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. Jaggard 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

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

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

Jason Nieh
Columbia University
nieh@cs.columbia.edu

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

Radu Sion
Stony Brook University
sion@cs.stonybrook.edu

Yinqian Zhang
Southern University of Science and Technology
yinqianz@acm.org

the essence of knowledge
Boston — Delft

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

Foundations and Trends® in Privacy and Security
Volume 3, Issue 2, 2022

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
- Artifical 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

Cloud Computing Security: Foundations and Research Directions

Anrin Chakraborti[1], Reza Curtmola[2], Jonathan Katz[3], Jason Nieh[4], Ahmad-Reza Sadeghi[5], Radu Sion[6] and Yinqian Zhang[7]

[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: 10.1561/3300000028.

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 costs[1] 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,

[1]But 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 III

Secure Storage

7

Data Access Privacy

Encryption alone is not sufficient to protect the confidentiality of data stored in the cloud since it does not hide metadata e.g., access patterns, timing information etc. Leaking access patterns in particular is a serious threat in client-server scenarios where the storage provider can easily observe and log user *access patterns*.

To see why this is a problem, first consider the following toy example: a user (client) stores an alphabetically-sorted encrypted dictionary of items on an untrusted storage platform. The storage provider observers all client accesses by logging API calls e.g., GET, PUT requests. *If an encrypted keyword is inserted/deleted, the sequence of requests for accessing the particular item as well as for other bookkeeping operations e.g., truncating the dictionary etc. can leak information about the keyword(s) such as the constituent letters etc.* In fact, the observer can make knowledgeable guesses about the exact keywords with varying degrees of accuracy based on information obtained *apriori* about the dictionary contents. Permuting the keywords in the storage does not solve this problem since this still allows attacks through frequency analysis – the "popularity" of a particular keyword leaks how often it is likely to appear in a typical text application.

The following two scenarios demonstrate the real world implications of leaking access patterns to an untrusted storage server.

- Consider a cloud-hosted database containing sensitive information e.g., hospital patient records. Regulatory compliance necessitates storing the database encrypted. However, even with encrypted records, *database reconstruction attacks* such as Grubbs *et al.* (2017) and Falzon *et al.* (2020) are able to infer records by observing query results – common attributes such as names, age, geographical location etc. are frequently queried and the attribute values follow known distributions e.g., an attacker might know that a certain disease is common in people of a certain age group. These attacks mainly leverage query access patterns as well the volume of query results.

- Cloud-hosted services often allow users to perform expensive computation on remote processors. While leveraging trusted execution environments e.g., Intel SGX can ensure that the computation runs in a tamper-proof environment and the results generated are correct, the memory access patterns (even within the enclave's cryptographically-protected memory region) can leak information about the computation (Nayak *et al.*, 2017), and the input data (Yu *et al.*, 2019).

- Access and search pattern leak significant amounts of information for searchable encryption systems (Liu *et al.*, 2014a; Oya and Kerschbaum, 2021).

Intuitively, these attacks succeed because many applications have data-dependent access patterns i.e., the order in which items are accessed by the application, the frequency of access etc. depends on the input. For instance, the memory access patterns of several sorting algorithms (e.g., bubble sort) reveal information about the input to the algorithm. This problem also manifests in more complex algorithms such as graph algorithms (Goodrich and Simons, 2014) and machine learning (Ohrimenko *et al.*, 2016).

7.1 Access Privacy

One way to solve the access privacy problem for encrypted databases is to always access it in its entirety. That is, on every access, each and every item is retrieved from the database, re-encrypted/modified and re-uploaded back. In this way, the server does not learn any information about the item of interest. Obviously, this approach does not scale to large databases usually outsourced to cloud servers. In light of this, several more efficient approaches have been considered. *Private information retrieval* (PIR) (Chor *et al.*, 1998) allows a client to access items from a database without revealing the item of interest. However, PIR is mainly a tool for static databases, or databases that are not updated often. Updating a database with PIR capabilities often entails re-uploading the entire database. Alternatively, the database can be encrypted using a fully (or partially) homomorphic encryption scheme. Then, the database can be accessed and updated server-side by computing on ciphertexts. However, the main drawback is that in order to hide access patterns, the computation must involve *all* items in the database, lest it leaks the item(s) of interest. Despite recent advances in making homomorphic encryption schemes efficient, the total computation required is generally considered far too expensive for real-world deployments.

Oblivious RAM (ORAM) Oblivious RAM protocols provide a more practical alternative to solve the access privacy problem by leveraging only basic cryptographic primitives e.g. symmetric key crypto-systems. An Oblivious RAM (ORAM) protocol allows a client to store and manipulate an array of N blocks on an untrusted server without revealing the data or access patterns. Specifically, the logical array of N blocks is indirectly stored into a specialized back-end data structure on the server, and an ORAM scheme specifies an access protocol that implements each logical access with a sequence of physical accesses to that back-end structure. An ORAM scheme is secure if for any two sequences of logical accesses of the same length, the physical accesses produced by the protocol are computationally indistinguishable. We refer the reader to the seminal work on oblivious RAM by Goldreich and Ostrovsky (1996) for more precise definitions.

Evaluation Metrics Intuitively, based on the informal definition, an ORAM protocol will fetch more items per access than what is actually required. This is to "obfuscate" the actual item that was requested by the client. Furthermore, once an item has been fetched, it needs to be randomly replaced to new a location server-side. This is to prevent the server from linking a future access with previous accesses for the same item. One way to do this securely is to *randomly reshuffle* the database after every access (or a batch of accesses). As the server is untrusted, either the client reshuffles the database, or tasks the server to reshuffle without decrypting the data by leveraging expensive cryptographic primitives e.g., homomorphic encryption. The former introduces communication overheads as a subset of the database has to be downloaded and re-uploaded, while the latter introduces server-side computation overheads. Additionally, the reshuffle mechanism may be interactive and require multiple *rounds* of communication. With these factors in mind, ORAMs are evaluated on the following metrics:

- **Communication Complexity (Bandwidth)** is defined as the total amount of data that a client needs to read and re-upload to the server in order to complete a request. Usually, communication is measured in terms of the number of physical data *blocks* that need to be transferred from storage to access one logical data block. Blocks are usually the same size as memory pages on the client system.

- **Round Complexity** measures the number of round trips required between the client and server in order to complete one logical request. Additional round trips add significant communication delays. Obviously, an efficient ORAM protocol will only require one round of communication per logical request.

- **Server-Side Computation** for ORAMs that employ expensive cryptographic primitives. Expensive computation affects overall performance from the standpoint of latency and the associated dollar costs.

Existing work on ORAMs has largely focused on optimizing one or more of these metrics. In the following, we will highlight the noteworthy constructions and refer the reader to the original works for more details.

7.2 Communication-Efficient ORAMs

The seminal work on ORAMs by Goldreich and Ostrovsky (1996) identified communication-efficiency as the primary optimization criteria. The desired goal is to ensure that communication costs scale sublinearly in the database size. Theoretically, it is possible to design ORAM schemes where communication scales poly-logarithmically with the database size (Goldreich and Ostrovsky, 1996).

The construction by Goldreich and Ostrovsky (popularly known as the "hierarchical ORAM") has $O\left(\log^3 n\right)$ communication overhead and is based on a simple design called the square-root ORAM. The idea is to randomly arrange n blocks (server-side) with a permutation known only to the client. In addition, there is a cache (or originally called shelter) of size $\sqrt{n}$ (may be a user-defined parameter), which may be stored either client-side or on the server. As required, the client accesses blocks from the server and adds the accessed blocks to the cache. Crucially, for each access, the client also scans the entire cache *even if the block is already found in the main storage*.

The security of this scheme is immediately obvious: i) the server does not know the secret permutation and hence cannot correlate blocks that are accessed from the main storage, and ii) the cache holds blocks that have been accessed once and is scanned entirely every access. Once the cache fills up, the combined blocks remaining in the main storage and the cache are reshuffled and rearranged again using a new random permutation. This is the most expensive step of the protocol as it requires rebuilding the entire storage. In fact, for security, the rebuilding has to be done obliviously i.e., the intermediate steps should not reveal the final locations of the blocks. This step is usually performed using an oblivious sorting algorithm. The sorting in the original construction is performed using a sorting network which has a communication overhead of $O\left(n \log^3 n\right)$. Overall, since the reshuffling needs to be performed only when the cache fills up after $\sqrt{n}$ accesses, the communication cost of the protocol is $\mathrm{O}\left(\sqrt{n}\log^3 n\right)$. We note that although more efficient oblivious sorting mechanisms exists (Shi, 2020) with communication cost $\mathrm{O}(n\log n)$, replacing the expensive sorting network still does not remove the cost dependence on $\sqrt{n}$ in the original construction.

To overcome this dependence, Goldreich and Ostrovsky proposed a construction that essentially amortizes the level reconstruction cost. The ORAM organizes data on the server-side in a hierarchy of levels. The i-th level holds 4^i blocks and the $(i+1)$-th level (which can hold 4^{i+1} blocks) is the cache for the i-th level. Conceptually, the top level is an append log. On every read/write, the block that is accessed is re-encrypted and placed in the top level. Obviously, the top level overflows after a fixed number of accesses. At this stage, its constituent blocks are flushed and uniformly randomly placed in the second level. This process is generalized across all the levels and the ORAM has $\mathrm{O}(\log n)$ levels. The hierarchical construction has a communication complexity of $\mathrm{O}\left(\log^3 n\right)$ amortized over all accesses.

Improvements Several subsequent works have addressed the high communication complexity of the original hierarchical construction, while retaining the overall structure. Williams and Sion (2008) presented a construction with amortized communication complexity $\mathrm{O}\left(\log^2 n\right)$ under the assumption that the client has at least $\mathrm{O}(\sqrt{n})$ dedicated storage to perform the reshuffles using an oblivious version of the merge sort algorithm. Subsequently, Williams *et al.* (2008) presented an ORAM with more efficient search by storing per-level encrypted bloom filters.

Under assumptions of constant client storage, Pinkas and Reinman (2010) used *cuckoo hashing* and randomized shell sort over the original Goldreich and Ostrovsky solution and achieve an amortized communication complexity of $\mathrm{O}\left(\log^2 n\right)$ Pinkas and Reinman (2010) uses. However, Goodrich and Mitzenmacher (2011) highlight a leak in the construction and provides an alternate construction with amortized communication complexity of $\mathrm{O}(\log n)$ with the assumption of $\mathrm{O}\left(n^{1/r}\right)$ client storage with $r > 1$.

Deamortization One major drawback of the hierarchical ORAM constructions is that client queries need to wait for the duration of a reshuffle, and this is especially impractical for the larger levels. De-amortized constructions allow queries and reshuffles to proceed together and thus eliminate clients waiting for reshuffles after a level overflow. Goodrich *et al.* (2011) showed how to de-amortize the original square root solution and hierarchical solution and achieve a worst-case complexity of $\mathrm{O}(\log n)$ in the presence of $\mathrm{O}\left(n^{1/r}\right)$ client side storage where $r > 1$. Kushilevitz *et al.* (2012) used cuckoo hashing and rotating buffers to provide a de-amortized construction with a worst-case communication complexity of $\mathrm{O}\left(\frac{\log^2 n}{\log\log n}\right)$. PD-ORAM (Williams *et al.*, 2012) is a de-amortized hierarchical ORAM where level reconstructions are performed in the background while allowing queries to proceed simultaneously. This is achieved by keeping two copies of the data: a read-only copy for the queries and a writable copy where new levels are constructed. Level reconstruction starts as soon as a level is created. To ensure that a new level is available on demand when required for the next round of queries, the level construction is synchronized with the queries by tracking the progress of the reshuffle.

7.2.1 Tree-Based ORAMs

The high worst case costs of hierarchical ORAMs makes them impractical and while deamortized construction fare better in this regard, they often introduce additional overheads e.g., increased storage costs. Tree-based ORAMs provide a more viable alternative to hierarchical ORAMs since they are naturally un-amortized i.e., the worst-case query cost is equal to the average cost. Tree-based ORAMs organize the database blocks in the form of a binary (or ternary) tree. Each node of the tree (denoted as a bucket) contains a fixed number of blocks (which can be real or dummy). Blocks are stored along unique randomly selected paths. To track the location of blocks in tree (the corresponding path), a *position map* associates blocks identifiers. e.g., logical addresses to path identifiers e.g., leaf labels. Once a block is stored along some path, the ORAM maintains the following invariant: *A block mapped to a path resides either in any one of the buckets on the path from the root* *to the corresponding leaf, or in a secure storage.* Due to the random association of blocks to paths, writes may fail when all the buckets along the path are occupied up to capacity. In this case the block needs to be stored temporarily in a secure storage, called the *stash*, which is probabilistically bounded in size, and is usually stored client-side.

