

Cloud Computing Security: Foundations and Research Directions

Other titles in Foundations and Trends® in Privacy and Security

Expressing Information Flow Properties

Elisavet Kozyri, Stephen Chong and Andrew C. Myers

ISBN: 978-1-68083-936-4

Accountability in Computing: Concepts and Mechanisms

Joan Feigenbaum, Aaron D. Jagard and Rebecca N. Wright

ISBN: 978-1-68083-784-1

A Pragmatic Introduction to Secure Multi-Party Computation

David Evans, Vladimir Kolesnikov and Mike Rosulek

ISBN: 978-1-68083-508-3

Contextual Integrity through the Lens of Computer Science

Sebastian Benthall, Seda Gurses and Helen Nissenbaum

ISBN: 978-1-68083-384-3

Methods for Location Privacy: A comparative overview

Kostantinos Chatzikokolakis, Ehab ElSalamouny, Catuscia Palamidessi
and Pazii Anna

ISBN: 978-1-68083-366-9

Cloud Computing Security: Foundations and Research Directions

Anrin Chakraborti

Duke University
anrin.chakraborti@duke.edu

Jonathan Katz

University of Maryland, College Park
jkatz@cs.umd.edu

Ahmad-Reza Sadeghi

Technische Universität Darmstadt
ahmad.sadeghi@trust.tu-darmstadt.de

Yinqian Zhang

Southern University of Science and Technology
yinqianz@acm.org

Reza Curtmola

New Jersey Institute of Technology
reza.curtmola@njit.edu

Jason Nieh

Columbia University
nieh@cs.columbia.edu

Radu Sion

Stony Brook University
sion@cs.stonybrook.edu

now

the essence of knowledge

Boston — Delft



Documents

Document Sharing Hub

Foundations and Trends® in Privacy and Security

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

A. Chakraborti *et al.* . *Cloud Computing Security: Foundations and Research Directions*. Foundations and Trends® in Privacy and Security, vol. 3, no. 2, pp. 103–213, 2022.

ISBN: 978-1-68083-959-3
© 2022 A. Chakraborti *et al.*

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

Foundations and Trends® in Privacy and Security

Volume 3, Issue 2, 2022

Editorial Board

Editors-in-Chief

Anupam Datta

Carnegie Mellon University, USA

Jeannette Wing

Columbia University, USA

Editors

Martín Abadi

*Google and University of California,
Santa Cruz*

Michael Backes

Saarland University

Dan Boneh

Stanford University, USA

Véronique Cortier

LORIA, CNRS, France

Lorrie Cranor

Carnegie Mellon University

Cédric Fournet

Microsoft Research

Virgil Gligor

Carnegie Mellon University

Jean-Pierre Hubaux

EPFL

Deirdre Mulligan

University of California, Berkeley

Andrew Myers

Cornell University

Helen Nissenbaum

New York University

Michael Reiter

University of North Carolina

Shankar Sastry

University of California, Berkeley

Dawn Song

University of California, Berkeley

Daniel Weitzner

Massachusetts Institute of Technology

Editorial Scope

Topics

Foundations and Trends® in Privacy and Security publishes survey and tutorial articles in the following topics:

- Access control
- Accountability
- Anonymity
- Application security
- Artificial intelligence methods in security and privacy
- Authentication
- Big data analytics and privacy
- Cloud security
- Cyber-physical systems security and privacy
- Distributed systems security and privacy
- Embedded systems security and privacy
- Forensics
- Hardware security
- Human factors in security and privacy
- Information flow
- Intrusion detection
- Malware
- Metrics
- Mobile security and privacy
- Language-based security and privacy
- Network security
- Privacy-preserving systems
- Protocol security
- Security and privacy policies
- Security architectures
- System security
- Web security and privacy

Information for Librarians

Foundations and Trends® in Privacy and Security, 2022, Volume 3, 4 issues. ISSN paper version 2474-1558. ISSN online version 2474-1566. Also available as a combined paper and online subscription.

Contents

1	Introduction	2
1.1	Secure Infrastructure	3
1.2	Secure Computation	4
1.3	Secure Storage	5
I	Secure Infrastructure	7
2	Virtualization	8
2.1	Trusted Designs	9
2.2	Formally-Verified Hypervisors	11
2.3	Future Directions	13
3	Hardware-Enabled Security	14
3.1	Trusted Execution Environments	15
3.2	Open Challenges and Future Research Direction	23
4	Side-Channels	24
4.1	Side-channel Attacks	25
4.2	Side-channel Defenses	33
4.3	Future Directions	36

II	Secure Computation	38
5	Secure Distributed Computation	39
5.1	The Ideal World	41
5.2	The Real World	44
5.3	Defining Security	45
5.4	Feasibility Results and Secure Protocols	47
5.5	Open Questions	48
6	Encrypted Search	49
6.1	Searchable Encryption	49
6.2	Encrypted Databases	52
III	Secure Storage	53
7	Data Access Privacy	54
7.1	Access Privacy	56
7.2	Communication-Efficient ORAMs	58
7.3	Round-Trip Efficient ORAMs	61
7.4	Compute-Efficient ORAMs	62
7.5	Other Practical Considerations	62
8	Provable Data Possession	67
8.1	Prior Approaches	69
8.2	Provable Data Possession	69
8.3	Dynamic Provable Data Possession	71
8.4	Proofs of Retrievability	72
8.5	Towards Auditing Distributed Storage Systems	73
8.6	Remote Data Integrity Checking With Server-side Repair	76
8.7	Future Research Directions	78
9	Acknowledgements	80
	References	81

Cloud Computing Security: Foundations and Research Directions

Anrin Chakraborti¹, Reza Curtmola², Jonathan Katz³, Jason Nieh⁴, Ahmad-Reza Sadeghi⁵, Radu Sion⁶ and Yinqian Zhang⁷

¹*Duke University, USA; anrin.chakraborti@duke.edu*

²*New Jersey Institute of Technology, USA; reza.curtmola@njit.edu*

³*University of Maryland, College Park, USA; jkatz@cs.umd.edu*

⁴*Columbia University, USA; nieh@cs.columbia.edu*

⁵*Technische Universität Darmstadt, Germany;*

ahmad.sadeghi@trust.tu-darmstadt.de

⁶*Stony Brook University, USA; sion@cs.stonybrook.edu*

⁷*Southern University of Science and Technology, China;*

yinqianz@acm.org

ABSTRACT

Cloud services have revolutionized modern computing. The benefits of outsourcing data and computation come with security and privacy concerns. This monograph explores the advances in cloud security research across both industry and academia, with a special focus on secure infrastructure, services and storage. Besides overviewing the state of the art, the monograph highlights open problems, and possible future research directions.

