp. 1–248, 2017.

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

ISBN: 978-1-68083-300-3
© 2017 V. Costan, I. Lebedev, and S. Devadas

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording or otherwise, without prior written permission of the publishers.

Photocopying. In the USA: This journal is registered at the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923. Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, is granted by now Publishers Inc for users registered with the Copyright Clearance Center (CCC). The 'services' for users can be found on the internet at: www.copyright.com

For those organizations that have been granted a photocopy license, a separate system of payment has been arranged. Authorization does not extend to other kinds of copying, such as that for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. In the rest of the world: Permission to photocopy must be obtained from the copyright owner. Please apply to now Publishers Inc., PO Box 1024, Hanover, MA 02339, USA; Tel. +1 781 871 0245; www.nowpublishers.com; sales@nowpublishers.com

now Publishers Inc. has an exclusive license to publish this material worldwide. Permission to use this content must be obtained from the copyright license holder. Please apply to now Publishers, PO Box 179, 2600 AD Delft, The Netherlands, www.nowpublishers.com; e-mail: sales@nowpublishers.com

Full text available at: http://dx.doi.org/10.1561/1000000051

Foundations and Trends® in Electronic Design Automation
Volume 11, Issue 1-2, 2017

Editorial Board

Editor-in-Chief

Radu Marculescu
Carnegie Mellon University
United States

Editors

Robert K. Brayton
UC Berkeley

Raul Camposano
Nimbic

K.T. Tim Cheng
UC Santa Barbara

Jason Cong
UCLA

Masahiro Fujita
University of Tokyo

Georges Gielen
KU Leuven

Tom Henzinger
Institute of Science and Technology Austria

Andrew Kahng
UC San Diego

Andreas Kuehlmann
Coverity

Sharad Malik
Princeton University

Ralph Otten
TU Eindhoven

Joel Phillips
Cadence Berkeley Labs

Jonathan Rose
University of Toronto

Rob Rutenbar
University of Illinois at Urbana-Champaign

Alberto Sangiovanni-Vincentelli
UC Berkeley

Leon Stok
IBM Research

Full text available at: http://dx.doi.org/10.1561/1000000051

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.

Full text available at: http://dx.doi.org/10.1561/1000000051

Foundations and Trends® in Electronic Design Automation
Vol. 11, No. 1-2 (2017) 1–248
© 2017 V. Costan, I. Lebedev, and S. Devadas
DOI: 10.1561/1000000051

now
the essence of knowledge

Secure Processors Part I: Background, Taxonomy for Secure Enclaves and Intel SGX 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

Full text available at: http://dx.doi.org/10.1561/1000000051

Contents

Full text available at: http://dx.doi.org/10.1561/1000000051

Full text available at: http://dx.doi.org/10.1561/1000000051

Full text available at: http://dx.doi.org/10.1561/1000000051

Abstract

This manuscript is the first 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. This manuscript first examines the relevant concepts in computer architecture and cryptography, and then surveys attack vectors and existing processor systems claiming security for remote computation and/or software isolation. This work examines in detail the modern isolation container (enclave) primitive as a means to minimize trusted software given practical trusted hardware and reasonable performance overhead. Specifically, this work examines in detail the programming model and software design considerations of Intel's Software Guard Extensions (SGX), as it is an available and documented enclave-capable system.

Part II of this work is a deep dive into the implementation and security evaluation of two modern enclave-capable secure processor systems: SGX and MIT's Sanctum. The complex but insufficient threat model employed by SGX motivates Sanctum, which achieves stronger security guarantees under software attacks with an equivalent programming model.

This work advocates a principled, transparent, and well-scrutinized approach to secure 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 I: Background, Taxonomy for Secure Enclaves and Intel SGX Architecture.* Foundations and Trends® in Electronic Design Automation, vol. 11, no. 1-2, pp. 1–248, 2017.
DOI: 10.1561/1000000051.

Full text available at: http://dx.doi.org/10.1561/1000000051

1

Introduction

A user wishing to perform computation remotely faces a complex trade-off: how much trust can be placed in the remote system? How much of a performance overhead is considered acceptable for the given security properties? How strong an adversary can the remote system defend against? An ideal system would offer overhead-free trustworthy private remote computation with no assumptions of trust at all, yet no such system exists.

At one extreme, expensive cryptographic techniques including garbled circuits [Yao, 1986] and fully homomorphic encryption [Gentry, 2009] offer trust-free computation at prohibitive cost. A typical cloud computing scenario lies much closer to the opposite extreme: weak security guarantees achievable with minimal overhead assuming nearly unchecked trust in the remote system. This work aims to illustrate that significant security properties can be achieved given very modest trust in the remote system. A long lineage of secure processors explore the space of trusted hardware enabling inexpensive remote computation robust against a variety of threat models.

A rigorous conversation about security requires a precisely stated thread model: trusted hardware must be secure, meaning it must show

Full text available at: http://dx.doi.org/10.1561/1000000051

resilience against a well-specified threat model. For example, few systems can offer meaningful guarantees against an adversary capable of physically tampering with the system's hardware. While the space of projects fitting the description of "secure processor" is large indeed, this work focuses on systems enabling *secure remote computation*, defined in § 1.1. Specifically, this work aims to illuminate the programming model, historical context, design decisions, and threat models relevant to secure software enclaves – the latest and so far the most capable paradigm for secure remote computation. We survey Intel's Software Guard Extensions (SGX) and MIT's Sanctum systems to exemplify enclave-capable systems.