Shi *et al.* (2011a) presented the first tree-based ORAM with worst-case communication cost of $\mathrm{O}\left(\log^3 n\right)$. Subsequently, Gentry *et al.* (2013) improved the communication costs of the construction by a constant factor. The major breakthrough in tree-based ORAM designs is due to Stefanov *et al.* (2013), in the form of a construction called Path ORAM. Path ORAM achieves $\mathrm{O}(\log n)$ communication costs when the client can spare $\mathrm{O}(n)$ local storage, and $\mathrm{O}\left(\log^2 n\right)$ otherwise. In fact, under certain assumptions (e.g., non-uniform server-side block sizes), Path ORAM can still achieve $\mathrm{O}(\log n)$ communication costs. This matches the known lower bound on communication costs. Subsequently, Ren *et al.* (2015) and Wang *et al.* (2015a) have improved on the practical overheads of Path ORAM.

7.3 Round-Trip Efficient ORAMs

Optimizing round-trips for ORAM protocols is as critical for performance as the overall communication since multiple round-trips to fetch data leads to high latency of access. Unfortunately, none of the aforementioned communication-efficient constructions optimize round-trips. There are two notable constructions that address this problem. SR-ORAM (Williams and Sion, 2012) is a constant round ORAM requiring two round trips with overall communication complexity of $\mathrm{O}\left(\log^2 n \log\log n\right)$. Since, SR-ORAM follows a hierarchical construction, the worst case complexity is $\Omega(n)$. TWORAM (Garg *et al.*, 2016) overcomes this problem; it features a worst-case communication complexity of $\mathrm{O}\left(\log^3 n\right)$ and performs accesses in two rounds. Another notable construction is Bucket ORAM (Fletcher *et al.*, 2015) which features single round-trip accesses and communication complexity of $\mathrm{O}(\log n)$ under certain block size assumptions.

7.4 Compute-Efficient ORAMs

A straightforward way to make ORAM protocols more communication efficient is by leveraging server-side computation. If the server could compute on the data without learning the contents, then the communication burden can be reduced as the server only returns the data block required. A line of work explores this trade-off in communication and computation assuming different server-side compute capabilities.

A version of Ring ORAM (Ren *et al.*, 2015) achieves $O(1)$ communication cost for fetching a block from the server under the assumption that the server can execute XORs over the data blocks before returning them to the client. The overall complexity of the construction is however $O\left(\log^2 n\right)$ due to other necessary bookkeeping operations. Onion ORAM (Devadas *et al.*, 2016) has a communication complexity of $O(B)$ where B is the block size of the ORAM. The construction may use either additively homomorphic encryption (AHE) or somewhat homomorphic encryption scheme (SWHE) with different trade-off; see Devadas *et al.* (2016) for more details. Recently, Chen *et al.* (2019b) proposed Onion Ring ORAM which makes practical improvements to the construction. An alternate line of work assumes multiple servers to aid the computation. One notable example of this line of work is $\mathsf{S^3ORAM}$ (Hoang *et al.*, 2017) utilizing secret sharing as the underlying primitive.

7.5 Other Practical Considerations

7.5.1 Parallel Access

All aforementioned ORAMs are designed for single-client deployments, that is at any point in time, there is a single-client performing accesses to the ORAM. This naturally ensures consistency and privacy. However, in this setting, clients experience unreasonably long wait times making the schemes impractical.

Boyle *et al.* first introduced an oblivious parallel RAM (OPRAM) construction assuming inter-client communication for synchronization (Boyle *et al.*, 2016). Clients coordinate with each other through an *oblivious aggregation* operation and prevent simultaneous queries for the same block. For colliding client accesses, only *one representative* client queries for the required item while all other clients query for dummy items. The *representative* client then communicates the read item to all other colliding clients through an *oblivious multi-cast* operation. Subsequent works (Chan *et al.*, 2017a; Nayak and Katz, 2016; Chen *et al.*, 2016; Chan *et al.*, 2017b; Hubert Chan and Shi, 2017) have optimized Parallel RAMs matching the overhead of a sequential ORAM construction.

TaoStore (Sahin *et al.*, 2016) takes a different approach towards building a parallel ORAM. The construction introduces a trusted proxy such that all client queries are redirected to the trusted proxy which then queries for the corresponding paths from the PathORAM data tree. Further, the proxy runs a secure scheduler to ensure that the multiple path reads do not overlap and leak correlations in the underlying queries. TaoStore achieves a significant increase in throughput but can support only a limited number of parallel clients before the throughput plateaus due to the proxy's bandwidth constraints.

ConcurORAM (Chakraborti and Sion, 2019) is a parallel ORAM construction which overcomes the bandwidth limitations of TaoStore, and reduces the assumption footprint by removing the need for a trusted proxy and inter-client communication. The construction is aided by several auxilliary data structures that allow queries to proceed in the background with full privacy guarantees without blocking other queries.

7.5.2 Write-Only Privacy

Full ORAM privacy is often unnecessary for practical settings. In several data outsourcing scenarios, it is enough to protect the privacy of write operations. A notable example of this is secure data backup on cloud services like DropBox (Aviv *et al.*, 2017). This privacy definition is satisfied by a class of ORAMs called write-only ORAMs. Li and Datta proposed the first write-only ORAM scheme with an amortized write complexity of $O(B \times \log n)$ where B is the block size of the ORAM and n is the total number of blocks (Li and Datta, 2017). However, the construction suffers from poor (linear in the database size) read complexity. Hive (Blass *et al.*, 2014) is a write-only ORAM scheme with constant read complexity. It maps data from a logical address space uniformly randomly to the physical blocks on the underlying device. The construction requires a $O(\log n)$-sized stash. DetWoORAM (Roche *et al.*, 2017) overcomes the requirement of a stash and achieves $O(\log n)$ read complexity and $O(1)$ write complexity.

7.5.3 Range ORAMs

A new ORAM variant, namely Range ORAM, was recently proposed by Asharov *et al.* (2019). Unlike traditional ORAMs optimized for single-block accesses, Range ORAMs are optimized for efficiently accessing ranges of blocks. This notion is especially useful when considering the fact that typical filesystems deployed on top of ORAMs usually access contiguous blocks at once e.g., when reading/writing a file. The efficiency goal for Range ORAMs is to ensure that range accesses can be performed with minimal number of disk seeks across the storage device. This is in contrast to traditional ORAMs which randomly place blocks (belonging to the same file) all across the device making file accesses inefficient on high latency drives like HDDs. As a security trade-off range ORAM reveal the sizes of the ranges accessed; see Asharov *et al.* (2019) for more details.

Asharov *et al.* (2019) presented a construction with $O\left(r \cdot \log^3 n\right)$ communication complexity (amortized) to access r contiguous blocks. The number of seeks required is $O\left(\log^3 n \cdot (\log \log n)^2\right)$ (notice that the number of seeks is independent of r). Chakraborti *et al.* (2019) improved this result by providing an unamortized construction with $O\left(r \cdot \log^2 n\right)$ communication complexity and requiring $O\left(\log^2 n\right)$ seeks.

7.5.4 Hardware-Assisted ORAMs

Oblivious RAM protocols have been used in conjunction with trusted execution environments (TEEs) to design systems with access privacy. ZeroTrace (Sasy *et al.*, 2018) combines ORAMs and Intel SGX, and builds a block-level memory controller that provides oblivious execution against software adversaries. Other noteworthy examples include databases with oblivious query capabilities (Eskandarian and Zaharia, 2019; Hoang *et al.*, 2018) and oblivious file systems (Ahmad *et al.*, 2018). Typically in these systems, the ORAM logic runs securely in a SGX enclave and the data is hosted on an untrusted storage backend. In this way, the expensive bookkeeping operations are performed by the enclave-hosted trusted logic without any client intervention thereby reducing overall communication. The controller also receives and serves requests from the client; a secure communication channel between the client and the enclave ensures that the block requests remain hidden to the server.

7.5.5 Future Research Directions

Although there is a large volume of work dedicated to optimizing ORAMs for clouds, the state of the art is still impractical for real-world deployments. Firstly, the communication costs are still too high. Patel *et al.* (2018) and Asharov *et al.* (2020) have made significant strides in this direction by achieving the known communication lower bound. However, these constructions are mainly of theoretical interest as the constants are impractically high. Making these constructions practical, while keeping in mind the aforementioned performance metrics (e.g., round trips, parallelism), encourage more research in this direction.

Secondly, ORAMs are not cost-effective. The high dollar costs of employing ORAMs often outweigh the cost advantages of outsourcing data to a public cloud (Bindschaedler *et al.*, 2015). This is a largely overlooked drawback of existing protocols which needs to be further investigated. The costs are due to communication and storage overheads. Interestingly, cloud services often price communication asymmetrically: uploads are costlier than downloads. Therefore, building ORAMs that exploit this asymmetry (e.g., lower upload costs for higher download costs) is an interesting research direction. ORAM constructions also come with significant storage overheads: all aforementioned constructions require at least $2\times$ the storage, as that required for the raw database. Exploring storage-efficient protocols is an important consideration for future research.

Finally, for real-world deployments it is important to consider actively malicious adversaries i.e., cloud servers who may modify data or replay old data to the clients. This not only introduces integrity/consistency issues but also impacts privacy. While in a single client scenario, this problem may be solved by integrity-preserving mechanisms (Ren *et al.*, 2013), the problem is significantly amplified in multi-client scenarios. When considering a setting where even the clients can be malicious, Maffei *et al.* (2017) showed that to ensure security the server-side computation required is $\Omega(n)$, that is the server must touch all the items in the database for every access. In this setting, a scheme is presented with communication complexity of $\mathrm{O}(\sqrt{n})$. The lower bound on the server-side compute costs only holds in a single-server setting. Hoang *et al.* (2020) recently presented a construction in a multi-server setting with $\mathrm{O}(1)$ client-server communication complexity and $\mathrm{O}(\log n)$ server-server communication complexity. The construction builds on $\mathsf{S}^3\mathsf{ORAM}$ (Hoang *et al.*, 2017) and adapts it for a malicious server(s) setting. Future work in this direction may explore new constructions in both the single-server and multi-server settings with lower communication complexities.

8

Provable Data Possession

The increasing popularity of third-party cloud storage services in recent years has brought with it numerous advantages, such as reduced cost, the ability to access the data from anywhere, and the ability to easily share data. These benefits however, did not come without challenges, especially from a security and privacy point of view. Due to trust concerns in the third-party cloud storage provider, security and privacy have been identified among the main challenges that hamper data migration to/from a cloud environment.

Unfortunately, none of the cloud storage services offered verifiable guarantees with regard to the integrity and long-term reliability of the stored data. Basically, in the cloud storage commercial landscape, if data is lost, the best a data owner can hope for is compensation proportional with the size of the data (if any), which may be orders of magnitude away from the actual value of the data.

Circa 2007, Ateniese *et al.* (2007; 2011) introduced a new framework for remote data integrity checking using provable data possession (PDP). In this model, the storage server is not trusted to store the data and may try to convince the client (data owner) that it possesses (i.e., stores) the data even if the data is totally or partially corrupted. Protection against corruption of a large portion of the data is necessary in order to handle servers that discard a significant fraction of the data. This applies to servers that are financially motivated to sell the same storage resource to multiple clients. Protection against corruption of a small portion of the data is necessary in order to handle servers that try to hide data loss incidents. This applies to servers that wish to preserve their reputation. Data loss incidents may be accidental (e.g., management errors or hardware failures) or malicious (e.g., insider attacks).

Remote data integrity checking (RDIC) allows an auditor to challenge a remote server to provide a *proof of data possession* in order to validate that the server possesses the data that were originally stored by a client. An RDIC scheme seeks to provide a *data possession guarantee*.

Requirements. Conforming to an outsourced storage relationship, the client (i.e., data owner) should only be required to store a small, ideally constant, piece of metadata.

Oftentimes, cloud storage presents unique performance demands. Given that file data are large and are stored at remote sites, accessing an entire file is expensive in I/O costs to the storage server and in transmitting the file across a network. Reading an entire archive, even periodically, greatly limits the scalability of network stores. Furthermore, I/O incurred to establish data possession interferes with on-demand bandwidth to store and retrieve data. As such, clients need to be able to verify that a server has retained file data without retrieving the data from the server and without having the server access the entire file.