Anrin Chakraborti, Reza Curtmola, Jonathan Katz, Jason Nieh, Ahmad-Reza Sadeghi, Radu Sion and Yinqian Zhang (2022), “Cloud Computing Security: Foundations and Research Directions”, Foundations and Trends® in Privacy and Security: Vol. 3, No. 2, pp 103–213. DOI: 10.1561/33000000028.

©2022 A. Chakraborti *et al.*

1

Introduction

Cloud services have revolutionized computing in the modern world. In an increasingly networked ecosystem, it is commonplace for enterprises and private parties alike to leverage cloud services for storage and compute. The most obvious benefits include scalability, increased availability, and the potential for reduced costs¹ when compared to lower-scale on-premise infrastructures. In addition, cloud-hosted data (and compute) is accessible across platforms and is not limited by geographical constraints making collaboration attractively viable.

However, these benefits come with their share of pitfalls. Over time, cloud architecture have become increasingly complex. Cloud platforms today run tens and sometimes hundreds of millions of lines of code to support a wide range of services and capabilities. From a security perspective, this results in an enormously large attack surface, which is now much more attractive to determined knowledge and resource-intensive attackers, mainly due to its potential to expose millions of customers' critical data. This is further exacerbated by the fact that multi-tenancy, inherent in the very fabric of the cloud value proposition,

¹But lower costs are not a given – and very often, applications not designed to scale properly may incur comparably astronomical costs when run in the cloud.

can bring forth significant and unforeseen security issues including the now-ubiquitous side channels – not usually of concern in single-user systems and enterprise networks – but that now completely compromise vast swaths of the infrastructure and customer workloads. As a result, cloud security has become a focal point for security researchers over the past decade.

This monograph discusses a number of key issues critical in this endeavour. We focus here on challenges that the authors found particularly interesting. We provide an overview of some of the solutions while highlighting noteworthy designs and discussing remaining open problems. We also note that **a holistic complete view of this vast problem space, effectively spanning all layers of modern computing, is out of scope and cannot be addressed in any one piece of work.**

Structure of the Monograph Cloud security is a broad topic encompassing concepts from a large cross section of domains. To make this monograph concise and meaningful, we target several topics and challenges that are almost entirely specific to clouds. For this reason, general computing security topics such as intrusion detection, software protection, phishing etc. are excluded. While these are important building blocks that need to be considered in an end-to-end cloud-centric design, they have been extensively addressed elsewhere.

The monograph is divided into three parts based on a broad clustering into hardware, computation, and storage. Specifically the intuition is that a typical cloud stack will need to: i) secure the platforms on top of which clouds services e.g., on-demand VMs run, ii) secure the services e.g., by providing secure compute capabilities, and iii) secure data stored at rest on the cloud-hosted storage platforms. We now give a brief overview for each part.

1.1 Secure Infrastructure

Cloud infrastructure is extremely complex involving several components such as networks, hardware etc. Nevertheless, a critical cornerstone

component of any contemporary cloud architecture is the underlying computation virtualization technology.

Secure Hypervisors Arguably, systems that enable virtualization e.g., hypervisors constitute the most security-sensitive component of a cloud architecture since they usually run millions of lines of code at the highest privilege level with full access to the underlying hardware and user data. Securing this software is one of the foremost challenges to building secure clouds. Section 2 discusses the state of the art in trusted hypervisor designs, in addition to techniques that formally verify hypervisors for secure deployments.

Hardware-Enabled Security Trusted execution environments (TEEs) are an integral part of cloud infrastructure. They protect the confidentiality and integrity of client application data from other tenants, as well as from an untrusted cloud provider. Widely-deployed TEEs like Intel SGX (Intel Corporation, 2014) and AMD SEV (Kaplan *et al.*, 2016) make it possible to run computation isolated from all the other untrusted software running in the same system, with strong hardware-backed guarantees. Section 3 discusses these technologies and highlights their merits and demerits.

Side-Channels Multi-tenancy in clouds supported by virtualization also introduces other challenges, specifically in the form of side-channels. This is because a cloud tenant may have its computation co-located with other potentially untrusted and malicious parties. This unrestricted sharing of resources between mutually distrustful parties creates new attack vectors that is not typical to single-user systems or even enterprise networks. Section 4 discusses the potential pitfalls of multi-tenancy and outlines solutions that effectively defend against side-channels in cloud environments.

1.2 Secure Computation

With Platform-as-a-service (PaaS) and Software-as-a-service (SaaS), cloud services provide various ways for users to outsource and compute

on cloud hardware. One particularly popular instance of this is machine learning as a service (MLaaS). Naturally, in these settings, the user would like to ensure that the computation is performed with certain verifiable guarantees which includes confidentiality of input/output, correctness of results, etc.

Secure Distributed Computation Multi-party cryptographic techniques are important building blocks for secure cloud computation. Informally, secure multiparty computation involves two (or more) mutually-untrusting parties jointly computing on shared data while ensuring that they learn nothing more than what the protocol specifies about each others' inputs. In the cloud setting, the untrusted party is the cloud service, while the users constitute the trusted parties. Section 5 discusses the results in distributed secure computation which enable secure computing in the cloud.

Encrypted Search Client-side encryption is an essential first step towards protecting data stored on cloud platforms. This strong protection comes at the cost of usability and performance since the server can no longer search and compute on the data. Encrypted search techniques (e.g., searchable encryption) take a middleground approach and enable keyword searches in encrypted documents. Encrypted databases further extend this idea and support rich query functionalities like joins etc. Section 6 discusses these techniques and highlights their merits and demerits.

1.3 Secure Storage

Most cloud services provide Storage-as-a-services (STaaS). This has become a popular option for enterprises to store data in a cost-effective way, as opposed to setting up on-premise data centers. However, the abundance of data on online (often public) spaces raises important security concerns that are not handled by conventional encryption. We discuss two such challenges.