This work is presented in two parts, the first covering the technical background and taxonomy of computer architecture (§ 2) and security concepts (§ 3) as relevant to an in-depth discussion of secure processors. This same part presents a survey of prior work (§ 4) and an in-depth discussion of the programming model presented by secure software enclaves, as exemplified by Intel's Software Guard Extensions (§ 5).

Part II [Costan et al., 2017] of this review is a deep dive into the implementation and security properties of two modern enclave-capable secure processor systems: SGX and MIT's Sanctum. This work aims to rigorously analyze the security properties and trade-offs employed buy the secure properties to achieve their stated goals.

1.1 Secure Remote Computation

Secure remote computation (Figure 1.1) is the problem of executing software on a remote computer **owned and maintained by an untrusted party**, with some integrity and confidentiality guarantees. In the general setting, secure remote computation is an unsolved problem. Fully Homomorphic Encryption [Gentry, 2009] addresses the problem for a limited family of computations, but has an impractical performance overhead [Naehrig et al., 2011].

Intel's Software Guard Extensions (SGX) is the latest iteration in a long line of trusted computing (Figure 1.2) designs, which aim to solve the secure remote computation problem by leveraging trusted

Full text available at: http://dx.doi.org/10.1561/1000000051

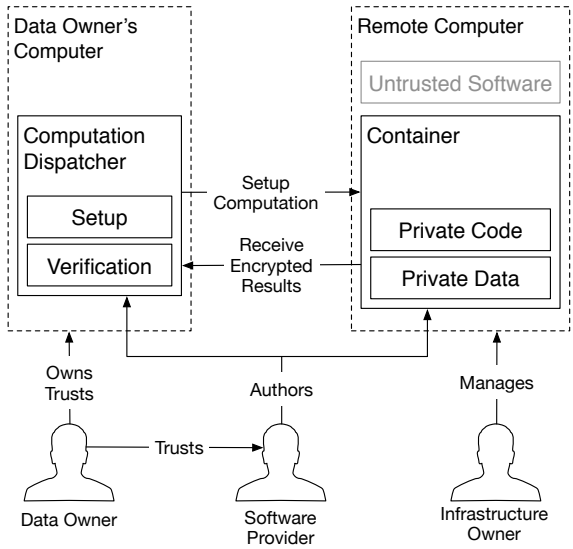

Figure 1.1: Secure remote computation. A user relies on a remote computer, owned by an untrusted party, to perform some computation on her data. The user has some assurance of the computation's integrity and confidentiality.

hardware in the remote computer. The trusted hardware establishes a secure container, and the remote computation service user uploads the desired computation and data into the secure container. The trusted hardware protects the confidentiality and integrity of data while the computation is being performed on it.

SGX, Sanctum, and similar work rely on *software attestation*, like their predecessors, the TPM [TCG, 2003] and TXT [Grawrock, 2009]. Attestation (Figure 1.3) proves to a user that she is communicating with a specific piece of software running in a secure container hosted by the trusted hardware. The proof is a cryptographic signature that certifies the hash of the secure container's contents. It follows that the remote computer's owner can load any software in a secure container, but the remote computation service user is able to refuse to send private data to a secure container with a hash that does not match an expected value.

The remote computation service user verifies the *attestation key* used to produce the signature against an *endorsement certificate* cre-

Full text available at: http://dx.doi.org/10.1561/1000000051

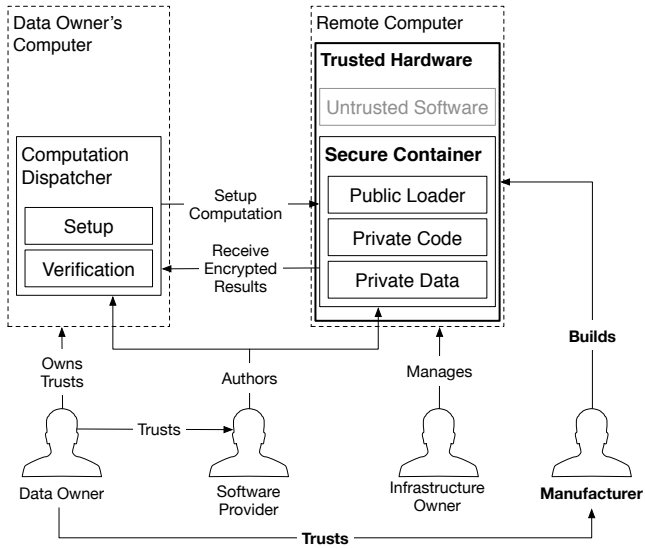

Figure 1.2: Trusted computing. The user trusts the manufacturer of a piece of hardware in the remote computer, and entrusts her data to a secure container hosted by the secure hardware.