A scheme for auditing remote data should be both *lightweight* and *robust*. Lightweight means that it does not unduly burden the cloud storage provider (CSP); this includes both overhead (i.e., computation and I/O) at the CSP and communication between the CSP and the auditor. This goal can be achieved by relying on *spot checking*, in which the auditor randomly samples small portions of the data and checks their integrity, thus minimizing the I/O at the CSP. Spot checking allows the client to detect if a fraction of the data stored at the server has been corrupted, but it cannot detect corruption of small parts of the data (e.g., 1 byte). Robust means that the auditing scheme incorporates mechanisms for mitigating arbitrary amounts of data corruption. Protecting against large corruptions ensures the CSP has committed the contracted storage resources. Little space can be reclaimed undetectably, making it unattractive to delete data to save on storage costs or sell the same storage multiple times. Protecting against small corruptions protects the data itself, not just the storage resource. Many data have value well beyond their storage costs, making attacks that corrupt small amounts of data practical. For example, modifying a single bit may destroy an encrypted file or invalidate authentication information.

8.1 Prior Approaches

Before the PDP model, several other mechanisms had been proposed that do not meet the above requirements for remote data integrity checking. Some schemes (Golle *et al.*, 2002) provide a weaker guarantee by enforcing storage complexity: The server has to store an amount of data at least as large as the client's data, but not necessarily the same exact data. Moreover, most previous techniques require the server to access the entire file, which is not feasible when dealing with large amounts of data, or require storage on the client linear with the size of the data, which does not conform with the notion of storage outsourcing (Deswarte *et al.*, 2003; Sebe *et al.*, 2004; Filho and Baretto, 2006; Shah *et al.*, 2007). A notable exception is the work of Schwarz and Miller (2006), which meets most of the requirements for proving data possession, but provides a less formal security analysis. This scheme relies on a special construct called an "algebraic signature," which is a function that fingerprints a block and has the property that the signature of the parity block equals the parity of the signatures of the data blocks.

8.2 Provable Data Possession

A Provable Data Possession (PDP) protocol checks that an outsourced storage site retains a file, which consists of n blocks. The client C (data owner) pre-processes the file, generating a small piece of metadata that is stored locally, transmits the file to the server S, and may delete its local copy. The server stores the file and responds to challenges issued by the client. Storage at the server is $\Omega(n)$ and storage at the client is O(1), conforming to an outsourced storage relationship.

As part of pre-processing, the client may alter the file to be stored at the server. The client may encrypt, encode or expand the file, or may include additional metadata to be stored at the server.

At a later time, an auditor issues a challenge to the server to establish that the server has retained the file. The auditor requests that the server compute a function of the stored file, which it sends back to the client. Using its local metadata, the auditor verifies the response.

For ease of exposition, the client (data owner) is assumed to be the same entity as the auditor. However, the model can be easily extended to a setting where these two may be separate entities (e.g., if business requirements require separation, or if data privacy is a concern and the auditor should not have access to the plain data (Shah *et al.*, 2008).

Ateniese *et al.* (2007; 2011) proposed two PDP schemes which rely on *homomorphic verifiable tags*. The client pre-computes tags for each block of a file and then stores the file and its tags with a server. At a later time, the client can verify that the server possesses the file by generating a random challenge against a randomly selected set of file blocks. The server retrieves the queried blocks and their corresponding tags, using them to generate a proof of possession. The client is thus convinced of data possession, without actually having to retrieve file blocks. Because of the homomorphic property, tags computed for multiple file blocks can be combined into a single value, and so a challenge uses $O(1)$ network bandwidth.

These PDP schemes sample the server's storage, accessing a random subset of blocks. Sampling proves data possession with high probability based on accessing a few blocks in the file, which radically alters the performance of proving data possession.

Achieving robustness. An RDIC scheme can be enhanced to provide robustness by using forward error-correcting codes (FECs). Attacks that corrupt small amounts of data do no damage, because the corrupted data may be recovered by the FEC. Attacks that do unrecoverable amounts of damage are easily detected because they must corrupt many blocks of data to overcome the redundancy.

Ateniese *et al.* (2011) propose a generic transformation that encodes a file using FECs in order to add robustness to any RDIC scheme that relies on spot checking (Curtmola *et al.*, 2008a). A robust RDIC scheme provides protection against arbitrary small amounts of data corruption.

Additional features. The PDP schemes introduced by Ateniese *et al.* (2007; 2011) provide several additional useful features. First, they provide *data format independence*, meaning they put no restriction on the format of the data. In particular, files stored at the server do not have to be encrypted. This feature is relevant since PDP schemes may have a significant impact when used with large public repositories (e.g., digital libraries, astronomy/medical/legal repositories, archives etc.) Second, they put no restriction on the number of times the client can challenge the server to prove data possession. Third, they pioneer the notion of *public verifiability*, which allows anyone, not just the data owner, to challenge the server for data possession. For example, an independent third-party auditor can verify possession of the data. The advantages of having public verifiability are akin to those of public-key over symmetric-key cryptography.

8.3 Dynamic Provable Data Possession

The original PDP model was introduced in the context of static data, i.e., data that is not modified after being stored initially. This matches a variety of application scenarios that fall under the umbrella of archival storage. The model was shown to also securely support the append operation (i.e., data blocks are appended at the end of the file), which covers application scenarios such as version control systems (Chen and Curtmola, 2014). The model was subsequently extended by Erway *et al.* (2009; 2015) to support the full range of dynamic updates to the stored data – i.e., the client can insert, modify, or delete stored data blocks – while maintaining data possession guarantees. Dynamic PDP (DPDP) can thus cover a wider range of cloud computing scenarios, including file storage, database services, and peer-to-peer storage. The proposed DPDP schemes are based on a new variant of authenticated dictionaries which permit efficient membership queries (i.e., a rankbased authenticated dictionary built over a skip list). Different from a static PDP scheme, for a dynamic PDP scheme to be efficient, it must not include order information in the tags, since otherwise an update may cause all tags to be updated. From a performance perspective, the most important cost introduced by a dynamic PDP scheme compared to a static PDP scheme is that the size of a data possession proof grows from $O(1)$ to $O(\log n)$, where n is the number of file blocks.

Subsequently, Etemad and Küpçü (2020) show a general framework for constructing DPDP schemes that encompass existing DPDP-like schemes as different instantiations.

8.4 Proofs of Retrievability

Simultaneously with PDP, Juels and Kaliski (2007) have introduced a similar notion, that of *proof of retrievability* (PoR), which allows a client to be convinced that it can retrieve a file previously stored at the server. This PoR scheme uses disguised blocks (called sentinels) hidden among regular file blocks in order to detect data corruption by the server. Although comparable in scope with PDP, this PoR scheme can only be applied to encrypted files and can handle a limited number of queries, which has to be fixed a priori. At a high level, a PoR scheme provides similar guarantees as an RDIC scheme (i.e., a PDP scheme that incorporates robustness to provide protection against small amounts of data corruption). Shacham and Waters (2008; 2013) improve the PoR state of the art by introducing the most-widely-accepted definitions for PoR-type schemes and giving two PoR protocols based on homomorphic authenticators. The first is based on bilinear maps and achieves public verifiability, whereas the second is based on pseudo-random functions, more efficient, but is only privately verifiable. Erway *et al.* (2015, Section 7.3) provide a detailed comparison of PDP and PoR schemes.

Although initially proposed for a static setting, PoR schemes were subsequently extended to a dynamic setting (i.e., the stored data can be updated in time). Initial dynamic PoR schemes were mostly of theoretical interest: Stefanov *et al.* (2012) (due to imposing a large amount of client storage and a large audit cost) and Cash *et al.* (2013b) (due to imposing large audit overhead). Shi *et al.* (2013) proposed the first practical dynamic PoR scheme that achieves comparable communication overhead and client-side computation with a standard Merkle hash tree. Like prior PoR and RDIC schemes, this scheme uses FEC codes (erasure codes more precisely) to achieve protection against small data corruptions, but ensures that data updates can be done efficiently by maintaining on the server side an erasure-coded hierarchical log structure that contains recently written blocks. This structure needs a special erasure coding scheme that can be incrementally built over time. Due to the use of this additional metadata, the actual erasure-encoded data only needs to be rebuilt every n write operations, where n is the number of file blocks.

8.5 Towards Auditing Distributed Storage Systems

In many practical cloud storage systems, data should be replicated in order to deal with data loss accidents. Preferably, the replicas should be stored in different geographical locations, in order to ensure failure independence. Replication is a useful mechanism in the context of proving data possession by a cloud storage provider. Whereas techniques such as PDP and PoR are useful to verify remotely the integrity of a single replica, they provide limited value when that single replica is irreparably damaged.

When data is replicated at multiple storage servers, an auditor can execute independently data possession protocols with each of the storage servers. In case any of the replicas is found to be damaged, the data owner can use the healthy replicas to restore the desired level of data replication.

Establishing a guarantee that t replicas of a file are in fact stored by a set of storage servers becomes more challenging when we assume that the storage servers can behave fully malicious (i.e., can collude with each other). The servers that appear to be storing multiple replicas may be in fact storing only a single copy of the data. In general, this can be done by redirecting and forwarding challenges from the multiple sites to the single site that stores the data. In this way, clients (data owners) remain unaware of the reduction in the availability and durability of data that results from the loss of replicas. Even if the client initially stores replicas on servers in different geographic locations, the servers can then move all the replicas to one location and access them from that location on demand. Such a system is not more reliable than a single-replica system, even though it leads the client to believe so.

Replication systems that rely on untrusted servers have another generic limitation. To prove data availability, the servers can produce replicas on demand upon a client's challenge; however, this does not prove that the actual replicas are stored at all times. For example, malicious servers may choose to introduce dependencies among replicas, by encrypting the replicas before storing them. Replicas can then be decrypted and served on demand whenever they are requested by clients. By storing the encryption key in a single location, the malicious servers can effectively negate any reliability improvements achieved by storing the replicas at different locations. Loss of the encryption key means loss of all the replicas.

Given these generic limitations of replication systems that rely on fully dishonest servers, Curtmola *et al.* (2008b) consider a model in which storage servers are rational and economically motivated. In this context, cheating is meaningful only if it cannot be detected and if it achieves some economic benefit (e.g., using less storage than required by the contract). Such an adversarial model is reasonable and captures many practical settings in which malicious servers will not cheat and risk their reputation, unless they can achieve a clear financial gain. Curtmola *et al.* (2007; 2011) extend PDP to apply to multiple replicas so that a client that initially stores t replicas can later receive a guarantee that the storage system can produce t replicas, each of which can be used to reconstruct the original file data. A replica comprises the original file data masked with randomness generated by a pseudo-random function (PRF). As each replica uses a different PRF, replicas cannot be compared or compressed with respect to each other. The homomorphic verification tags of PDP are modified so that a single set of tags can be used to verify any number of replicas. These tags need to be generated a single time against the original file data. Thus, replica creation is efficient and incremental; it consists of unmasking an existing replica and re-masking it with new randomness. In fact, the proposed multiple-replica PDP scheme is almost as efficient as a single-replica PDP scheme in all the relevant parameters.

In the context of distributed storage, other solutions have subsequently been proposed, to cover various points in the two-dimensional feature-cost space. For example, Bowers *et al.* (2009) introduced HAIL, a system that stores a file across multiple servers using redundancy. They consider a mobile adversary, which is capable to corrupt all storage servers, although at different moments in time (i.e., the adversary can corrupt any servers, as long as at most a fixed number of servers are corrupted at any one time). To deal with such a strong adversary, HAIL employs a careful interleaving of different types of error-correcting, which exploits both within-server redundancy and cross-server redundancy. At a high level, HAIL can be thought of as extending the RAID concept into the cloud, by spreading redundancy across multiple cloud servers.

Etemad and Küpçü (2013) explore a Dynamic PDP (DPDP) model in the context of a distributed, replicated storage system. Chen *et al.* (2010) propose remote data integrity mechanisms optimized for a setting when data is distributed across multiple storage servers using network coding (Dimakis *et al.*, 2007; Dimakis *et al.*, 2010). Li and Lazos (2020) introduce a mechanism for verifying that a file is redundantly stored across multiple physical storage nodes according to a pre-agreed layout and can, therefore, survive node failures. Leontiadis and Curtmola (2018) seek to deduplicate replicated storage and design a secure storage system that provides users with strong integrity, reliability, and transparency guarantees about data that is outsourced at cloud storage providers. Users store multiple replicas of their data at different storage servers, and the data at each storage server is deduplicated across users.