Access Pattern Privacy Section 7 discusses the problem of access privacy for data stored on clouds. It is well-known that revealing access patterns to data can reveal a wealth of information about the contents, even when the data is encrypted. This problem is especially concerning in clouds where the (potentially untrusted) cloud provider has easy access to the data access patterns through access logs etc.

Provable Data Possession Section 8 discusses the problem of provable data possession. Cloud services rarely provide verifiable guarantees with regards to the integrity and long-term reliability of the stored data. If the data is lost, damaged or revealed to unauthorized sources, the damage is irreversible. Provable data possession ensures that clients can verify that an untrusted provider indeed ensures all the proper guarantees for the stored data, and detect corruptions if any.

Part I

Secure Infrastructure

2

Virtualization

Virtualization technologies are the backbone of cloud architectures, enabling users to move their data and computation off-site to cloud-hosted virtual machines (VMs). The most critical component of a virtualized system is the hypervisor which provides the VM abstraction (Bugnion *et al.*, 2017). The hypervisor controls the hardware and typically runs at the highest privilege level. However, commodity hypervisors, that are often integrated with a host operating system kernel, present a large potential attack surface with access to VM data in CPU registers, memory, I/O data and boot images. Attackers that exploit hypervisor vulnerabilities may gain access to all VM data (potentially belonging to different clients) compromising privacy and integrity guarantees. Therefore, securing hypervisors against unauthorized access and other potential attack vectors is of paramount importance to securing cloud services. There is a long line of work on designing more secure hypervisors for clouds. In the following, we highlight various approaches for addressing this problem.

2.1 Trusted Designs

Bare-Metal Hypervisors Unlike hosted hypervisors which integrate with a host operating system kernel, bare-metal hypervisors are standalone hypervisors. They often claim a smaller TCB as an advantage over hosted hypervisors, but in practice, the aggregate TCB of the widely-used Xen (Barham *et al.*, 2003) bare-metal hypervisor includes *Dom0* (Colp *et al.*, 2011; Zhang *et al.*, 2011a), which includes an entire Linux kernel, and therefore is no smaller than hosted hypervisors like KVM (Kivity *et al.*, 2007; Dall and Nieh, 2014). Some work thus focuses on reducing Xen’s attack surface by redesigning *Dom0* (Murray *et al.*, 2008; Colp *et al.*, 2011; Butt *et al.*, 2012).

Microhypervisors Microhypervisors (Steinberg and Kauer, 2010; Heiser and Leslie, 2010) take a microkernel approach to build clean-slate hypervisors from scratch to reduce the hypervisor TCB. For example, NOVA (Steinberg and Kauer, 2010) moves various aspects of virtualization such as CPU and I/O virtualization to user space services. The virtualization services are trusted but instantiated per VM so that compromising them only affects the given VM. Others simplify the hypervisor to reduce its TCB by removing (Shinagawa *et al.*, 2009) or disabling (Nguyen *et al.*, 2012) virtual device I/O support in hypervisors, or partitioning VM resources statically (Keller *et al.*, 2010; Siemens, 2019). HyperLock (Wang *et al.*, 2012), DeHype (Wu *et al.*, 2013), and Nexen (Shi *et al.*, 2017) focus on deconstructing existing monolithic hypervisors by segregating hypervisor functions to per VM instances. While this can isolate an exploit of hypervisor functions to a given VM instance, if a vulnerability is exploitable in one VM instance, it is likely to be exploitable in another as well. Nexen builds on Nested Kernel to retrofit Xen in this manner, though it does not protect against vulnerabilities in its shared hypervisor services.

Retrofitting Commodity Hypervisors A different approach to building trusted hypervisors is retrofitting a commodity hypervisor to reduce the TCB based on microkernel principles while inheriting its extensively virtualization features. Hypsec (Li *et al.*, 2019) partitions a mono-

lithic hypervisor into a trusted core, called the *corevisor* and a large untrusted host called the *hostvisor*. Hypsec leverages hardware virtualization support to isolate and protect the corevisor and execute it at a higher privilege level than the hostvisor. The corevisor has full access to hardware resources, provides basic CPU and memory utilization and mediates all exception and interrupts. More complex operations such as I/O and interrupt virtualization, and resource management is delegated to the hostvisor.

CloudVisor (Zhang *et al.*, 2011a) uses a small, specialized host hypervisor to support nested virtualization and protect user VMs against an untrusted Xen guest hypervisor, though Xen modifications are required. CloudVisor encrypts VM I/O and memory but does not fully protect CPU state, contrary to its claims of “providing both secrecy and integrity to a VM’s states, including CPU states.” For example, the VM program counter is exposed to Xen to support I/O. As with any nested virtualization approach, performance overhead on application workloads is a problem. Furthermore, CloudVisor does not support widely used paravirtual I/O. CloudVisor has a smaller TCB by not supporting public key cryptography, making key management problematic.

2.1.1 Software Protection

Various projects extend a trusted hypervisor to protect software within VMs, including protecting applications running on an untrusted guest OS in the VM (Chen *et al.*, 2008; Yang and Shin, 2008; McCune *et al.*, 2010; Chhabra *et al.*, 2011; Hofmann *et al.*, 2013), ensuring kernel integrity and protecting against rootkits and code injection attacks or to isolate I/O channels (Wang *et al.*, 2009; Riley *et al.*, 2008; Seshadri *et al.*, 2007; Wang *et al.*, 2015b; Zhou *et al.*, 2014), and dividing applications and system components in VMs then relying on the hypervisor to safeguard interactions among secure and insecure components (Garfinkel *et al.*, 2003; Strackx and Piessens, 2012; Ta-Min *et al.*, 2006; Liu *et al.*, 2015b). Overshadow (Chen *et al.*, 2008) and Inktag (Hofmann *et al.*, 2013) use a more trusted hypervisor component to protect against untrusted kernel software. Overshadow and Inktag also assume applications use end-to-end encrypted network I/O, though

they protect file I/O by replacing it with memory-mapped I/O to encrypted memory.