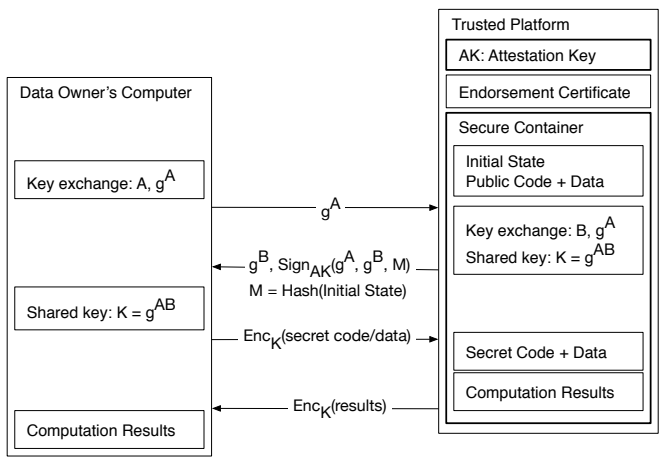

Figure 1.3: Software attestation proves to a remote computer that it is communicating with a specific secure container hosted by a trusted platform. The proof is an attestation signature produced by the platform's secret attestation key. The signature covers the container's initial state, a challenge nonce produced by the remote computer, and a message produced by the container.

Full text available at: http://dx.doi.org/10.1561/1000000051

ated by the trusted hardware's manufacturer. The certificate states that the attestation key is only known to the trusted hardware, and only used for the purpose of attestation.

SGX stands out from its predecessors by the amount of code covered by the attestation, which is in the Trusted Computing Base (TCB) for the system using hardware protection. The attestations produced by the original TPM design covered the whole of the software running on a computer, and TXT attestations covered the code inside a VMX [Uhlig et al., 2005] virtual machine. In SGX, an *enclave* (secure container) only contains the private data in a computation, and the code that operates on it.

For example, a cloud service that performs image processing on confidential medical images could be implemented by having users upload encrypted images. The users would send the encryption keys to software running inside an enclave. The enclave would contain the code for decrypting images, the image processing algorithm, and the code for encrypting the results. The code that receives the uploaded encrypted images and stores them would be left outside the enclave. This example is illustrated in Figure 1.4.

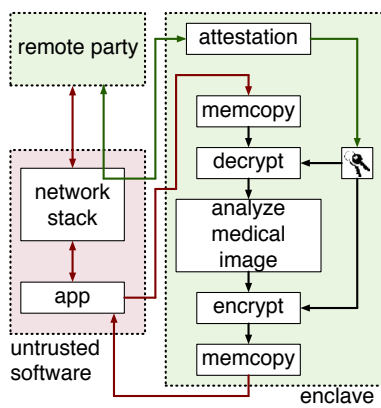

Figure 1.4: An example software application that uses SGX to implement a private function analyzing a medical image.

An SGX-enabled processor protects the integrity and confidentiality of the computation inside an enclave by isolating the enclave's code

Full text available at: http://dx.doi.org/10.1561/1000000051

and data from other software, including the operating system and hypervisor, and hardware devices attached to the system bus. At the same time, the SGX model remains compatible with the traditional software layering in the Intel architecture, where the OS kernel and hypervisor manage the computer's resources.

This work discusses the original version of SGX, also referred to as SGX 1. While SGX 2 brings very useful improvements for enclave authors, it is a small incremental improvement, from a design and implementation standpoint. After understanding the principles behind SGX 1 and its security properties, the reader should be well equipped to face Intel's reference documentation and learn about the changes brought by SGX 2 and more recent work.

1.2 SGX Lightning Tour

While this manuscript seeks to educate the reader of the challenges, history, and state of the art in secure processors for remote computation, this discussion is grounded in the example of Intel's Software Guard Extensions (SGX), as it is an available, documented, and modern system that aims to offer useful security guarantees to remotely executed programs. This section presents a brief overview of the SGX platform, directing the reader to other sections of the manuscript for a deeper look at each aspect of SGX.

SGX sets aside a memory region, called the *Processor Reserved Memory* (PRM, § 5.1). The CPU protects the PRM from all non-enclave memory accesses, including kernel, hypervisor and management engine (SMM, § 2.3) accesses, and DMA accesses (§ 2.9.1) from peripherals.

The PRM holds the *Enclave Page Cache* (EPC, § 5.1.1), which consists of 4 KB pages that store enclave code and data. The system software, which is untrusted, is in charge of assigning EPC pages to enclaves. The CPU tracks each EPC page's state in the *Enclave Page Cache Metadata* (EPCM, § 5.1.2), to ensure that each EPC page is assigned exclusively, belonging to exactly one enclave.

Full text available at: http://dx.doi.org/10.1561/1000000051

The initial code and data in an enclave is loaded by untrusted system software. During loading (§ 5.3), system software asks the CPU to copy data from unprotected memory (outside PRM) into EPC pages, and assigns the pages to the enclave being setup (§ 5.1.2). It follows that the initial enclave state is known to the system software.

After the enclave's pages are loaded into EPC, the system software asks the CPU to mark the enclave as initialized (§ 5.3), at which point application software may execute code inside the enclave. After an enclave is initialized, the loading mechanism briefly described above is no longer available to system software.

While an enclave is loaded, its contents and configuration are cryptographically hashed by the CPU. When the enclave is initialized, this hash is finalized, and becomes the enclave's *measurement hash* (§ 5.6).