Bowers *et al.* (2011) proposed RAFT, a mechanism that allows a data owner to check that a storage server has stored a file F across multiple disk drives, so it can support a desired level of fault tolerance (e.g., data can be recovered if any set of t drives has failed). RAFT is designed specifically for data stored on rotational drives, and exploits the performance limitations of such drives as a bounding parameter.

Damgård *et al.* (2019) proposed *proofs of replicated storage*. Such a proof guarantees that a set of servers have reserved the space necessary to store n copies of a file. Previous attempts to achieve a similar guarantee rely on timing assumptions (Protocol Labs, 2017a; Protocol

Labs, 2017b). A replica is encoded using a process that is slow, so that an auditor can distinguish between the time an honest server computes a proof and the time a dishonest server would need to re-encode the file at the time of the challenge. In contrast, Damgard *et al.* propose a construction for proofs of replicated storage that does not rely on timing assumptions. As opposed to time-bounded approaches which rely on a public deterministic encoding function, their approach is to use probabilistic encoding, which makes the re-encoding unfeasible. In addition, they focus on achieving *public verifiability*, which allows anyone (not just the data owner) to play the role of the verifier in an audit protocol. In practical terms, this means that decoding a replica can be done by anyone.

8.6 Remote Data Integrity Checking With Server-side Repair

When a distributed storage system is used in tandem with remote data integrity checking (RDIC), several phases can be distinguished throughout the lifetime of the storage system: Setup, Challenge, and Repair. To outsource a file F, the data owner creates multiple replicas of the file during Setup and stores them at multiple storage servers (one replica per server). During the Challenge phase, the data owner can ask periodically each server to provide a proof that the server's replica has remained intact. If a replica is found corrupt during the Challenge phase, the data owner can take actions to Repair the corrupted replica using data from the healthy replicas, thus restoring the desired redundancy level in the system. The Challenge and Repair phases will alternate over the lifetime of the system.

In cloud storage outsourcing, a data owner stores data in a distributed storage system that consists of multiple cloud storage servers. The storage servers may belong to the same CSP (e.g., Amazon has multiple data centers in different locations), or to different CSPs. The ultimate goal of the data owner is that the data will be retrievable at any point of time in the future. Conforming to this notion of storage outsourcing, the data owner would like to outsource *both the storage and the management* of the data. In other words, after the Setup phase, the data owner should only have to store a small, constant, amount of data and should be involved as little as possible in the maintenance of the data. Minimal involvement in the Challenge phase can be achieved when using an RDIC scheme that has public verifiability. However, traditionally, the Repair phase imposes a significant burden on the data owner, who needs to expend a significant amount of computation and communication. For example, to repair data at a failed server, the data owner needs to first download an amount of data equal to the file size, re-generate the data to be stored at a new server, and then upload this data at a new healthy server (Curtmola *et al.*, 2008b; Bowers *et al.*, 2009). Archival storage deals with large amounts of data (Terabytes or Petabytes) and thus maintaining the health of the data imposes a heavy burden on the data owner.

Chen and Curtmola (2013; 2017) explore a model which minimizes the data owner's involvement in the Repair phase, thus fully realizing the vision of outsourcing both the storage and management of data. During Repair, the data owner simply acts as a *repair coordinator*, which allows the data owner to manage data using a lightweight device. This is in contrast with previous work, which imposes a heavy burden on the data owner during Repair.

The main challenge is how to ensure that the untrusted servers manage the data properly over time (i.e., take necessary actions to maintain the desired level of redundancy when some of the replicas have failed). They consider a new storage system architecture in which each storage server exposes an interface for data manipulation so that the data owner can coordinate the actions of the storage servers in the Repair phase. To repair a faulty server during Repair, the data owner identifies healthy servers and instructs them to collaborate. In this process, most of the communication occurs between the storage servers, and the communication between data owner and storage servers is minimized.

Their approach is based on two insights. First, the replicas stored by the storage servers must be different. Second, to enable server-side repair, the data owner gives the servers both access to the original file and the means to generate new replicas. This will allow the servers to generate a new replica by collaborating between themselves during Repair. However, this approach opens the door to a new attack, in which the servers falsely claim they generate a new replica whenever an existing replica has failed, but in reality they collaborate to only generate a replica *on the fly* during the Challenge phase (this attack is referred to as the *replicate on the fly (ROTF) attack*). To overcome the ROTF attack, the proposed approach is to *make replica creation to be time consuming*. In this way, malicious servers cannot generate replicas on the fly during Challenge without being detected. Two schemes are proposed to generate distinct replicas: The first uses a controllable amount of masking to deal with weaker adversaries, and the second uses a variant of butterfly encoding (Dijk *et al.*, 2012) to create dependencies between each of the replica blocks and multiple original file blocks in order to deal with stronger adversaries.

Towards a similar goal to allow servers to generate new replicas, Armknecht *et al.* (2016) propose Mirror, a PoR-based solution that leverages tunable cryptographic RSA-based puzzles to impose significant resource constraints on the storage servers. As a result, a rational cloud storage provider will be incentivized to correctly store and replicate the client's data or risk detection with high probability otherwise.

8.7 Future Research Directions

Ensuring the integrity and long-term reliability of cloud stored data has been an active research area over the past few years and, considering the security and privacy-sensitive nature of the cloud storage paradigm, will likely continue to attract interest for the foreseeable future.

Despite significant progress and despite the plethora of security guarantees put forth by the academic community, adoption by major cloud storage providers remains an elusive target. Short of native deployment of auditing and data maintenance capabilities by the cloud providers themselves, one can imagine a business model where such services could be offered by a third party auditor running in the same data center where the data is located. This introduces additional concerns, especially when auditing private data, as data owners would need to allow access to their data for the auditor.

The lack of adoption in a commercial setting is a multifaceted problem. Certainly, performance is a significant concern: Providing such strong guarantees as the ones aforementioned in this work could degrade performance. Related to this issue may be the lack of efficient and production level implementations. There are also economic, regulatory and policy reasons. Lack of adoption may seem surprising, because providing such strong guarantees could be seen as a business differentiator. Yet, cloud providers do not seem to have clear economic incentives to provide such strong guarantees, and have focused on more basic security guarantees such as ensuring the privacy and secure sharing of the data.

There are still open problems, especially when trying to achieve simultaneously multiple different guarantees. For example, designing RDIC schemes that are both robust and fully meet data format independence has been challenging. This is because robustness usually imposes encrypting (parts of) the data. As another example, remotely verifying the geographical location of cloud data remains an elusive target, despite some early attempts (Benson *et al.*, 2011; Peterson *et al.*, 2011; Watson *et al.*, 2012; Gondree and Peterson, 2013; Dang *et al.*, 2017) based on time assumptions and distance-bounding protocols.

We conclude by briefly surveying some recent work that may be indicative of the current and future directions in this area. He *et al.* (2020) propose to relax some of the trust assumptions through the use of Intel SGX. Shen *et al.* (2020) propose a protocol that optimizes the communication overhead when data that needs to be audited changes ownership. Leontiadis and Curtmola (2019) study RDIC protocols when applied to compressed data. A user delegates the compression to the cloud in a provably secure way: The user can verify correctness of compression without having to download the entire uncompressed file and check it against the compressed version. Armknecht *et al.* (2021) consider a setting in which third party auditors may be dishonest and data owners can efficiently keep the auditors in check. Chen *et al.* (2021) introduce a decentralized system for proofs of data retrievability and replication which is incentive-compatible and realizes automated auditing atop off-the-shelf blockchain platforms. Ateniese *et al.* (2020) study proofs of storage-time, which enable a verifier to audit that the outsourced data is continuously available to the server during the entire storage period, not only at the time a valid proof is processed.

9

Acknowledgements

Reza Curtmola's work was supported by the NSF through awards 1801430, 2043104, and 1565478. He would like to thank his co-authors without which some of the work referenced in this monograph would not have been possible.

Jason Nieh's work was supported in part by two Amazon Research Awards, a Guggenheim Fellowship, DARPA contract N66001-21-C-4018, and NSF grants CCF-1918400, CNS-2052947, and CCF-2124080.

The work of Ahmad-Reza Sadeghi and his co-contributors Emmanuel Stapf, Patrick Jauernig and Ferdinand Brasser was funded by the German Federal Ministry of Education and Research (BMBF) in the StartUpSecure funding program (16KIS1417).

Radu Sion's work was supported by the NSF through award 2052951, and ONR through award N000142112407.

References

Aciiçmez, O. (2007). "Yet another microarchitectural attack: exploiting I-cache". In: *Proceedings of the 2007 ACM workshop on Computer security architecture*. 11–18.

Aciiçmez, O. and Ç. K. Koç. (2006). "Trace-driven cache attacks on AES". In: *Proceedings of the 8th International Conference on Information and Communications Security (ICICS '06)*.

Aciiçmez, O., W. Schindler, and C. K. Koç. (2005). "Improving Brumley and Boneh timing attack on unprotected SSL implementations". In: *Proceedings of the 12th ACM conference on Computer and communications security (CCS 05)*. 139–146.

Acıiçmez, O., W. Schindler, and Ç. K. Koç. (2007). "Cache based remote timing attack on the AES". In: *Cryptographers' track at the RSA conference*. Springer. 271–286.

Ahmad, A., K. Kim, M. I. Sarfaraz, and B. Lee. (2018). "OBLIVIATE: A Data Oblivious Filesystem for Intel SGX". In: *Proceedings of the 2018 ISOC Network and Distributed System Security Symposium (NDSS 18)*.

Aldaya, A. C., B. B. Brumley, S. ul Hassan, C. P. Garcia, and N. Tuveri. (2019). "Port contention for fun and profit". In: *Proceedings of the 40th IEEE Symposium on Security and Privacy (S&P 19)*. 870–887.

AMD. (2020). "AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More". URL: https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf.

AMD. (2021). "AMD Secure Encryption Virtualization (SEV) Information Disclosure, AMD-SB-1013". URL: https://www.amd.com/en/corporate/product-security/bulletin/amd-sb-1013.

Anandtech. (2021). "Arm Announces Neoverse V1, N2 Platforms & CPUs, CMN-700 Mesh: More Performance, More Cores, More Flexibility". URL: https://www.anandtech.com/show/16640/arm-announces-neoverse-v1-n2-platforms-cpus-cmn700-mesh.

Armknecht, F., L. Barman, J.-M. Bohli, and G. O. Karame. (2016). "Mirror: Enabling Proofs of Data Replication and Retrievability in the Cloud". In: *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association. 1051–1068.

Armknecht, F., J.-M. Bohli, G. Karame, and W. Li. (2021). "Outsourcing Proofs of Retrievability". *IEEE Transactions on Cloud Computing*. 9(1): 286–301.

Arnautov, S., B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. L. Stillwell, D. Goltzsche, D. Eyers, R. Kapitza, P. Pietzuch, and C. Fetzer. (2016). "SCONE: Secure Linux Containers with Intel SGX". In: *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USA: USENIX Association.

Asharov, G., T.-H. Hubert Chan, K. Nayak, R. Pass, L. Ren, and E. Shi. (2019). "Locality-Preserving Oblivious RAM". In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Y. Ishai and V. Rijmen.

Asharov, G., I. Komargodski, W.-K. Lin, K. Nayak, E. Peserico, and E. Shi. (2020). "OptORAMa: Optimal Oblivious RAM". In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by A. Canteaut and Y. Ishai.

Ateniese, G., R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song. (2011). "Remote data checking using provable data possession". *ACM Trans. Inf. Syst. Secur.* 14(1): 12:1–12:34.

Ateniese, G., R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. (2007). "Provable data possession at untrusted stores". In: *Proceedings of the 14th ACM SIGSAC Conference on Computer and Communication Security (CCS 07)*. 598–609.

Ateniese, G., L. Chen, M. Etemad, and Q. Tang. (2020). "Proof of Storage-Time: Efficiently Checking Continuous Data Availability". In: *Proceedings of the 27th ISOC Network and Distributed System Security Symposium (NDSS 20)*. The Internet Society.

Aumann, Y. and Y. Lindell. (2010). "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries". *J. Cryptology*. 23(2): 281–343.

Aviv, A. J., S. G. Choi, T. Mayberry, and D. S. Roche. (2017). "ObliviSync: Practical Oblivious File Backup and Synchronization". In: *Proceedings of the 24th ISOC Network and Distributed System Security Symposium (NDSS 17)*.

"AWS CloudHSM". URL: https://aws.amazon.com/cloudhsm/.