Protecting User Data with Specialized Hardware To protect user data in virtualization systems, there is work that require VM support for specialized hardware such as Intel SGX Intel Corporation, 2014 or ARM TrustZone. These hardware-backed designs are discussed in more detail in Section 3. Haven (Baumann *et al.*, 2014a) and S-NFV (Shih *et al.*, 2016) use Intel SGX to protect application data. Others (Zhu *et al.*, 2017; Hua *et al.*, 2017) run a security monitor in ARM TrustZone and rely on ARM IP features such as TrustZone Address Space Controller to protect VMs. vTZ (Hua *et al.*, 2017) virtualizes TrustZone and protects the guest TEE against an untrusted hypervisor, but does not protect the normal world VM. HA-VMSI (Zhu *et al.*, 2017) protects the normal world VM against a compromised hypervisor but supports limited virtualization features.

2.2 Formally-Verified Hypervisors

Formally-verified hypervisors provide a theoretically-sound approach to building trusted hypervisors for clouds. seL4 (Klein *et al.*, 2009; Klein *et al.*, 2014) and CertiKOS (Gu *et al.*, 2016; Gu *et al.*, 2019) are verified systems with hypervisor functionality. While various versions of seL4 exist, noninterference properties and functional correctness have only been verified on a single uniprocessor version (*seL4 Supported Platforms n.d.*); bugs have been discovered in other seL4 versions (Oberhauser *et al.*, 2021). The verified version only supports Armv7 hardware and has no virtualization support (*seL4 Supported Platforms n.d.*). Another seL4 Armv7 version verifies the functional correctness of some hypervisor features, but not MMU functionality (*seL4 Supported Platforms n.d.*; Klein *et al.*, 2018), which is at the core of a functional hypervisor. seL4 does not support shared page tables (*seL4 Reference Manual Version 11.0.0* 2019), and verifying multiprocessor and hypervisor support remain future work (*Frequently Asked Questions on seL4 n.d.*). It lacks most features expected of a hypervisor. Its device support via virtio is unverified and also needs to be ported to its platform, limiting its virtio

functionality. For example, seL4 lacks support for virtio block devices and has no vhost optimization. Building a system using seL4 is much harder than using Linux ([Frequently Asked Questions on seL4](#) n.d.).

CertiKOS proves noninterference for the sequential mCertiKOS kernel (Costanzo *et al.*, 2016) without virtualization support and goes beyond seL4 in verifying the functional correctness of the mC2 multiprocessor kernel with virtualization. However, mC2 provides no data confidentiality and integrity among VMs. Like seL4, CertiKOS also cannot verify shared page tables, so it does not provide verified support for multiprocessor VMs. The verified kernel does not work on modern 64-bit hardware. It lacks many hypervisor features, including dynamically allocated page tables for multi-level paging, huge page support, device passthrough, and VM migration. Its virtio support does not include vhost, is limited to only certain block devices, and requires porting virtio to its platform, making it difficult to keep up with virtio improvements and updates.

Other works have only partially verified their hypervisor code to reduce proof effort. The VCC framework has been used to verify 20% of Microsoft's Hyper-V multiprocessor hypervisor, but global security properties remain unproven (Cohen *et al.*, 2009; Leinenbach and Santen, 2009). überSpark has been used to verify the üXMHF hypervisor, but their architecture does not support concurrent hardware access, and their verification approach foregoes functional correctness (Vasudevan *et al.*, 2013; Vasudevan *et al.*, 2016).

Despite these advances, it is infeasible to fully verify commercially-deployed hypervisors which may contain more than 2M LOC e.g., KVM, the widely-used Linux hypervisor. As an elegant and scalable solution to this problem, Li *et al.* (2021a) introduce the idea of *microverification* to reduce proof effort by retrofitting a commodity system into a small core and a set of untrusted services, so that it is possible to reason about the properties of the entire system by verifying the core alone. Based on this idea, they introduced *MicroV*, a new framework for verifying the security properties of large, multiprocessor commodity systems to provably guarantee data confidentiality and integrity. They used this verification framework to verify the security properties of a commodity hypervisor for the first time, specifically KVM (Li *et al.*,

2021a; Li *et al.*, 2021b; Tao *et al.*, 2021). The approach does not verify that KVM is entirely functionally correct, but rather proves that its core is functionally correct and that is sufficient to ensure the security properties hold for the entire hypervisor despite potential vulnerabilities in the rest of KVM.

2.3 Future Directions

Formal verification remains an important ongoing area of work in proving the security of hypervisors. Although the security properties of KVM have been verified, it remains to be seen whether similar approaches can be used to verify other commercially-deployed hypervisors with different architectures and implementations, such as Xen or VMware. Another important consideration is whether formal verification can be used to verify other security properties other than confidentiality and integrity. For example, in some domains, availability may be of greater concern, such as in the context of security-critical systems for cyber-physical systems such as self-driving cars, in which security critical components may be isolated in VMs but must be guaranteed to be able to run. It remains to be seen how formal verification techniques might be used to prove availability guarantees for a commodity hypervisor.

Another important direction of future work is hardware security for virtualization. Hardware mechanisms have increasingly been introduced to support various security features for applications. As hypervisors are part of the critical software infrastructure that must be secured, it is only a matter of time until hardware mechanisms are introduced to support the improved security of hypervisors. For example, if hardware were available that could guarantee the confidentiality and integrity of applications despite potential hypervisor vulnerabilities, this would substantially mitigate the security risk due to complex commercially-deployed hypervisors. Such hardware might provide mechanisms that a hypervisor could leverage to manage VMs without giving the hypervisor access to VM data.

3

Hardware-Enabled Security

In this section, we discuss *Trusted Execution Environments (TEEs)*, often enabled by specialized secure hardware, which is increasingly finding its way into cloud infrastructures. Contemporary workload isolation mechanisms in cloud computing focus on protecting the cloud environment from accesses by untrusted tenants, i.e., they provide security *for* the cloud service provider. Very often however, tenants want to protect the confidentiality and integrity of their application data not only from other tenants but also *from* the cloud itself, and specifically higher privileged system software, such as the hypervisor. In most of the current cloud infrastructures, clients rely on the service provider for data and code integrity. However, this is problematic since the cloud infrastructures may be compromised by malicious actors or software bugs. Since formally-verified hypervisors, discussed in Section 2, are not yet ubiquitously deployed, cloud customers are left with no choice other than to rely on the cloud service provider security measures. Trust must be placed in the complete software stack, in particular the hypervisor, with the inherent underlying risk whereby a compromised component can lead to severe data leakages.