A remote party can communicate with the enclave to perform *software attestation* (§ 5.8) to convince itself that it is communicating with an enclave that has a specific measurement hash, and is running in a secure environment.

Execution flow can only enter an enclave via special CPU instructions (§ 5.4), similar to the mode switching mechanism for transitioning between user and kernel modes of execution in a typical system. An enclave must execute in protected mode, at ring 3, and uses virtual address translation as set up by the OS kernel and hypervisor.

To avoid leaking private information, a CPU executing enclave code does not directly service any interrupt, fault (e.g., a page fault) or VM exit. Instead, the CPU first performs an Asynchronous Enclave Exit (§ 5.4.3) to switch from enclave code to ring 3 code, and then services the interrupt, fault, or VM exit given scrubbed fault information. The CPU performs an AEX by saving the CPU state into a predefined area inside the enclave and transferring control to a predefined address outside of the enclave, replacing CPU registers with synthetic values.

The allocation of EPC pages to enclaves is delegated to the OS kernel (or hypervisor). The OS communicates its allocation decisions to the SGX platform via special ring 0 CPU instructions (§ 5.3). The OS can also evict EPC pages into untrusted DRAM and later load them back, again using dedicated CPU instructions. SGX uses a cryptographic

Full text available at: http://dx.doi.org/10.1561/1000000051

mechanism to enforce the confidentiality, integrity and freshness of the evicted EPC pages while they are stored in untrusted memory.

1.3 Outline

Reasoning about the security properties of Intel's SGX requires a significant amount of background information that is currently scattered across many sources. For this reason, a significant portion of this work is dedicated to summarizing this prerequisite knowledge.

§ 2 summarizes the relevant subset of modern computer architecture and the micro-architectural properties of recent Intel processors. § 3 outlines the landscape of trusted hardware systems, including cryptographic tools and relevant classes of attacks. Lastly, § 4 briefly describes other trusted hardware systems as context in which SGX was created.

Following this background information, § 5 provides a (sometimes painstakingly) detailed description of SGX's programming model, largely drawing from Intel's Software Development Manual.

A deep analysis of Intel's enclave infrastructure is deferred to part II of this publication (§ II.2), and will analyze other public sources of information, such as Intel's patents relevant to SGX, in order to fill in some of the missing detail in the SGX specification. This discussion is organized into an overview of Intel's implementation of SGX (§ II.2.1), a discussion and analysis of the mechanism by which SGX offers memory access protection to an enclave (§ II.2.2, § II.2.3), and examines SGX as a system for remote attestation (§ II.2.5, § II.2.6). Finally, part II presents a security analysis of SGX overall, and discusses the classes of attacks against which SGX does not offer guarantees(§ II.2.7). The main focus of part II is a detailed review of SGX's security properties to motivate and give context to the MIT Sanctum project (§ II.3) – a flexible, secure, and open source implementation of enclave-capable hardware that offers strong security guarantees against an insidious software adversary.

Full text available at: http://dx.doi.org/10.1561/1000000051

References

FIPS 140-2 Consolidated Validation Certificate No. 0003. 2011.

IBM 4765 Cryptographic Coprocessor Security Module - Security Policy. Dec 2012.

7-Zip LZMA benchmark: Intel Haswell. `http://www.7-cpu.com/cpu/Haswell.html`, 2014. [Online; accessed 10-Februrary-2015].

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].

IPC2 hardware specification. `http://fit-pc.com/download/intense-pc2/documents/ipc2-hw-specification.pdf`, Sep 2014. [Online; accessed 2-Dec-2015].

Gradually sunsetting SHA-1. `http://googleonlinesecurity.blogspot.com/2014/09/gradually-sunsetting-sha-1.html`, 2014. [Online; accessed 4-May-2015].

NIST'S policy on hash functions. `http://csrc.nist.gov/groups/ST/hash/policy.html`, 2014. [Online; accessed 4-May-2015].

BIOS freedom status. `https://puri.sm/posts/bios-freedom-status/`, Nov 2014. [Online; accessed 2-Dec-2015].

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

Full text available at: http://dx.doi.org/10.1561/1000000051

SHA-1 deprecation countdown. https://blogs.windows.com/msedgedev/2016/11/18/countdown-to-sha-1-deprecation/#MPDwCxdpw3IqPPBR.97, 2016. [Online; accessed 18-June-2017].

Seth Abraham. Time to revisit REP;MOVS - comment. https://software.intel.com/en-us/forums/topic/275765, Aug 2006. [Online; accessed 23-January-2015].

Tiago Alves and Don Felton. TrustZone: Integrated hardware and software security. *Information Quarterly*, 3(4):18–24, 2004.

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.

Ross Anderson. *Security engineering: A guide to building dependable distributed systems*. Wiley, 2001.

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].

AMBA® AXI Protocol. ARM Limited, Mar 2004. Reference no. IHI 0022B, IHI 0024B, AR500-DA-10004.

ARM Security Technology Building a Secure System using TrustZone® Technology. ARM Limited, Apr 2009. Reference no. PRD29-GENC-009492C.

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

Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management – part 1: General (revision 3). *Federal Information Processing Standards (FIPS) Special Publications (SP)*, 800-57, Jul 2012.

Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Secure hash standard (SHS). *Federal Information Processing Standards (FIPS) Publications (PUBS)*, 180-4, Aug 2015.

Friedrich Beck. *Integrated Circuit Failure Analysis: a Guide to Preparation Techniques*. John Wiley & Sons, 1998.

Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1. In *Advances in Cryptology – CRYPTO'98*, pages 1–12. Springer, 1998.

Full text available at: http://dx.doi.org/10.1561/1000000051

D. D. Boggs and S. D. Rodgers. Microprocessor with novel instruction for signaling event occurrence and for providing event handling information in response thereto, 1997. US Patent 5,625,788.

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

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.

John Butterworth, Corey Kallenberg, Xeno Kovah, and Amy Herzog. BIOS chronomancy: Fixing the core root of trust for measurement. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security*, pages 25–36. ACM, 2013.

David Champagne and Ruby B. Lee. Scalable architectural support for trusted software. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–12. IEEE, 2010.

Daming D. Chen and Gail-Joon Ahn. Security analysis of x86 processor microcode. 2014. [Online; accessed 7-January-2015].

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.

Lily Chen. Recommendation for key derivation using pseudorandom functions. *Federal Information Processing Standards (FIPS) Special Publications (SP)*, 800-108, Oct 2009.

Coreboot. Developer manual, Sep 2014. [Online; accessed 4-March-2015].

M. P. Cornaby and B. Chaffin. Microinstruction pointer stack including speculative pointers for out-of-order execution, 2007. US Patent 7,231,511.

Intel Corporation. *Intel® Xeon® Processor E5 v3 Family Uncore Performance Monitoring Reference Manual*, Sep 2014. Reference no. 331051-001.

Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. Cryptology ePrint Archive, Report 2015/564, 2015.

Full text available at: http://dx.doi.org/10.1561/1000000051

Victor Costan, Ilia Lebedev, and Srinivas Devadas. Secure processors part II: Intel SGX security analysis and MIT sanctum architecture. In *FnTEDA*, 2017.

J. Daemen and V. Rijmen. AES proposal: Rijndael, AES algorithm submission, Sep 1999.

S. M. Datta and M. J. Kumar. Technique for providing secure firmware, 2013. US Patent 8,429,418.

S. M. Datta, V. J. Zimmer, and M. A. Rothman. System and method for trusted early boot flow, 2010. US Patent 7,752,428.

Pete Dice. Booting an Intel architecture system, part i: Early initialization. *Dr. Dobb's*, Dec 2011. [Online; accessed 2-Dec-2015].

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

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

Morris Dworkin. Recommendation for block cipher modes of operation: Methods and techniques. *Federal Information Processing Standards (FIPS) Special Publications (SP)*, 800-38A, Dec 2001.

Morris Dworkin. Recommendation for block cipher modes of operation: The CMAC mode for authentication. *Federal Information Processing Standards (FIPS) Special Publications (SP)*, 800-38B, May 2005.

Morris Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. *Federal Information Processing Standards (FIPS) Special Publications (SP)*, 800-38D, Nov 2007.

D. Eastlake and P. Jones. RFC 3174: US Secure Hash Algorithm 1 (SHA1). *Internet RFCs*, 2001.

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

Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering: Design Principles and Practical Applications*. John Wiley & Sons, 2011.

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.

Full text available at: http://dx.doi.org/10.1561/1000000051

Agner Fog. Instruction tables - lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs. Dec 2014. [Online; accessed 23-January-2015].

Andrew Furtak, Yuriy Bulygin, Oleksandr Bazhaniuk, John Loucaides, Alexander Matrosov, and Mikhail Gorobets. BIOS and secure boot attacks uncovered. *The 10th ekoparty Security Conference*, 2014. [Online; accessed 22-October-2015].

William Futral and James Greene. *Intel® Trusted Execution Technology for Server Platforms*. Apress Open, 2013.

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.

Blaise Gassend, G. Edward Suh, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Caches and hash trees for efficient memory integrity verification. In *Proceedings of the 9th International Symposium on High-Performance Computer Architecture*, pages 295–306. IEEE, 2003.

Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. Cryptology ePrint Archive, Report 2013/857, 2013.

Daniel Genkin, Itamar Pipman, and Eran Tromer. Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs. Cryptology ePrint Archive, Report 2014/626, 2014.

Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation. Cryptology ePrint Archive, Report 2015/170, 2015.

Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

R. T. George, J. W. Brandt, K. S. Venkatraman, and S. P. Kim. Dynamically partitioning pipeline resources, 2009. US Patent 7,552,255.

A. Glew, G. Hinton, and H. Akkary. Method and apparatus for performing page table walks in a microprocessor capable of processing speculative instructions, 1997. US Patent 5,680,565.

A. F. Glew, H. Akkary, R. P. Colwell, G. J. Hinton, D. B. Papworth, and M. A. Fetterman. Method and apparatus for implementing a non-blocking translation lookaside buffer, 1996. US Patent 5,564,111.

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.

Full text available at: http://dx.doi.org/10.1561/1000000051

J. R. Goodman and H. H. J. Hum. MESIF: A two-hop cache coherency protocol for point-to-point interconnects. 2009.

Joe Grand. Advanced hardware hacking techniques, Jul 2004.