Bahmani, R., F. Brasser, G. Dessouky, P. Jauernig, M. Klimmek, A.-R. Sadeghi, and E. Stapf. (2021). "CURE: A Security Architecture with Customizable and Resilient Enclaves". In: *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*.

Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. (2003). "Xen and the Art of Virtualization". In: *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003)*. Bolton Landing, NY. 164–177.

Bates, A., B. Mood, J. Pletcher, H. Pruse, M. Valafar, and K. Butler. (2012). "Detecting co-residency with active traffic analysis techniques". In: *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop (CCSW 12)*. 1–12.

Baumann, A., M. Peinado, and G. Hunt. (2014a). "Shielding Applications from an Untrusted Cloud with Haven". In: *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation (OSDI 2014)*. Broomfield, CO. 267–283.

Baumann, A., M. Peinado, and G. Hunt. (2014b). "Shielding Applications from an Untrusted Cloud with Haven". In: *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*.

Beaver, D. (1989). "Multiparty Protocols Tolerating Half Faulty Processors". In: *Advances in Cryptology—Crypto '89*. Vol. 435. *Lecture Notes in Computer Science*. Springer. 560–572.

Beaver, D. and S. Goldwasser. (1989). "Multiparty Computation with Faulty Majority". In: *Advances in Cryptology—Crypto '89*. Vol. 435. *Lecture Notes in Computer Science*. Springer. 589–590.

Ben-Or, M., S. Goldwasser, and A. Wigderson. (1988). "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*. ACM. 1–10.

Benson, K., R. Dowsley, and H. Shacham. (2011). "Do You Know Where Your Cloud Files Are?" In: *Proceedings of the 3rd ACM Cloud Computing Security Workshop (CCSW 11)*. Association for Computing Machinery. 73–82.

Bernstein, D. J. (2005). "Cache-timing attacks on AES". URL: https://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

Bhattacharyya, A., A. Sandulescu, M. Neugschwandtner, A. Sorniotti, B. Falsafi, M. Payer, and A. Kurmus. (2019). "SMoTherSpectre: exploiting speculative execution through port contention". In: *Proceedings of the 26th ACM SIGSAC Conference on Computer and Communications Security (CCS 19)*. 785–800.

Bindschaedler, V., M. Naveed, X. Pan, X. Wang, and Y. Huang. (2015). "Practicing Oblivious Access on Cloud Storage: The Gap, the Fallacy, and the New Way Forward". In: *Poceedings of the 2015 ACM SIGSAC Conference on Computer and Communication Security (CCS 15)*.

Blass, E., T. Mayberry, G. Noubir, and K. Onarlioglu. (2014). "Toward Robust Hidden Volumes Using Write-Only Oblivious RAM". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS 14)*.

Bogdanov, D., S. Laur, and J. Willemson. (2008). "Sharemind: A Framework for Fast Privacy-Preserving Computations". In: *Proceedings of the 13th European Symposium on Research in Computer Security (ESORICS)*. Springer-Verlag. 192–206.

Boneh, D., G. D. Crescenzo, R. Ostrovsky, and G. Persiano. (2004). "Public key encryption with keyword search". In: *Proceedings of Eurocrypt 2004*. LNCS 3027. 506–522.

Bonneau, J. and I. Mironov. (2006). "Cache-collision timing attacks against AES". In: *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 201–215.

Bowers, K. D., A. Juels, and A. Oprea. (2009). "HAIL: a high-availability and integrity layer for cloud storage". In: *Proceedings of 2009 ACM SIGSAC Conference on Computer and Communication Security (CCS 09)*.

Bowers, K. D., M. van Dijk, A. Juels, A. Oprea, and R. L. Rivest. (2011). "How to Tell If Your Cloud Files Are Vulnerable to Drive Crashes". In: *Proceedings of the 18th ACM SIGSAC Conference on Computer and Communications Security (CCS 11)*. ACM. 501–514.

Boyle, E., K.-M. Chung, and R. Pass. (2016). "Oblivious Parallel RAM and Applications". In: *Theory of Cryptography*. Ed. by E. Kushilevitz and T. Malkin.

Brasser, F., T. Frassetto, K. Riedhammer, A.-R. Sadeghi, T. Schneider, and C. Weinert. (2018). "VoiceGuard: Secure and Private Speech Processing". In: *Interspeech*.

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

Bugnion, E., J. Nieh, and D. Tsafrir. (2017). *Hardware and Software Support for Virtualization. Synthesis Lectures on Computer Architecture*. Morgan and Claypool Publishers.

Butt, S., H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy. (2012). "Self-service Cloud Computing". In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS 2012)*. Raleigh, NC. 253–264.

Cash, D., S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner. (2013a). "Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries". In: *Advances in Cryptology – CRYPTO 2013*. Ed. by R. Canetti and J. A. Garay. Berlin, Heidelberg: Springer Berlin Heidelberg. 353–373.

Cash, D., A. Küpçü, and D. Wichs. (2013b). "Dynamic Proofs of Retrievability via Oblivious RAM". In: *Advances in Cryptology – EUROCRYPT 2013*. Springer Berlin Heidelberg. 279–295.

Chakraborti, A., A. J. Aviv, S. G. Choi, T. Mayberry, D. S. Roche, and R. Sion. (2019). "rORAM: Efficient Range ORAM with O(log2N) Locality". In: *Proceedings of the 26th ISOC Network and Distributed System Security Symposium (NDSS 19)*.

Chakraborti, A. and R. Sion. (2019). "ConcurORAM: High-Throughput Stateless Parallel Multi-Client ORAM". In: *Proceedings of the 26th ISOC Network and Distributed System Security Symposium (NDSS 19)*.

Chan, T.-H. H., K.-M. Chung, and E. Shi. (2017a). "On the Depth of Oblivious Parallel RAM". In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by T. Takagi and T. Peyrin.

Chan, T.-H. H., Y. Guo, W.-K. Lin, and E. Shi. (2017b). "Oblivious Hashing Revisited, and Applications to Asymptotically Efficient ORAM and OPRAM". In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by T. Takagi and T. Peyrin.

Chang, Y.-C. and M. Mitzenmacher. (2005). "Privacy Preserving Keyword Searches on Remote Encrypted Data". In: *Applied Cryptography and Network Security*. Ed. by J. Ioannidis, A. Keromytis, and M. Yung. Berlin, Heidelberg: Springer Berlin Heidelberg.

Chaum, D., C. Crépeau, and I. Damgård. (1988). "Multiparty Unconditionally Secure Protocols (Extended Abstract)". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*. ACM. 11–19.

Chen, B., R. Curtmola, G. Ateniese, and R. Burns. (2010). "Remote Data Checking for Network Coding-based Distributed Storage Systems". In: *Proceedings of the 2010 ACM Cloud Computing Security Workshop (CCSW 10)*.

Chen, B., H. Lin, and S. Tessaro. (2016). "Oblivious Parallel RAM: Improved Efficiency and Generic Constructions". In: *Theory of Cryptography*. Ed. by E. Kushilevitz and T. Malkin. Berlin, Heidelberg: Springer Berlin Heidelberg. 205–234.

Chen, B. and R. Curtmola. (2013). "Towards Self-Repairing Replication-Based Storage Systems Using Untrusted Clouds". In: *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*. ACM. 377–388.

Chen, B. and R. Curtmola. (2014). "Auditable Version Control Systems". In: *Proceedings of the 21th ISOC Network and Distributed System Security Symposium (NDSS '14)*.

Chen, B. and R. Curtmola. (2017). "Remote data integrity checking with server-side repair". *Journal of Computer Security*. 25(6): 537–584.

Chen, D., H. Yuan, S. Hu, Q. Wang, and C. Wang. (2021). "BOSSA: A Decentralized System for Proofs of Data Retrievability and Replication". *IEEE Transactions on Parallel and Distributed Systems*. 32(4): 786–798.

Chen, G., M. Li, F. Zhang, and Y. Zhang. (2019a). "Defeating Speculative-Execution Attacks on SGX with HyperRace". In: *Proceedings of the 2019 IEEE Conference on Dependable and Secure Computing (DSC 19)*.

Chen, H., I. Chillotti, and L. Ren. (2019b). "Onion Ring ORAM: Efficient Constant Bandwidth Oblivious RAM from (Leveled) TFHE". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 19)*. Dallas, Texas, USA: Association for Computing Machinery. 491–505.

Chen, S., F. Liu, Z. Mi, Y. Zhang, R. B. Lee, H. Chen, and X. Wang. (2018). "Leveraging hardware transactional memory for cache side-channel defenses". In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 601–608.

Chen, S., X. Zhang, M. K. Reiter, and Y. Zhang. (2017). "Detecting privileged side-channel attacks in shielded execution with Déjá Vu". In: *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security (AsiaCCS 17)*. ACM. 7–18.

Chen, X., T. Garfinkel, E. C. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, J. Dwoskin, and D. R. Ports. (2008). "Overshadow: A Virtualization-based Approach to Retrofitting Protection in Commodity Operating Systems". In: *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2008)*. Seattle, WA. 2–13.

Chhabra, S., B. Rogers, Y. Solihin, and M. Prvulovic. (2011). "SecureME: A Hardware-software Approach to Full System Security". In: *Proceedings of the 25th International Conference on Supercomputing (ICS 2011)*. Tucson, AZ. 108–119.

Chiappetta, M., E. Savas, and C. Yilmaz. (2016). "Real time detection of cache-based side-channel attacks using hardware performance counters". *Applied Soft Computing*. 49: 1162–1174.

Chor, B., E. Kushilevitz, O. Goldreich, and M. Sudan. (1998). "Private information retrieval". *Journal of the ACM*. 45(6).

Cohen, E., M. Dahlweid, M. Hillebrand, D. Leinenbach, M. Moskal, T. Santen, W. Schulte, and S. Tobies. (2009). "VCC: A Practical System for Verifying Concurrent C". In: *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2009)*. Munich, Germany. 23–42.

Colp, P., M. Nanavati, J. Zhu, W. Aiello, G. Coker, T. Deegan, P. Loscocco, and A. Warfield. (2011). "Breaking Up is Hard to Do: Security and Functionality in a Commodity Hypervisor". In: *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP 2011)*. Cascais, Portugal. 189–202.

Costanzo, D., Z. Shao, and R. Gu. (2016). "End-to-End Verification of Information-Flow Security for C and Assembly Programs". In: *Proceedings of the 37th ACM Conference on Programming Language Design and Implementation (PLDI 2016)*.

"Cryptographic Module Validation Program". URL: https://csrc.nist.gov/projects/cryptographic-module-validation-program.

Curtmola, R., O. Khan, and R. Burns. (2008a). "Robust Remote Data Checking". In: *Proceedings of the 2008 ACM Workshop on Storage Security and Survivability (StorageSS)*.

Curtmola, R., O. Khan, R. Burns, and G. Ateniese. (2008b). "MR-PDP: Multiple-Replica Provable Data Possession". In: *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS 08)*.

Curtmola, R., J. Garay, S. Kamara, and R. Ostrovsky. (2006). "Searchable symmetric encryption: improved definitions and efficient constructions". In: *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*.

Dall, C. and J. Nieh. (2014). "KVM/ARM: The Design and Implementation of the Linux ARM Hypervisor". In: *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*. 333–348.

Damgård, I., C. Ganesh, and C. Orlandi. (2019). "Proofs of Replicated Storage Without Timing Assumptions". In: *Advances in Cryptology – CRYPTO 2019*. Springer International Publishing. 355–380.

Dang, H., E. Purwanto, and E.-C. Chang. (2017). "Proofs of Data Residency: Checking Whether Your Cloud Files Have Been Relocated". In: *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security (AsiaCCS 17)*. Association for Computing Machinery. 408–422.

Deswarte, Y., J.-J. Quisquater, and A. Saidane. (2003). "Remote integrity checking". In: *Proceedings of Conference on Integrity and Internal Control in Information Systems (IICIS'03)*.

Devadas, S., M. van Dijk, C. W. Fletcher, L. Ren, E. Shi, and D. Wichs. (2016). "Onion ORAM: A Constant Bandwidth Blowup Oblivious RAM". In: *Theory of Cryptography*. Ed. by E. Kushilevitz and T. Malkin.

Didier, G. and C. Maurice. (2021). "Calibration Done Right: Noiseless Flush+ Flush Attacks". In: *Proceedings of the 18th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2021)*.

Dijk, M. van, A. Juels, A. Oprea, R. L. Rivest, E. Stefanov, and N. Triandopoulos. (2012). "Hourglass Schemes: How to Prove That Cloud Files Are Encrypted". In: *Proceedings of the 2012 ACM SIGSAC Conference on Computer and Communications Security (CCS 12)*. ACM. 265–280.