Hardware-enforced isolated execution environments, also known as TEEs, represent a paradigm shift in computer architectures, which will impact the computing landscape fundamentally. It will revolutionize cloud computing and enable many new applications and scenarios, similar to hardware-based virtualization kick-started cloud computing in general. In this section we will discuss TEEs in the context of cloud computing in detail. First, we give an overview of some of the major TEEs, or TEE architectures as we call them in the reminder of this section, that are already available on today's server platforms. Next, we discuss in detail which problems TEE architectures currently face. On the one side, these are security problems mainly in the area of side-channel attacks. On the other side, TEE architectures also face performance and functionality problems, e.g., when it comes to the support of legacy cloud-native workloads. In the end, we point to recent research which aims to tackle the aforementioned problems and we identify important future research directions.

3.1 Trusted Execution Environments

TEE architectures are already actively used to protect sensitive cloud computing workloads. The biggest players in the server market, Intel and AMD, both designed TEE architectures, namely, Intel Software Guard Extensions (SGX) and AMD Secure Encrypted Virtualization (SEV) which we describe in this section. Moreover, we point to new emerging TEE architectures not yet available which were specifically designed to secure cloud computing.

3.1.1 Intel Software Guard Extensions

The server market is still dominated by Intel which achieved a market share of 92% in 2020 (TrendForce, [2021](#)). In 2015, Intel SGX (Intel, [2014](#)) was introduced with the Skylake microarchitecture as one of the first TEE architectures which protects sensitive applications in isolated execute context, called *enclaves*. Intel SGX was first introduced in the desktop processors but later also included in the server processors to enable the protection of secure microservices in the cloud.

Adversary model. The adversary model of SGX assumes a strong adversary that can compromise all system software, even the operating system kernel and hypervisor. Furthermore, the adversary is able to compromise or misconfigure system peripherals in order to perform Direct Memory Access (DMA) attacks (Markettos *et al.*, 2019; Zhou *et al.*, 2012). Cache side-channel attacks are not considered in the adversary model of SGX (Johnson, 2018). Also, importantly, while the adversary is allowed to perform simple physical attacks on associated components such as DRAM (e.g., cold-boot attacks (Halderman *et al.*, 2009)), she is assumed to *not be able to perform more sophisticated physical penetration or exfiltration attacks on the overall system and its CPUs.*

Design & features. In SGX, instances of a TEE, called enclaves, are used to execute sensitive program code in user space, isolated from each other and from a potentially malicious operating system or hypervisor. Each enclave is bundled with a regular non-sensitive application which invokes the enclave as a child process. During the enclave setup, the integrity of the enclave code is verified (attested), i.e., an authenticated measurement (i.e., typically a binary hash) of the code loaded into the enclave is reported either locally or remotely. When an enclave is executed, it shares its virtual address space with its host process. The untrusted operating system performs the enclave memory management, handles the enclave exceptions and provides I/O services to the enclaves (Intel, 2017). All enclave code or data leaving the CPU is always encrypted before being stored in memory. This allows SGX to protect enclaves from simple hardware attacks and malicious DMA requests. Furthermore, it enables SGX to persistently store the enclave states after their execution.

Hardware primitives & TCB. The enclave code and data (e.g., its page tables) is protected from unauthorized access by system software through primitives in hardware. SGX is mostly implemented inside of the CPU through microcode, e.g., a new set of instructions to control and communicate with the enclaves was introduced. Additionally, minimal hardware changes are made at the page table walker. When a page fault

is triggered as a result of a TLB miss, the SGX microcode checks if the virtual address points to enclave memory and subsequently performs access control on it. Only if the current execution context is allowed to access the enclave memory, the address translation is performed and the physical address loaded into the TLB. The transparent encryption outside of the CPU package is performed by a new hardware component called the Memory Encryption Engine (MEE). Together with the MEE, the CPU forms the hardware TCB of the system. The enclave attestation scheme is implemented in a set of enclaves provided by Intel, e.g. the Quoting Enclave which computes attestation signatures. These privileged enclaves have access to the keys in the SGX hardware and represent the software TCB of the system.

3.1.2 AMD Secure Encrypted Virtualization

The second biggest player in the server market, AMD, with a market share of 8% in 2020 (TrendForce, [2021](#)) introduced their own TEE architecture for servers platforms, Secure Encrypted Virtualization (SEV) (Kaplan *et al.*, [2016](#)), in 2017.

Adversary model. The adversary model of AMD SEV is comparable to the one assumed in Intel SGX. AMD also assumes a strong adversary which can even compromise the hypervisor layer and perform DMA and simple hardware attacks. As in the adversary model of SGX, cache side-channel attacks are not considered.

Design & features. SEV provides protection for complete Virtual Machines (VM) in cloud scenarios and is designed on top of AMD's Secure Memory Encryption (SME) technology. SEV isolates VMs, which represent a TEE instance, from each other and the underlying hypervisor layer by encrypting each VM transparently with an individually-generated encryption key. The keys are generated from random sources at the VM launch time. Access to these keys is limited to hardware, thus, the hypervisor or any other software component outside the VM cannot interfere with the encryption. An identification key embedded into the firmware can be used from the VM owner to verify the initial state of

its VM through remote attestation. In the first version of SEV, only the VM memory was encrypted when stored in memory outside of the CPU package. The later introduced SEV-ES (Encrypted State) (Kaplan, 2017) extended the protected VM state to the CPU registers to prevent information leakage to the hypervisor (Hetzelt and Buhren, 2017). In contrast to SGX, SEV did not provide integrity for the encrypted memory pages in its first version which was exploited in injection and replay attacks (Hetzelt and Buhren, 2017). However, in 2020, AMD introduced SEV Secure Nested Paging (SNP) (AMD, 2020) which added the missing integrity protection capability.