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

Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Rowhammer.js: A remote software-induced fault attack in JavaScript. *CoRR*, abs/1507.06955, 2015. URL `http://arxiv.org/abs/1507.06955`.

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

Ben Hawkes. Security analysis of x86 processor microcode. 2012. [Online; accessed 7-January-2015].

John L. Hennessy and David A. Patterson. *Computer Architecture - a Quantitative Approach (5 ed.)*. Mogran Kaufmann, 2012. ISBN 978-0-12-383872-8.

Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In *Applied cryptography and Network security*, pages 239–252. Springer, 2006.

G. Hildesheim, I. Anati, H. Shafi, S. Raikin, G. Gerzon, U. R. Savagaonkar, C. V. Rozas, F. X. McKeen, M. A. Goldsmith, and D. Prashant. Apparatus and method for page walk extension for enhanced security checks, 2014. US Patent App. 13/730,563.

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.

Gael Hofemeier. Intel manageability firmware recovery agent. Mar 2013. [Online; accessed 2-Dec-2015].

George Hotz. PS3 glitch hack. 2010. [Online; accessed 7-January-2015].

Andrew Huang. *Hacking the Xbox: an Introduction to Reverse Engineering*. No Starch Press, 2003.

C. J. Hughes, Y. K. Chen, M. Bomb, J. W. Brandt, M. J. Buxton, M. J. Charney, S. Chennupaty, J. Corbal, M. G. Dixon, M. B. Girkar, Jonathan C. Hall, Hideki (Saito) Ido, Peter Lachner, Gilbert Neiger, Chris J. Newburn, Rajesh S. Parthasarathy, Bret L. Toll, Robert Valentine, and Jeffrey G. Wiedemeier. Gathering and scattering multiple data elements, 2013. US Patent 8,447,962.

Full text available at: http://dx.doi.org/10.1561/1000000051

IEEE Standard for Ethernet. IEEE Computer Society, Dec 2012. IEEE Std. 802.3-2012.

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.

Intel® Processor Serial Number. Intel Corporation, Mar 1999. Order no. 245125-001.

An Introduction to the Intel® QuickPath Interconnect. Intel Corporation, Mar 2010a. Reference no. 323535-001.

Minimal Intel® Architecture Boot Loader–Bare Bones Functionality Required for Booting an Intel® Architecture Platform. Intel Corporation, Jan 2010b. Reference no. 323246.

Intel® Core 2 Duo and Intel® Core 2 Solo Processor for Intel® Centrino® Duo Processor Technology Intel® Celeron® Processor 500 Series - Specification Update. Intel Corporation, Dec 2010c. Reference no. 314079-026.

Intel® architecture Platform Basics. Intel Corporation, Sep 2010d. Reference no. 324377.

Intel® Trusted Execution Technology (Intel® TXT) LAB Handout. Intel Corporation, 2010e. [Online; accessed 2-July-2015].

Intel® Xeon® Processor 7500 Series Uncore Programming Guide. Intel Corporation, Mar 2010f. Reference no. 323535-001.

Intel® 7 Series Family - Intel® Management Engine Firmware 8.1 - 1.5MB Firmware Bring Up Guide. Intel Corporation, Jul 2012a. Revision 8.1.0.1248 - PV Release.

Intel® Xeon® Processor E5-2600 Product Family Uncore Performance Monitoring Guide. Intel Corporation, Mar 2012b. Reference no. 327043-001.

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

Intel® Xeon® Processor 7500 Series Datasheet - Volume Two. Intel Corporation, Mar 2014a. Reference no. 329595-002.

Intel® Xeon® Processor E7 v2 2800/4800/8800 Product Family Datasheet - Volume Two. Intel Corporation, Mar 2014b. Reference no. 329595-002.

Intel® 64 and IA-32 Architectures Optimization Reference Manual. Intel Corporation, Sep 2014c. Reference no. 248966-030.

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

Full text available at: http://dx.doi.org/10.1561/1000000051

Intel® 100 Series Chipset Family Platform Controller Hub (PCH) Datasheet - Volume One. Intel Corporation, Aug 2015a. Reference no. 332690-001EN.

Mobile 4th Generation Intel® Core® Processor Family I/O Datasheet. Intel Corporation, Feb 2015b. Reference no. 329003-003.

Intel® Xeon® Processor E5-1600, E5-2400, and E5-2600 v3 Product Family Datasheet - Volume Two. Intel Corporation, Jan 2015c. Reference no. 330784-002.

Intel® Xeon® Processor 5500 Series - Specification Update. Intel Corporation, 2 2015d. Reference no. 321324-018US.

Intel® Xeon® Processor E5 Product Family - Specification Update. Intel Corporation, Jan 2015e. Reference no. 326150-018.

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

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

Intel® C610 Series Chipset and Intel® X99 Chipset Platform Controller Hub (PCH) Datasheet. Intel Corporation, Oct 2015h. Reference no. 330788-003.

Bruce Jacob and Trevor Mudge. Virtual memory: Issues of implementation. *Computer*, 31(6):33–43, 1998.

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.

Jakob Jonsson and Burt Kaliski. RFC 3447: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. *Internet RFCs*, Feb 2003.

Burt Kaliski. RFC 2313: PKCS #1: RSA Encryption Version 1.5. *Internet RFCs*, Mar 1998.