Dimakis, A. G., B. Godfrey, M. J. Wainwright, and K. Ramchandran. (2007). "Network Coding for Distributed Storage Systems". In: *INFOCOM*.

Dimakis, A. G., P. B. Godfrey, Y. Wu, M. O. Wainwright, and K. Ramchandran. (2010). "Network Coding for Distributed Storage Systems". *IEEE Transactions on Information Theory*.

Dwork, C. and A. Roth. (2014). "The Algorithmic Foundations of Differential Privacy". *Foundations and Trends® in Theoretical Computer Science*. 9(3-4): 211–407. URL: http://dx.doi.org/10.1561/0400000042.

"ENFORCER Server". URL: https://enforcerserver.com/.

Erway, C. C., A. Küpçü, C. Papamanthou, and R. Tamassia. (2009). "Dynamic Provable Data Possession". In: *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS 09)*.

Erway, C. C., A. Küpçü, C. Papamanthou, and R. Tamassia. (2015). "Dynamic Provable Data Possession". *ACM Trans. Inf. Syst. Secur.* 17(4).

Eskandarian, S. and M. Zaharia. (2019). "ObliDB: Oblivious Query Processing for Secure Databases". *Proc. VLDB Endow.* 13(2).

Etemad, M. and A. Küpçü. (2013). "Transparent, Distributed, and Replicated Dynamic Provable Data Possession". In: *Applied Cryptography and Network Security*. Springer Berlin Heidelberg. 1–18.

Etemad, M. and A. Küpçü. (2020). "Generic Dynamic Data Outsourcing Framework for Integrity Verification". *ACM Comput. Surv.* 53(1).

Evans, D., V. Kolesnikov, and M. Rosulek. (2018). "A Pragmatic Introduction to Secure Multi-Party Computation". *Foundations and Trends® in Privacy and Security*. 2(2-3): 70–246. URL: http://dx.doi.org/10.1561/3300000019.

Falzon, F., E. A. Markatou, Akshima, D. Cash, A. Rivkin, J. Stern, and R. Tamassia. (2020). "Full Database Reconstruction in Two Dimensions". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS 20)*. New York, NY, USA: Association for Computing Machinery.

Filho, D. L. G. and P. S. L. M. Baretto. (2006). "Demonstrating data possession and uncheatable data transfer". IACR ePrint archive.

"FIPS 140-2". URL: https://en.wikipedia.org/wiki/FIPS_140-2.

"FIPS 140-3". URL: https://en.wikipedia.org/wiki/FIPS_140-3.

Fletcher, C., M. Naveed, L. Ren, E. Shi, and E. Stefanov. (2015). "Bucket ORAM: Single Online Roundtrip, Constant Bandwidth Oblivious RAM". Cryptology ePrint Archive, Report 2015/1065.

"Frequently Asked Questions on seL4". URL: https://docs.sel4.systems/projects/sel4/frequently-asked-questions.html.

Fuller, B., M. Varia, A. Yerukhimovich, E. Shen, A. Hamlin, V. Gadepally, R. Shay, J. D. Mitchell, and R. K. Cunningham. (2017). "SoK: Cryptographically Protected Database Search". In: *Proceedings of the 2017 IEEE Symposium on Security and Privacy (S&P 17)*. 172–191.

Garfinkel, T., B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. (2003). "Terra: A Virtual Machine-based Platform for Trusted Computing". In: *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP 2003)*. Bolton Landing, NY. 193–206.

Garg, S., P. Mohassel, and C. Papamanthou. (2016). "TWORAM: Efficient Oblivious RAM in Two Rounds with Applications to Searchable Encryption". In: *Advances in Cryptology – CRYPTO 2016*. Ed. by M. Robshaw and J. Katz. Springer Berlin Heidelberg.

Gentry, C., K. A. Goldman, S. Halevi, C. S. Jutla, M. Raykova, and D. Wichs. (2013). "Optimizing ORAM and Using It Efficiently for Secure Computation". In: *Proceedings of the 13th Privacy Enhancing Technologies Symposium (PETS 13)*.

Godfrey, M. and M. Zulkernine. (2014). "Preventing cache-based side-channel attacks in a cloud environment". *IEEE transactions on cloud computing*. 2(4): 395–408.

Goh, E. (2003). "Secure Indexes". Cryptology ePrint Archive, Report 2003/216.

Goldreich, O. (2004). *Foundations of Cryptography, vol. II: Basic Applications*. Cambridge University Press.

Goldreich, O., S. Micali, and A. Wigderson. (1987). "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority". In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*. ACM. 218–229.

Goldreich, O. and R. Ostrovsky. (1996). "Software Protection and Simulation on Oblivious RAMs". *Journal of the ACM*. 43: 431–473.

Goldwasser, S. and L. A. Levin. (1990). "Fair Computation of General Functions in Presence of Immoral Majority". In: *Advances in Cryptology—Crypto '90*. Vol. 537. *Lecture Notes in Computer Science*. Springer. 77–93.

Golle, P., S. Jarecki, and I. Mironov. (2002). "Cryptographic Primitives Enforcing Communication and Storage Complexity." In: *Proceedings of the Internation Conference on Financial Cryptography and Data Security (FC 02)*. 120–135.

Golle, P., J. Staddon, and B. Waters. (2004). "Secure conjunctive keyword search over encrypted data". In: *Proceedings of ACNS*. Springer-Verlag; Lecture Notes in Computer Science 3089. 31–45.

Gondree, M. and Z. N. Peterson. (2013). "Geolocation of Data in the Cloud". In: *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*. Association for Computing Machinery. 25–36.

Goodrich, M. T. and M. Mitzenmacher. (2011). "Privacy-preserving Access of Outsourced Data via Oblivious RAM Simulation". In: *Proceedings of the 38th International Conference on Automata, Languages and Programming (ICALP 11)*.

Goodrich, M. T., M. Mitzenmacher, O. Ohrimenko, and R. Tamassia. (2011). "Oblivious RAM Simulation with Efficient Worst-case Access Overhead". In: *Proceedings of the 2011 ACM Cloud Computing Security Workshop (CCSW 11)*. 95–100.

Goodrich, M. T. and J. A. Simons. (2014). "Data-Oblivious Graph Algorithms in Outsourced External Memory". In: *Combinatorial Optimization and Applications*. Ed. by Z. Zhang, L. Wu, W. Xu, and D.-Z. Du.

"Google Cloud Key Management". URL: https://cloud.google.com/security-key-management.

Goyal, V., A. Polychroniadou, and Y. Song. (2021). "Unconditional Communication-Efficient MPC via Hall's Marriage Theorem". In: *Advances in Cryptology—Crypto 2021, Part II*. Vol. 12826. *Lecture Notes in Computer Science*. Springer. 275–304.

Gras, B., K. Razavi, H. Bos, and C. Giuffrida. (2018). "Translation leak-aside buffer: Defeating cache side-channel protections with {TLB} attacks". In: *Proceedings of the 27th USENIX Security Symposium (USENIX Security 18)*. 955–972.

Grubbs, P., M.-S. Lacharite, B. Minaud, and K. G. Paterson. (2018). "Pump up the Volume: Practical Database Reconstruction from Volume Leakage on Range Queries". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*.

Grubbs, P., T. Ristenpart, and V. Shmatikov. (2017). "Why Your Encrypted Database Is Not Secure". In: *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*.

Gruss, D., J. Lettner, F. Schuster, O. Ohrimenko, I. Haller, and M. Costa. (2017). "Strong and efficient cache side-channel protection using hardware transactional memory". In: *Proceedings of the 26th USENIX Security Symposium (USENIX Security 17)*. 217–233.

Gruss, D., C. Maurice, K. Wagner, and S. Mangard. (2016). "Flush+Flush: a fast and stealthy cache attack". In: *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 16)*. Springer. 279–299.

Gu, R., Z. Shao, H. Chen, J. Kim, J. Koenig, X. Wu, V. Sjöberg, and D. Costanzo. (2019). "Building Certified Concurrent OS Kernels". *Communications of the ACM*. 62(10): 89–99.

Gu, R., Z. Shao, H. Chen, X. N. Wu, J. Kim, V. Sjöberg, and D. Costanzo. (2016). "CertiKOS: An Extensible Architecture for Building Certified Concurrent OS Kernels". In: *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016)*. Savannah, GA. 653–669.

Gullasch, D., E. Bangerter, and S. Krenn. (2011). "Cache games–bringing access-based cache attacks on AES to practice". In: *Proceedings of the 32nd IEEE Symposium on Security and Privacy (S&P 11)*. IEEE. 490–505.

Halderman, J. A., S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. (2009). "Lest we remember: cold-boot attacks on encryption keys". *Communications of the ACM*. 52(5): 91–98.

"Hardware Security Module (HSM)". URL: https://en.wikipedia.org/wiki/Hardware%5C_security%5C_module.

Hastings, M., B. Hemenway, D. Noble, and S. Zdancewic. (2019). "SoK: General Purpose Compilers for Secure Multi-Party Computation". In: *Proceedings of the IEEE Symposium on Security and Privacy (S&P 19)*. IEEE. 1220–1237.

Hazay, C. and Y. Lindell. (2010). *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Springer.

He, Y., Y. Xu, X. Jia, S. Zhang, P. Liu, and S. Chang. (2020). "EnclavePDP: A General Framework to Verify Data Integrity in Cloud Using Intel SGX". In: *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. USENIX Association. 195–208.

Heiser, G. and B. Leslie. (2010). "The OKL4 Microvisor: Convergence Point of Microkernels and Hypervisors". In: *Proceedings of the 1st ACM Asia-pacific Workshop on Workshop on Systems (APSys 2010)*. New Delhi, India. 19–24.

Hetzelt, F. and R. Buhren. (2017). "Security analysis of encrypted virtual machines". *ACM SIGPLAN Notices*. 52(7): 129–142.

Hoang, T., J. Guajardo, and A. A. Yavuz. (2020). "MACAO: A Maliciously-Secure and Client-Efficient Active ORAM Framework". In: *Proeccedings of the 27th ISOC Network and Distributed System Security Symposium (NDSS 20)*.

Hoang, T., C. D. Ozkaptan, A. A. Yavuz, J. Guajardo, and T. Nguyen. (2017). "S³ORAM: A Computation-Efficient and Constant Client Bandwidth Blowup ORAM with Shamir Secret Sharing". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 17)*. Dallas, Texas, USA: Association for Computing Machinery. 491–505.

Hoang, T., M. O. Ozmen, Y. Jang, and A. A. Yavuz. (2018). "Hardware-Supported ORAM in Effect: Practical Oblivious Search and Update on Very Large Dataset". *Proceedings on Privacy Enhancing Technologies (PETS 18)*. 2019: 172–191.

Hofmann, O. S., S. Kim, A. M. Dunn, M. Z. Lee, and E. Witchel. (2013). "InkTag: Secure Applications on an Untrusted Operating System". In: *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2013)*. Houston, TX. 265–278.

Hua, Z., J. Gu, Y. Xia, H. Chen, B. Zang, and Haibing. (2017). "vTZ: Virtualizing ARM Trustzone". In: *Proceedings of the 26th USENIX Security Symposium (USENIX Security 2017)*. Vancouver, BC, Canada. 541–556.

Huang, Y., J. Katz, and D. Evans. (2012). "Quid-Pro-Quo-tocols: Strengthening Semi-honest Protocols with Dual Execution". In: *Proceedings of the IEEE Symposium on Security and Privacy 2012 (S&P 12)*. IEEE Computer Society. 272–284.

Hubert Chan, T.-H. and E. Shi. (2017). "Circuit OPRAM: Unifying Statistically and Computationally Secure ORAMs and OPRAMs". In: *Theory of Cryptography*. Ed. by Y. Kalai and L. Reyzin.

Inci, M. S., B. Gulmezoglu, G. Irazoqui, T. Eisenbarth, and B. Sunar. (2016). "Cache attacks enable bulk key recovery on the cloud". In: *Proceedings of the 2016 International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 368–388.

Intel. (2014). "Intel Software Guard Extensions Programming Reference". URL: https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf.

Intel. (2017). "Overview of Intel Protected File System Library Using Software Guard Extensions". URL: https://software.intel.com/en-us/articles/overview-of-intel-protected-file-system-library-using-software-guard-extensions.

Intel. (2021). "Intel Trust Domain CPU Architectural Extensions". URL: https://software.intel.com/content/dam/develop/external/us/en/documents-tps/intel-tdx-cpu-architectural-specification.pdf.