Hardware primitives & TCB. SEV utilizes the hardware primitives introduced with SME, namely, an AES crypto engine embedded into the memory controller that performs the encryption/decryption of the data leaving the CPU and the AMD Secure Processor (SP). The SP represents a dedicated security subsystem (based on the ARM architecture) that manages the cryptographic key and is integrated into the AMD SoC. Moreover, the SP performs the attestation of the VMs. In SEV-enabled SoCs, SME is combined with the virtualization technology from AMD (AMD-V) such that the SP manages one key per VM. When VM data is loaded into the CPU cache, it is protected by an access control mechanism relying on Address Space Identifiers (ASIDs). The hardware TCB of SEV comprises the CPU, SP and AES crypto engine. Moreover, the SP firmware needs to be trusted.

3.1.3 Emerging Cloud TEE Architectures

The most eye-catching design difference between Intel SGX and AMD SEV is the granularity at which both TEE architectures can protect sensitive applications. Intel SGX isolates at the process level, whereas SEV provides isolation at the VM level. SEV's focus on protecting complete VMs makes it much more suitable for the server market since existing VM workloads can be directly deployed within SEV-protected VMs. In Intel SGX, comparable results can only be achieved with cumbersome workarounds, as we discuss in Section 3.1.6.

Intel Trust Domain Extensions. In order to also provide isolation on a VM granularity level, Intel designed a new TEE architecture called Intel Trusted Domain Extensions (TDX) (Intel, 2021). Comparable to AMD, Intel achieves this by combining its memory encryption technology, called Total Memory Encryption (TME), with its virtualization technology, called Intel VT-x. The assumed adversary model is comparable to AMD SEV since TDX also aims to protect sensitive VMs even from a compromised hypervisor component and thus, from a potentially malicious cloud provider. To attest the isolated VMs (called *Trusted Domains*), TDX leverages the attestation technology introduced with Intel SGX.

Currently, AMD and Intel together control 99% of the global server market, however, there is a new competitor. The ARM architecture, which is the dominating processor architecture in the embedded market, offers characteristics which make their usage on server platforms very promising, namely, the energy efficiency of their processors, the high number of rather small processor cores which allows to manage computing resources more flexible, and the fact that ARM's licensing model allows customers to also modify the processor designs. At Amazon's AWS, already 50% of the newly deployed cloud servers are ARM-based (Anandtech, 2021) and just recently, Oracle announced that ARM server will be offered on the Oracle Cloud (Magouryk, 2021).

ARM Confidential Compute Architecture. In 2021, together with the new ARMv9 processor architecture, ARM announced a novel TEE architecture for their server processors with the name ARM Confidential Compute Architecture (CCA). The goals of CCA are aligned with those of AMD SEV and Intel TDX, i.e., CCA aims to protect sensitive cloud workloads in isolated VMs, even in the presence of a malicious cloud provider. Similarly to SGX and SEV, CCA provides a transparent memory encryption for the isolated VMs (called *Realms*) by introducing a Memory Protection Engine (MPE) in front of the memory controller. CCA is not substituting the older TrustZone security technology available on the majority of all ARM processors. Instead, CCA implements its Realms in an orthogonal security state of the processor. Moreover, the newly introduced hardware modifications also enable TrustZone to implement a more dynamic memory management.

3.1.4 Hardware Security Modules

HSMs (*Hardware Security Module (HSM) n.d.*) are physical computing devices that safeguard and manage digital keys for strong authentication and provide crypto-processing. These modules traditionally come in the form of a plug-in card or an external device that attaches directly to a computer or network server. HSMs mainly address an adversary that aims to physically tamper with the hardware with the aim of gaining access to the data residing therein. This is why HSMs are typically certified to recognized anti-tamper standards such as FIPS 140-2 (*Security Requirements for Cryptographic Modules n.d.*). Most commercially available HSMs are certified to FIPS 140-2 Level 3.

The Federal Information Processing Standard (FIPS) Publication 140-2 (*FIPS 140-2 n.d.*) and 140-3 (*FIPS 140-3 n.d.*) are U.S. government computer security standards used to accredit HSMs (*Cryptographic Module Validation Program n.d.*). The standard provides four increasing, qualitative levels of security intended to cover a wide range of potential applications and environments. The security requirements cover areas related to the secure design and implementation of an HSM. These areas include HSM specification; ports and interfaces; roles, services, and authentication; finite state model; physical security; operational environment; cryptographic key management; electromagnetic interference/compatibility (EMI/EMC); self-tests; design assurance; and mitigation of other attacks.

Security Levels 1-3. Security Level 1 provides the lowest level of security. Basic security requirements are specified (e.g., at least one Approved algorithm or Approved security function shall be used). No specific physical security mechanisms are required beyond the basic requirement for production-grade components. Level 2 improves upon the physical security mechanisms of Security Level 1 by requiring features that show evidence of tampering, including tamper-evident coatings or seals that must be broken to attain physical access to cryptographic keys and critical security parameters (CSPs) within the module, or pick-resistant locks on covers or doors to protect against unauthorized physical access. Level 3 attempts to prevent the intruder from gaining access to CSPs held within the HSM. Physical security mechanisms

are meant to have a higher probability of detecting and responding to attempts at physical access, use or modification.

Services such as AWS CloudHSM ([AWS CloudHSM n.d.](#)), Google Cloud Key Management ([Google Cloud Key Management n.d.](#)), and Azure Key Vault ([Microsoft Azure Key Vault n.d.](#)) provide NIST FIPS 140-2 Level 3 certified Hardware Security Modules as a service in the cloud to manage customer secrets (keys, credentials etc).

Unfortunately traditional HSMs are designed mainly for key management, cannot run arbitrary software, and have very limited general-purpose processing capability. This is why deploying HSMs, while useful in protecting encryption keys, simply cannot protect running workloads.

3.1.5 Next Generation HSMs: TEEs with Anti-Tamper Protections.

Fortunately, a new generation of HSM hardware is upcoming that combines the strong anti-tamper security of HSMs with the ability to run arbitrary workloads within the protection boundary. One example is the ENFORCER Anti-Tamper Server ([ENFORCER Server n.d.](#)), the first NIST FIPS 140-2 Level 4 certified server and high-performance next generation HSM that can run off-the-shelf x86 software. This server provides both standard HSM cryptographic accelerator functionality but can also *execute arbitrary computation tasks with privacy, software and data confidentiality, and provide remote hardware and software stack integrity attestation.*

FIPS Security Level 4 provides a higher level of security than that of a typical Level 3 HSM. At this level, physical security mechanisms provide a complete set of reactive envelopes of protection around the HSM, detecting and responding to all unauthorized attempts at physical access. Penetration of the HSM enclosure results in the immediate zeroization of all sensitive information. Security Level 4 is essential for operation in physically unprotected environments. Security Level 4 also protects against a security compromise due to environmental conditions or fluctuations outside of normal operating ranges for voltage and temperature.