Burt Kaliski and Jessica Staddon. RFC 2437: PKCS #1: RSA Encryption Version 2.0. *Internet RFCs*, Oct 1998.

Corey Kallenberg, Xeno Kovah, John Butterworth, and Sam Cornwell. Extreme privilege escalation on windows 8/UEFI systems, 2014.

Full text available at: http://dx.doi.org/10.1561/1000000051

Emilia Käsper and Peter Schwabe. Faster and timing-attack resistant AES-GCM. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 1–17. Springer, 2009.

Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC Press, 2014.

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.

Taesoo Kim and Nickolai Zeldovich. Practical and effective sandboxing for non-root users. In *USENIX Annual Technical Conference*, pages 139–144, 2013.

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.

L. A. Knauth and P. J. Irelan. Apparatus and method for providing eventing ip and source data address in a statistical sampling infrastructure, 2014. US Patent App. 13/976,613.

N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48 (177):203–209, 1987.

Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology (CRYPTO)*, pages 388–397. Springer, 1999.

Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology – CRYPTOâĂŹ96*, pages 104–113. Springer, 1996.

Hugo Krawczyk, Ran Canetti, and Mihir Bellare. HMAC: Keyed-hashing for message authentication. 1997.

Markus G. Kuhn. Electromagnetic eavesdropping risks of flat-panel displays. In *Privacy Enhancing Technologies*, pages 88–107. Springer, 2005.

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.

David Levinthal. Performance analysis guide for Intel® Core i7 processor and Intel® Xeon 5500 processors. `https://software.intel.com/sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf`, 2010. [Online; accessed 26-January-2015].

Full text available at: http://dx.doi.org/10.1561/1000000051

David Lie, Chandramohan Thekkath, Mark Mitchell, Patrick Lincoln, Dan Boneh, John Mitchell, and Mark Horowitz. Architectural support for copy and tamper resistant software. *ACM SIGPLAN Notices*, 35(11):168–177, 2000.

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.

Barbara Liskov and Stephen Zilles. Programming with abstract data types. In *ACM Sigplan Notices*, volume 9, pages 50–59. ACM, 1974.

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.

Martin Maas, Eric Love, Emil Stefanov, Mohit Tiwari, Elaine Shi, Krste Asanovic, John Kubiatowicz, and Dawn Song. Phantom: Practical oblivious computation in a secure processor. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 311–324. ACM, 2013.

James Manger. A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS# 1 v2.0. In *Advances in Cryptology – CRYPTO 2001*, pages 230–238. Springer, 2001.

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.

Jonathan M. McCune, Yanlin Li, Ning Qu, Zongwei Zhou, Anupam Datta, Virgil Gligor, and Adrian Perrig. TrustVisor: Efficient TCB reduction and attestation. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 143–158. IEEE, 2010.

David McGrew and John Viega. The galois/counter mode of operation (GCM). 2004. [Online; accessed 28-December-2015].

Full text available at: http://dx.doi.org/10.1561/1000000051

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.

Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.

National Institute of Standards and Technology (NIST). The advanced encryption standard (AES). *Federal Information Processing Standards (FIPS) Publications (PUBS)*, 197, Nov 2001.

National Institute of Standards and Technology (NIST). The digital signature standard (DSS). *Federal Information Processing Standards (FIPS) Processing Standards Publications (PUBS)*, 186-4, Jul 2013.

National Security Agency (NSA) Central Security Service (CSS). Cryptography today on suite B phase-out. `https://www.nsa.gov/ia/programs/suiteb_cryptography/`, Aug 2015. [Online; accessed 28-December-2015].

M. S. Natu, S. Datta, J. Wiedemeier, J. R. Vash, S. Kottapalli, S. P. Bobholz, and A. Baum. Supporting advanced RAS features in a secured computing system, 2012. US Patent 8,301,907.

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.

Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: the case of AES. In *Topics in Cryptology–CT-RSA 2006*, pages 1–20. Springer, 2006.

Scott Owens, Susmit Sarkar, and Peter Sewell. A better x86 memory model: x86-TSO (extended version). *University of Cambridge, Computer Laboratory, Technical Report*, (UCAM-CL-TR-745), 2009.

Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. ACCessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, page 9. ACM, 2012.

Full text available at: http://dx.doi.org/10.1561/1000000051

D. B. Papworth, G. J. Hinton, M. A. Fetterman, R. P. Colwell, and A. F. Glew. Exception handling in a processor that performs speculative out-of-order instruction execution, 1999. US Patent 5,987,600.

David A. Patterson and John L. Hennessy. *Computer Organization and Design: the hardware/software interface*. Morgan Kaufmann, 2013. ISBN 978-0-12-374750-1.

P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard. Reverse engineering Intel DRAM addressing and exploitation. *ArXiv e-prints*, Nov 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.

S. A. Qureshi and M. O. Nicholes. System and method for using a firmware interface table to dynamically load an ACPI SSDT, 2006. US Patent 6,990,576.

S. Raikin and R. Valentine. Gather cache architecture, 2014. US Patent 8,688,962.

S. Raikin, O. Hamama, R. S. Chappell, C. B. Rust, H. S. Luu, L. A. Ong, and G. Hildesheim. Apparatus and method for a multiple page size translation lookaside buffer (TLB), 2014. US Patent App. 13/730,411.