Intel Corporation. (2014). "Intel Software Guard Extensions Programming Reference". URL: https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf.

Irazoqui, G., T. Eisenbarth, and B. Sunar. (2015). "S $ A: A shared cache attack that works across cores and defies VM sandboxing–and its application to AES". In: *Proceedings of the 2015 IEEE Symposium on Security and Privacy (S&P 15)*. IEEE. 591–604.

Irazoqui, G., M. S. Inci, T. Eisenbarth, and B. Sunar. (2014). "Wait a minute! A fast, Cross-VM attack on AES". In: *Proceedings of the 2014 International Workshop on Recent Advances in Intrusion Detection*. Springer. 299–319.

Ishai, Y. and E. Kushilevitz. (2004). "On the Hardness of Information-Theoretic Multiparty Computation". In: *Advances in Cryptology—Eurocrypt 2004*. Vol. 3027. *Lecture Notes in Computer Science*. Springer. 439–455.

Johnson, S. (2018). "Intel SGX and Side-Channels". URL: https://software.intel.com/en-us/articles/intel-sgx-and-side-channels.

Juels, A. and B. S. Kaliski. (2007). "PORs: Proofs of Retrievability for Large Files". In: *Proceedings of the 2007 ACM SIGSAC Conference on Computer and Communication Security (CCS 07)*.

Kamara, S., C. Papamanthou, and T. Roeder. (2012). "Dynamic Searchable Symmetric Encryption". In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*.

Kaplan, D. (2017). "Protecting VM register state with SEV-ES". URL: https://www.amd.com/system/files/TechDocs/Protecting%5C%20VM%5C%20Register%5C%20State%5C%20with%5C%20SEV-ES.pdf.

Kaplan, D., J. Powell, and T. Woller. (2016). "AMD memory encryption". *White paper*.

Kellaris, G., G. Kollios, K. Nissim, and A. O'Neill. (2016). "Generic Attacks on Secure Outsourced Databases". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1329–1340.

Keller, E., J. Szefer, J. Rexford, and R. B. Lee. (2010). "NoHype: Virtualized Cloud Infrastructure Without the Virtualization". In: *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA 2010)*. Saint-Malo, France. 350–361.

Kelsey, J., B. Schneier, D. Wagner, and C. Hall. (1998). "Side channel cryptanalysis of product ciphers". In: *Proceedings of the 1998 European Symposium on Research in Computer Security (ESORICS 98)*. Springer. 97–110.

Kida, L. S., S. Desai, A. Trivedi, R. Lal, V. Scarlata, and S. K. Ghosh. (2020). "HCC: 100 Gbps AES-GCM Encrypted Inline DMA Transfers Between SGX Enclave and FPGA Accelerator". In: *Proceedings of the 22nd International Conference on Information and Communications Security (ICICS 20)*.

Kim, T., M. Peinado, and G. Mainar-Ruiz. (2012). "{STEALTHMEM}: System-level protection against cache-based side channel attacks in the cloud". In: *Proceedings of the 21st USENIX Security Symposium (USENIX Security 12)*. 189–204.

Kim, Y., R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu. (2014). "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors". *ACM SIGARCH Computer Architecture News*. 42(3): 361–372.

Kivity, A., Y. Kamay, D. Laor, U. Lublin, and A. Liguori. (2007). "KVM: the Linux Virtual Machine Monitor". In: *In Proceedings of the 2007 Ottawa Linux Symposium (OLS 2007)*. Ottawa, ON, Canada.

Klein, G., J. Andronick, K. Elphinstone, T. Murray, T. Sewell, R. Kolanski, and G. Heiser. (2014). "Comprehensive Formal Verification of an OS Microkernel". *ACM Transactions on Computer Systems*.

Klein, G., J. Andronick, M. Fernandez, I. Kuz, T. Murray, and G. Heiser. (2018). "Formally Verified Software in the Real World". *Communications of the ACM*.

Klein, G., K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. (2009). "seL4: Formal Verification of an OS Kernel". In: *Proceedings of the 22nd ACM Symposium on Operating Systems Principles (SOSP 2009)*. Big Sky, MT. 207–220.

Kocher, P. C. (1996). "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems". In: *Annual International Cryptology Conference*. Springer. 104–113.

Koschel, J., C. Giuffrida, H. Bos, and K. Razavi. (2020). "TagBleed: Breaking KASLR on the Isolated Kernel Address Space using Tagged TLBs". In: *Proceedings of the 2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 309–321.

Kushilevitz, E., S. Lu, and R. Ostrovsky. (2012). "On the (in)Security of Hash-based Oblivious RAM and a New Balancing Scheme". In: *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 12)*.

Leinenbach, D. and T. Santen. (2009). "Verifying the Microsoft Hyper-V hypervisor with VCC". In: *Proceedings of the 16th International Symposium on Formal Methods (FM 2009)*. Eindhoven, The Netherlands. 806–809.

Leontiadis, I. and R. Curtmola. (2018). "Secure Storage with Replication and Transparent Deduplication". In: *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. Association for Computing Machinery. 13–23.

Leontiadis, I. and R. Curtmola. (2019). "Auditable Compressed Storage". In: *International Conference on Information Security*. Springer International Publishing. 67–86.

Li, L. and L. Lazos. (2020). "Proofs of Physical Reliability for Cloud Storage Systems". *IEEE Transactions on Parallel and Distributed Systems*. 31(5): 1048–1065.

Li, L. and A. Datta. (2017). "Write-Only Oblivious RAM-Based Privacy-Preserved Access of Outsourced Data". *Int. J. Inf. Secur.*

Li, S.-W., J. S. Koh, and J. Nieh. (2019). "Protecting Cloud Virtual Machines from Hypervisor and Host Operating System Exploits". In: *Proceedings of the 28th USENIX Security Symposium (USENIX Security 2019)*. Santa Clara, CA. 1357–1374.

Li, S.-W., X. Li, R. Gu, J. Nieh, and J. Z. Hui. (2021a). "A Secure and Formally Verified Linux KVM Hypervisor". In: *Proceedings of the 42nd IEEE Symposium on Security & Privacy (IEEE SP 2021)*. San Francisco, CA. 1782–1799.

Li, S.-W., X. Li, R. Gu, J. Nieh, and J. Z. Hui. (2021b). "Formally Verified Memory Protection for a Commodity Multiprocessor Hypervisor". In: *Proceedings of the 30th USENIX Security Symposium (USENIX Security 2021)*. Vancouver, British Columbia, Canada. 3953–3970.

Lindell, Y. (2021). "Secure multiparty computation". *Communications of the ACM*. 64(1): 86–96.

Lindell, Y. and B. Pinkas. (2008). "Secure Multiparty Computation for Privacy-Preserving Data Mining".

Lindemann, J. and M. Fischer. (2018). "A Memory-Deduplication Side-Channel Attack to Detect Applications in Co-Resident Virtual Machines". In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*.

Liu, C., L. Zhu, M. Wang, and Y.-a. Tan. (2014a). "Search pattern leakage in searchable encryption: Attacks and new construction". *Inf. Sci.* 265: 176–188.

Liu, F., Q. Ge, Y. Yarom, F. Mckeen, C. Rozas, G. Heiser, and R. B. Lee. (2016). "Catalyst: Defeating last-level cache side channel attacks in cloud computing". In: *Proceedings of the 2016 IEEE international symposium on high performance computer architecture (HPCA 16)*. IEEE. 406–418.

Liu, F., Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee. (2015a). "Last-level cache side-channel attacks are practical". In: *Proceedings of the 2015 IEEE symposium on security and privacy (S&P 15)*. IEEE. 605–622.

Liu, F., L. Ren, and H. Bai. (2014b). "Mitigating cross-VM side channel attack on multiple tenants cloud platform." *J. Comput.* 9(4): 1005–1013.

Liu, Y., T. Zhou, K. Chen, H. Chen, and Y. Xia. (2015b). "Thwarting Memory Disclosure with Efficient Hypervisor-enforced Intra-domain Isolation". In: *Proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS 2015)*. Denver, CO. 1607–1619.

Maffei, M., G. Malavolta, M. Reinert, and D. Schröder. (2017). "Maliciously Secure Multi-Client ORAM". In: *Applied Cryptography and Network Security*. Ed. by D. Gollmann, A. Miyaji, and H. Kikuchi.

Magouryk, C. (2021). "Arm-based cloud computing is the next big thing: Introducing Arm on Oracle Cloud Infrastructure". URL: https://blogs.oracle.com/cloud-infrastructure/post/arm-based-cloud-computing-is-the-next-big-thing-introducing-arm-on-oracle-cloud-infrastructure.

Markettos, A., C. Rothwell, B. Gutstein, A. Pearce, P. Neumann, S. Moore, and R. Watson. (2019). "Thunderclap: Exploring vulnerabilities in Operating System IOMMU protection via DMA from untrustworthy peripherals". In: *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS 19)*.

Martin, R., J. Demme, and S. Sethumadhavan. (2012). "Timewarp: Rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks". In: *Proceedings of the 2012 39th Annual International Symposium on Computer Architecture (ISCA 12)*. IEEE. 118–129.

McCune, J. M., Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig. (2010). "TrustVisor: Efficient TCB Reduction and Attestation". In: *Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP 2010)*. Oakland, CA. 143–158.

"Microsoft Azure Key Vault". URL: https://azure.microsoft.com/en-us/services/key-vault/.

Ta-Min, R., L. Litty, and D. Lie. (2006). "Splitting Interfaces: Making Trust Between Applications and Operating Systems Configurable". In: *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI 2006)*. Seattle, WA. 279–292.

Moon, S.-J., V. Sekar, and M. K. Reiter. (2015). "Nomad: Mitigating arbitrary cloud side channels via provider-assisted migration". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS 15)*. 1595–1606.

Murray, D. G., G. Milos, and S. Hand. (2008). "Improving Xen Security Through Disaggregation". In: *Proceedings of the 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2008)*. Seattle, WA. 151–160.

Nayak, K., C. W. Fletcher, L. Ren, N. Chandran, S. Lokam, E. Shi, and V. Goyal. (2017). "HOP: Hardware makes Obfuscation Practical". In: *Proceedings of the 2017 ISOC Network and Distributed System Security Symposium (NDSS 17)*.

Nayak, K. and J. Katz. (2016). "An Oblivious Parallel RAM with O(log2 N) Parallel Runtime Blowup". *IACR Cryptology ePrint Archive*. 2016: 1141.

Nguyen, A., H. Raj, S. Rayanchu, S. Saroiu, and A. Wolman. (2012). "Delusional Boot: Securing Hypervisors Without Massive Re-engineering". In: *Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys 2012)*. Bern, Switzerland. 141–154.

Oberhauser, J., R. L. de Lima Chehab, D. Behrens, M. Fu, A. Paolillo, L. Oberhauser, K. Bhat, Y. Wen, H. Chen, J. Kim, and V. Vafeiadis. (2021). "VSync: Push-Button Verification and Optimization for Synchronization Primitives on Weak Memory Models". In: *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2021)*. Detroit, MI.

Ohrimenko, O., F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa. (2016). "Oblivious Multi-Party Machine Learning on Trusted Processors". In: *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*.

Osvik, D. A., A. Shamir, and E. Tromer. (2006). "Cache attacks and countermeasures: the case of AES". In: *Cryptographers' track at the RSA conference*. Springer. 1–20.

Owens, R. and W. Wang. (2011). "Non-Interactive OS Fingerprinting through Memory de-Duplication Technique in Virtual Machines". In: *Proceedings of the 30th IEEE International Performance Computing and Communications Conference*.

Oya, S. and F. Kerschbaum. (2021). "Hiding the Access Pattern is Not Enough: Exploiting Search Pattern Leakage in Searchable Encryption". In: *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*.

Paccagnella, R., L. Luo, and C. W. Fletcher. (2021). "Lord of the Ring (s): Side Channel Attacks on the {CPU} On-Chip Ring Interconnect Are Practical". In: *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*.

Page, D. (2005). "Partitioned cache architecture as a side-channel defence mechanism". URL: https://ia.cr/2005/280.

Page, D. (2002). "Theoretical use of cache memory as a cryptanalytic side-channel." *IACR Cryptol. ePrint Arch.* 2002(169): 1–23.

Pappas, V., F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, and S. Bellovin. (2014). "Blind Seer: A Scalable Private DBMS". In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy (S&P 14)*.