A number of ongoing efforts aim at integrating such next generation HSMs into cloud infrastructures to *provide TEEs with anti-tamper*

protections. If successful, they would address many of the challenges associated with traditional TEEs, discussed in the following.

3.1.6 Performance & Functionality

In the context of cloud computing TEE-architectures have to fulfil not only security requirements, which have been discussed in the previous sections, but they also have to fulfil performance and functionality requirements. On the one hand, the performance impact of security measures on the protected workloads must be minimal. On the other hand, the security mechanism must integrate into the cloud management paradigm allowing for flexible and scalable operation of isolated cloud workloads.

Cost of Security For years Intel SGX was the only commercially available TEE architecture that could provide enclaves to application developers and users on commodity platforms. It was designed to isolate and execute small security sensitive parts of an application, like cryptographic function operation on a secret key. Due to this goal setting, SGX enclaves can only utilize a small portion of system memory, called the enclave page cache, which is only 128 MB.¹ This limits the use of SGX enclaves for complex and resource demanding application.

Already before Intel SGX was available solutions were developed to isolate unmodified applications in SGX enclaves (Baumann *et al.*, 2014b). Over the following years further solutions have been proposed aiming to isolate existing workloads, e.g., docker software containers (Arnautov *et al.*, 2016) in SGX enclaves (Tsai *et al.*, 2014). These efforts show the demand for isolation solutions that allows the isolation of existing server and cloud workloads. However, the design of SGX requires costly and complex software extensions and abstractions to achieve the desired functionalities. In combination with its memory limitations this leads to performance overheads of such solutions, which prohibit their use in most practical applications (Baumann *et al.*, 2014b; Arnautov *et al.*, 2016; Tsai *et al.*, 2014).

¹Recently Intel has released now SGX-enabled systems that overcome this limitation, however, they are not widely available.

Scalable and Flexible Security AMD SEV – as well as the upcoming Intel TDX and Arm CCA – overcomes central limitations of Intel SGX, providing more memory for enclaves and enabling the isolation of typical cloud workloads, in particular virtual machines (VMs), by design. However, other cloud workloads, such as software containers with their advantages of swift creation and low resource usage, are not directly supported. Furthermore, the secure management and orchestration of secure enclaves in the cloud is an unsolved problem in practice (Vaucher *et al.*, 2018), limiting the scalability of isolation architectures in the context of cloud.

Many emerging cloud applications, in particular in the context of machine learning (ML), demand extensive computation power that cannot be satisfied by today's CPUs but requires accelerators like FPGAs, GPGPUs or TPUs. Current TEE solutions are exclusively available to application execution only on CPUs, and hence, they have severe performance disadvantages when used to protect ML application (Brasser *et al.*, 2018). The secure integration and combination of accelerators with CPU-enclaves is an active field of research (Kida *et al.*, 2020), yet practical solutions are not available in deployed systems.

3.2 Open Challenges and Future Research Direction

Only recently TEE-architectures designed to isolate cloud workloads have been developed, such as ADM SEV, Intel TDX or Arm CCA. However, those security architectures are also tailored to specific use-cases, i.e., isolation of VMs. Flexible TEE-architectures that support various enclave types, as have been developed by research (Bahmani *et al.*, 2021), are missing in practice. Furthermore, the secure management and orchestration of secure enclaves is an open challenge for enclave-isolated workloads in the cloud.

4

Side-Channels

Multi-tenancy is a cornerstone of most sustainable cloud services. To make their cloud business cost-effective, cloud providers typically share computing resources among multiple cloud tenants. The rationale behind such a multi-tenant cloud business model is an assumption that not all cloud workloads would fully utilize the claimed computing resources at all time. Therefore, the use of these resources can be maximized through over-provisioning and the cost of maintaining the cloud data centers can be amortized. For example, cloud providers like Google Cloud, Amazon Web Services (AWS) and Microsoft Azure usually schedule multiple virtual machines (VMs) on the same cloud physical machine, with the total virtual CPU time and virtual memory consumption of all these VMs larger than what is physically available.

While multi-tenancy makes the cloud business profitable, it is also the root cause of a variety of side channels. Because of cloud multi-tenancy, a cloud tenant may have its computation co-located with its competitors or a malicious tenant targeting random victims. The unrestricted sharing of computing resources among these mutually distrustful bedfellows creates new attack vectors that do not exist in tradition enterprise networks. Side channel is one such attack vector that

allows information leakage from isolated security domains. Side channel leaks confidential data through indirect inferences. While secrets of the isolated security domain cannot be directly learned by an external adversary, the use of such data during the execution may exhibit side effects that are observable from the outside. Such side effects may include power consumption, electromagnetic emission, execution time, acoustics, etc. Studies have shown that external adversaries may make inference on the observed side effects to *guess* the value of the secret data with very high precision.

Of particular interest in the cloud context are the side channels exploitable by software programs remotely controlled by the adversary. As physical accesses to the cloud data centers are unlikely, remote accesses with the highest privilege achievable by a cloud tenant become a reasonable assumption in the cloud threat model. In the cloud context, not all side effects can be exploited as side channels, such as electromagnetic emission, which requires special equipment and physical access to the device to carry out the attack. Therefore, side channels in cloud computing can be considered a subset of all side channels in the computing space.

In this section, we provide a literature review of research on both side-channel attacks and side-channel defenses that are relevant in the multi-tenant cloud settings.

4.1 Side-channel Attacks

Side-channel attacks can be categorized by the source of information leakage, which could be either micro-architectural or architectural computing resources. Micro-architectural resources are not directly visible by the software, such as CPU caches, Translation Lookaside Buffers (TLB), Branch Prediction Units (BPU). Architectural resources are outside the CPU package and typically visible to the software. The boundary between the two can be blurry, however. For example, in some cases, resources outside the CPU package, such as DRAM row buffers, are not directly visible to the software, either. In this section, we enumerate a list of computing resources that have been exploited as media of side channels.