Stefan Reinauer. x86 Intel: Add firmware interface table support. `http://review.coreboot.org/#/c/2642/`, 2013. [Online; accessed 2-July-2015].

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.

R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21 (2):120–126, 1978.

S. D. Rodgers, K. K. Tiruvallur, M. W. Rhodehamel, K. G. Konigsfeld, A. F. Glew, H. Akkary, M. A. Karnik, and J. A. Brayton. Method and apparatus for performing operations based upon the addresses of microinstructions, 1997. US Patent 5,636,374.

S. D. Rodgers, R. Vidwans, J. Huang, M. A. Fetterman, and K. Huck. Method and apparatus for generating event handler vectors based on both operating mode and event type, 1999. US Patent 5,889,982.

Full text available at: http://dx.doi.org/10.1561/1000000051

M. Rosenblum and T. Garfinkel. Virtual machine monitors: current technology and future trends. *Computer*, 38(5):39–47, May 2005.

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

Joanna Rutkowska. Intel x86 considered harmful. `https://blog.invisiblethings.org/papers/2015/x86_harmful.pdf`, Oct 2015. [Online; accessed 2-Nov-2015].

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

Jerome H. Saltzer and M. Frans Kaashoek. *Principles of Computer System Design: An Introduction*. Morgan Kaufmann, 2009.

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 and S. J. Robinson. Enabling virtualization of a processor resource, 2014. US Patent 8,806,104.

Stephen Shankland. Itanium: A cautionary tale. Dec 2005. [Online; accessed 11-February-2015].

Alan Jay Smith. Cache memories. *ACM Computing Surveys (CSUR)*, 14(3): 473–530, 1982.

Sean W. Smith and Steve Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks*, 31(8):831–860, 1999.

Sean W. Smith, Ron Perez, Steve Weingart, and Vernon Austel. Validating a high-performance, programmable secure coprocessor. In *22nd National Information Systems Security Conference*. IBM Thomas J. Watson Research Division, 1999.

Marc Stevens, Pierre Karpman, and Thomas Peyrin. Free-start collision on full SHA-1. Cryptology ePrint Archive, Report 2015/967, 2015.

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.

Full text available at: http://dx.doi.org/10.1561/1000000051

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.

Trusted Computing Group TCG. Tpm main specification. `http://www.trustedcomputinggroup.org/resources/tpm_main_specification`, 2003.

Alexander Tereshkin and Rafal Wojtczuk. Introducing ring-3 rootkits. Master's thesis, 2009.

Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Proceedings of the 28th European Solid-State Circuits Conference (ESSCIRC)*, pages 403–406. IEEE, 2002.

Unified Extensible Firmware Interface Specification, Version 2.5. UEFI Forum, 2015. [Online; accessed 1-Jul-2015].

Rich Uhlig, Gil Neiger, Dion Rodgers, Amy L. Santoni, Fernando C. M. Martins, Andrew V. Anderson, Steven M. Bennett, Alain Kagi, Felix H. Leung, and Larry Smith. Intel virtualization technology. *Computer*, 38(5):48–56, 2005.

Wim Van Eck. Electromagnetic radiation from video display units: an eavesdropping risk? *Computers & Security*, 4(4):269–286, 1985.

Amit Vasudevan, Jonathan M. McCune, Ning Qu, Leendert Van Doorn, and Adrian Perrig. Requirements for an integrity-protected hypervisor on the x86 hardware virtualized architecture. In *Trust and Trustworthy Computing*, pages 141–165. Springer, 2010.

Sathish Venkataramani. *Advanced Board Bring Up - Power Sequencing Guide for Embedded Intel Architecture.* Intel Corporation, Apr 2011. Reference no. 325268.

Vassilios Ververis. Security evaluation of Intel's active management technology. 2010.

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

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.

Full text available at: http://dx.doi.org/10.1561/1000000051

Rafal Wojtczuk and Joanna Rutkowska. Attacking intel TXT via SINIT code execution hijacking, 2011.

Rafal Wojtczuk and Alexander Tereshkin. Attacking Intel® BIOS. *Invisible Things Lab*, 2010.

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

Y. Wu and M. Breternitz. Genetic algorithm for microcode compression, 2008. US Patent 7,451,121.

Y. Wu, S. Kim, M. Breternitz, and H. Hum. Compressing and accessing a microcode ROM, 2012. US Patent 8,099,587.

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.

A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167, 1986.

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. *Using secure coprocessors*. PhD thesis, Carnegie Mellon University, 1994.

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.

Xiantao Zhang and Yaozu Dong. Optimizing Xen VMM based on Intel® virtualization technology. In *Internet Computing in Science and Engineering, 2008. ICICSE'08. International Conference on*, pages 367–374. IEEE, 2008.

Li Zhuang, Feng Zhou, and J. Doug Tygar. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):3, 2009.

Full text available at: http://dx.doi.org/10.1561/1000000051

V. J. Zimmer and S. H. Robinson. Methods and systems for microcode patching, 2012. US Patent 8,296,528.

V. J. Zimmer and J. Yao. Method and apparatus for sequential hypervisor invocation, 2012. US Patent 8,321,931.