4.1.1 CPU Cache

Cache is one of the most commonly used CPU micro-architectural components for constructing side channels. CPU caches hold copies of memory data for more efficient access from the execution units. Intel CPUs have multiple levels of caches. The first level that is closest to the CPU execution units is called the L1 cache, which is further split into the L1 data cache and L1 instruction cache. The second level cache is called the L2 cache, which is typically unified for both instruction and data. The third level cache is also called the Last Level Cache (LLC). While L1 and L2 caches are private to each CPU cores, LLC on Intel processors are shared among all cores in the same CPU package. The size of LLC may range from several megabytes to tens of megabytes, while that of the L1 is in the order of kilobytes. But the access latency of L1 cache is much faster than that of L2 and LLC.

The differences of access latency between CPU cache and memory and the nuance between different levels of CPU caches have been exploited to construct side channels. The very first cache-based side-channel attacks could date back to the mid-1990s, which were designed to crack cryptographic systems such as RSA and DSS in smart cards (Kocher, 1996; Kelsey *et al.*, 1998).

Categories. Generally speaking, cache side-channel attacks can be separated into two categories (Page, 2002):

1. *Time-driven*: In time-driven attacks (Kocher, 1996; Aciğmez *et al.*, 2007; Brumley and Boneh, 2005; Bernstein, 2005; Bonneau and Mironov, 2006; Osvik *et al.*, 2006), the attacker measures the execution time of certain operations of the victim, in order to infer the sensitive operation. As different control flows would result in different execution times, the attacker may utilize the timing information to track the control flows of the victim, which may further leak information of the execution, even the secret keys.
2. *Trace-driven*: In trace-driven attacks (Percival, 2005; Tromer *et al.*, 2010; Gullasch *et al.*, 2011; Osvik *et al.*, 2006; Aciğmez and Koç, 2006), the attacker infers whether certain cache lines or cache sets

are used by the victim by measuring the execution time of the attacker's own operations. Different actions of the victim would lead to a different cache state, which is observable by the attacker through manipulating the cache. Trace-driven attacks are more prevailing than time-driven attacks, since oftentimes it is easier for the attacker to manipulate the shared cache, rather than observe the victim's actions directly.

Researchers have proposed many cache side-channel attack techniques; the major ones include Evict+Time (Osvik *et al.*, 2006) (time-driven), Prime+Probe (Osvik *et al.*, 2006; Percival, 2005; Irazoqui *et al.*, 2015; Liu *et al.*, 2015a) (trace-driven), Flush+Reload (Yarom and Falkner, 2014) (trace-driven), and variations such as Flush+Flush (Gruss *et al.*, 2016; Didier and Maurice, 2021) (trace-driven). They can be used to infer sensitive information via L1 instruction/data cache (Percival, 2005; Tromer *et al.*, 2010) or the LLC (Irazoqui *et al.*, 2015; Liu *et al.*, 2015a; Yarom and Falkner, 2014).

Same-core attacks. Due to the Simultaneous Multi-Threading (SMT) of the modern CPU processors, the execution of different processes running on the same CPU core may be interleaved. Since these processes share the same L1 instruction and data cache, the attacker process may be able to learn sensitive information of the victim running on the same CPU core. There are many existing works (Tsunoo *et al.*, 2003; Brumley and Boneh, 2005; Aciğmez *et al.*, 2005; Percival, 2005; Aciğmez *et al.*, 2007; Osvik *et al.*, 2006; Aciğmez, 2007) that make use of L1 cache information to steal the victim's secret. Tsunoo *et al.* (2003) has demonstrated that a same-core attacker can make use of the L1 data cache to break DES. Percival (2005), Bernstein (2005) and Aciğmez *et al.* (2007) showed that it is also possible to break AES using L1 data cache. Brumley and Boneh (2005) introduced a remote L1 cache side-channel attack that can break RSA of an OpenSSL web server. Aciğmez (2007) showed that L1 instruction cache can also be used to break RSA implemented in OpenSSL. The Prime+Probe attack (Osvik *et al.*, 2006) was also designed for the same-core scenario. In the Prime+Probe attack (Osvik *et al.*, 2006), the attacker first occupies specific cache sets (Prime), wait for the victim to run, then access the same cache sets and

measure the time (Probe). The timing difference can be used to infer whether the victim has accessed the cache sets.

Cross-core attacks. In modern CPU architecture, different CPU cores of the same CPU package often share the last-level cache (LLC) to reduce the communication overheads. Due to this reason, when the attacker and the victim run on different cores of the same package, the attacker can still learn information of the victim by introducing LLC cache contentions with the knowledge of cache inclusiveness, cache replacement policy or cache mapping. The Prime+Probe attack (Osvik *et al.*, 2006) has been extended to the LLC by Liu *et al.* (2015a) and Irazoqui *et al.* (2015). Flush+Reload (Yarom and Falkner, 2014) was designed to work on the LLC; the attacker first flushes the targeted cache line out of the LLC (Flush), wait for a certain period of time, then measure the time of reloading it (Reload) to see if the victim has accessed the cache line.

Cross-VM attacks. In cloud side-channel attacks, the attacker runs the attacker VM, and tries to infer sensitive information of the victim VM. The most important prerequisite of launching cache side-channel attacks is to make sure that the attacker VM and the victim VM are co-located on the same physical machine, i.e., to achieve *co-residency*. Ristenpart *et al.* (2009) proposed the first work to mount the attacker VM to be co-resident with the victim VM on Amazon EC2. They designed a way to detect whether co-residency is achieved between the attacker VM and the victim VM, so that further cache side-channel attacks can be launched. After this work, many defenses have been proposed (Zhang and Reiter, 2013; Zhang *et al.*, 2011b), and leading companies have taken actions to mitigate the threats (e.g., remove the ability to perform co-residency check), but new ways to facilitate co-residency detection continues to be found (Varadarajan *et al.*, 2015; Xu *et al.*, 2015; Bates *et al.*, 2012; Inci *et al.*, 2016).

With the co-residency detection, cross-VM side-channel attacks can be further conducted on the cloud. Zhang *et al.* (2012) demonstrated a cross-VM cache side-channel attack on Xen platform, which used the Prime+Probe technique on the L1 instruction cache. They showed that after co-residency detection, they can perform cross-VM attacks and