Patel, S., G. Persiano, M. Raykova, and K. Yeo. (2018). "PanORAMa: Oblivious RAM with Logarithmic Overhead". In: *IEEE Annual Symposium on Foundations of Computer Science (FOCS 18)*.

Percival, C. (2005). "Cache missing for fun and profit".

Pessl, P., D. Gruss, C. Maurice, M. Schwarz, and S. Mangard. (2016). "DRAMA: Exploiting Dram Addressing for Cross-Cpu Attacks". In: *Proceedings of the 25th USENIX Conference on Security Symposium. SEC'16*. Austin, TX, USA: USENIX Association. 565–581.

Peterson, Z. N. J., M. Gondree, and R. Beverly. (2011). "A Position Paper on Data Sovereignty: The Importance of Geolocating Data in the Cloud". In: *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*.

Pinkas, B. and T. Reinman. (2010). "Oblivious RAM Revisited". In: *Proceedings of the 30th Annual Conference on Advances in Cryptology (CRYPTO 10)*.

Popa, R. A., C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. (2011). "CryptDB: Protecting Confidentiality with Encrypted Query Processing". In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*.

Protocol Labs. (2017a). "Filecoin: A decentralized storage network". URL: https://filecoin.io/filecoin.pdf.

Protocol Labs. (2017b). "Proof of replication". URL: https://filecoin.io/proof-of-replication.pdf.

Rabin, T. and M. Ben-Or. (1989). "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority". In: *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*. ACM. 73–85.

Razavi, K., B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos. (2016). "Flip feng shui: Hammering a needle in the software stack". In: *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*. 1–18.

Ren, L., C. Fletcher, A. Kwon, E. Stefanov, E. Shi, M. van Dijk, and S. Devadas. (2015). "Constants Count: Practical Improvements to Oblivious RAM". In: *Proceedings of the 24th USENIX Security Symposium (USENIX Security 15)*.

Ren, L., X. Yu, C. W. Fletcher, M. van Dijk, and S. Devadas. (2013). "Design Space Exploration and Optimization of Path Oblivious RAM in Secure Processors".

Riley, R., X. Jiang, and D. Xu. (2008). "Guest-Transparent Prevention of Kernel Rootkits with VMM-Based Memory Shadowing". In: *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID 2008)*. Cambridge, MA. 1–20.

Ristenpart, T., E. Tromer, H. Shacham, and S. Savage. (2009). "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*. 199–212.

Roche, D. S., A. Aviv, S. G. Choi, and T. Mayberry. (2017). "Deterministic, Stash-Free Write-Only ORAM". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 17)*.

Sahin, C., V. Zakhary, A. E. Abbadi, H. Lin, and S. Tessaro. (2016). "TaoStore: Overcoming Asynchronicity in Oblivious Data Storage". In: *Proceedings of the 2016 IEEE Symposium on Security and Privacy (S&P 16)*.

Sasy, S., S. Gorbunov, and C. W. Fletcher. (2018). "ZeroTrace: Oblivious Memory Primitives from Intel SGX". In: *Proceedings of the 2018 ISOC Network and Distributed System Security Symposium (NDSS 18)*.

Schwarz, T. S. J. and E. L. Miller. (2006). “Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage.” In: *Proceedings of the 2006 IEEE International Conference on Distributed Computing Systems (ICDCS 06)*. IEEE Computer Society.

Sebe, F., A. Martinez-Balleste, Y. Deswarte, J. Domingo-Ferrer, and J.-J. Quisquater. (2004). “Time-Bounded Remote File Integrity Checking”. *Tech. rep.* No. 04429. LAAS.

“Security Requirements for Cryptographic Modules”. URL: https://nvlpubs.nist.gov.

“seL4 Reference Manual Version 11.0.0”. (2019). Data61. URL: http://sel4.systems/Info/Docs/seL4-manual-11.0.0.pdf.

“seL4 Supported Platforms”. URL: https://docs.sel4.systems/Hardware.

Seshadri, A., M. Luk, N. Qu, and A. Perrig. (2007). “SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes”. In: *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP 2007)*. Stevenson, WA. 335–350.

Shacham, H. and B. Waters. (2008). “Compact Proofs of Retrievability”. In: *Proceedings of the Annual International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt 2008)*.

Shacham, H. and B. Waters. (2013). “Compact Proofs of Retrievability”. *J. Cryptol.* 26(3): 442–483.

Shah, M., M. Baker, J. C. Mogul, and R. Swaminathan. (2007). “Auditing to Keep Online Storage Services Honest”. In: *Proceedings of the 11th Workshop on Hot Topics in Operating Systems (HotOS XI)*. Usenix.

Shah, M. A., R. Swaminathan, and M. Baker. (2008). “Privacy-Preserving Audit and Extraction of Digital Contents”. *ePrint Archive Report.* (2008/186).

Shen, J., F. Guo, X. Chen, and W. Susilo. (2020). “Secure Cloud Auditing with Efficient Ownership Transfer”. In: *Computer Security – ESORICS 2020*. Springer International Publishing. 611–631.

Shi, E. (2020). "Path Oblivious Heap: Optimal and Practical Oblivious Priority Queue". In: *Proceedings of the 2020 IEEE Symposium on Security and Privacy (S&P 20)*.

Shi, E., T.-H. H. Chan, E. Stefanov, and M. Li. (2011a). "Oblivious RAM with O((Logn)3) Worst-case Cost". In: *Proceedings of the 17th International Conference on The Theory and Application of Cryptology and Information Security (ASIACRYPT 11)*.

Shi, E., E. Stefanov, and C. Papamanthou. (2013). "Practical Dynamic Proofs of Retrievability". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS 13)*. ACM. 325–336.

Shi, J., X. Song, H. Chen, and B. Zang. (2011b). "Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring". In: *Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE. 194–199.

Shi, L., Y. Wu, Y. Xia, N. Dautenhahn, H. Chen, B. Zang, and J. Li. (2017). "Deconstructing Xen". In: *24th Annual Network and Distributed System Security Symposium (NDSS 2017)*. San Diego, CA.

Shih, M.-W., M. Kumar, T. Kim, and A. Gavrilovska. (2016). "S-NFV: Securing NFV States by Using SGX". In: *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization (SDN-NFV Security 2016)*. New Orleans, LA. 45–48.

Shinagawa, T., H. Eiraku, K. Tanimoto, K. Omote, S. Hasegawa, T. Horie, M. Hirano, K. Kourai, Y. Oyama, E. Kawai, K. Kono, S. Chiba, Y. Shinjo, and K. Kato. (2009). "BitVisor: A Thin Hypervisor for Enforcing I/O Device Security". In: *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2009)*. Washington, DC. 121–130.

Siemens. (2019). "jailhouse - Linux-based partitioning hypervisor". URL: https://github.com/siemens/jailhouse.

Song, D. X., D. Wagner, and A. Perrig. (2000). "Practical Techniques for Searches on Encrypted Data". In: *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P 2000)*. IEEE Computer Society.

Sprabery, R., K. Evchenko, A. Raj, R. B. Bobba, S. Mohan, and R. H. Campbell. (2017). "A novel scheduling framework leveraging hardware cache partitioning for cache-side-channel elimination in clouds". *arXiv preprint arXiv:1708.09538*.

Stefanov, E., M. van Dijk, A. Juels, and A. Oprea. (2012). "Iris: A Scalable Cloud File System with Efficient Integrity Checks". In: *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC 12)*. ACM. 229–238.

Stefanov, E., M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. (2013). "Path ORAM: An Extremely Simple Oblivious RAM Protocol". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS 13)*.

Steinberg, U. and B. Kauer. (2010). "NOVA: A Microhypervisor-based Secure Virtualization Architecture". In: *Proceedings of the 5th European Conference on Computer Systems (EuroSys 2010)*. Paris, France. 209–222.

Strackx, R. and F. Piessens. (2012). "Fides: Selectively Hardening Software Application Components Against Kernel-level or Process-level Malware". In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS 2012)*. Raleigh, NC. 2–13.

Suzaki, K., K. Iijima, T. Yagi, and C. Artho. (2011). "Memory Deduplication as a Threat to the Guest OS". In: *Proceedings of the Fourth European Workshop on System Security*.

Tao, R., J. Yao, X. Li, S.-W. Li, J. Nieh, and R. Gu. (2021). "Formal Verification of a Multiprocessor Hypervisor on Arm Relaxed Memory Hardware". In: *Proceedings of the 28th ACM Symposium on Operating Systems Principles (SOSP 2021)*. Virtual Event, Germany. 866–881.

TrendForce. (2021). "Intel Responds to AMD's Challenge with Ice Lake CPUs as Competition in Server Market Intensifies". URL: https://www.trendforce.com/presscenter/news/20210318-10723.html.

Tromer, E., D. A. Osvik, and A. Shamir. (2010). "Efficient cache attacks on AES, and countermeasures". *Journal of Cryptology*. 23(1): 37–71.

Tsai, C.-C., K. S. Arora, N. Bandi, B. Jain, W. Jannen, J. John, H. A. Kalodner, V. Kulkarni, D. Oliveira, and D. E. Porter. (2014). "Cooperation and Security Isolation of Library OSes for Multi-process Applications". In: *Proceedings of the 9th European Conference on Computer Systems (EuroSys 2014)*. Amsterdam, The Netherlands. 9:1–9:14.

Tsunoo, Y., T. Saito, T. Suzaki, M. Shigeri, and H. Miyauchi. (2003). "Cryptanalysis of DES implemented on computers with cache". In: *Proceedings of the 2003 International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 62–76.

Varadarajan, V., T. Ristenpart, and M. Swift. (2014). "Scheduler-based defenses against cross-VM side-channels". In: *Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14)*. 687–702.

Varadarajan, V., Y. Zhang, T. Ristenpart, and M. Swift. (2015). "A placement vulnerability study in multi-tenant public clouds". In: *Proceedings of the 24th USENIX Security Symposium (USENIX Security 15)*. 913–928.

Vasudevan, A., S. Chaki, L. Jia, J. McCune, J. Newsome, and A. Datta. (2013). "Design, Implementation and Verification of an eXtensible and Modular Hypervisor Framework". In: *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP 2013)*. San Francisco, CA. 430–444.

Vasudevan, A., S. Chaki, P. Maniatis, L. Jia, and A. Datta. (2016). "überSpark: Enforcing Verifiable Object Abstractions for Automated Compositional Security Analysis of a Hypervisor". In: *Proceedings of the 25th USENIX Security Symposium (USENIX Security 2016)*. Austin, TX. 87–104.

Vaucher, S., R. Pires, P. Felber, M. Pasin, V. Schiavoni, and C. Fetzer. (2018). "SGX-Aware Container Orchestration for Heterogeneous Clusters". In: *Proceedings of the 38th International Conference on Distributed Computing Systems (ICDCS 18)*.

Wan, J., Y. Bi, Z. Zhou, and Z. Li. (2021). "Volcano: Stateless Cache Side-channel Attack by Exploiting Mesh Interconnect". *arXiv preprint arXiv:2103.04533*.

Wang, W., G. Chen, X. Pan, Y. Zhang, X. Wang, V. Bindschaedler, H. Tang, and C. A. Gunter. (2017). "Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*.

Wang, X., H. Chan, and E. Shi. (2015a). "Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound". In: *Proceedings of the ACM Conference on Cloud and Computer Security (CCS 15)*.

Wang, X., A. J. Malozemoff, and J. Katz. (2016). "EMPtoolkit: Efficient MultiParty computation toolkit". URL: https://github.com/emp-toolkit.

Wang, X., Y. Chen, Z. Wang, Y. Qi, and Y. Zhou. (2015b). "SecPod: A Framework for Virtualization-based Security Systems". In: *Proceedings of the 2015 USENIX Annual Technical Conference (USENIX ATC 2015)*. Santa Clara, CA. 347–360.

Wang, Z. and R. B. Lee. (2007). "New cache designs for thwarting software cache-based side channel attacks". In: *Proceedings of the 34th annual international symposium on Computer architecture*. 494–505.

Wang, Z. and R. B. Lee. (2008). "A novel cache architecture with enhanced performance and security". In: *2008 41st IEEE/ACM International Symposium on Microarchitecture*. IEEE. 83–93.

Wang, Z., X. Jiang, W. Cui, and P. Ning. (2009). "Countering Kernel Rootkits with Lightweight Hook Protection". In: *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*. Chicago, IL. 545–554.

Wang, Z., C. Wu, M. Grace, and X. Jiang. (2012). "Isolating Commodity Hosted Hypervisors with HyperLock". In: *Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys 2012)*. Bern, Switzerland. 127–140.

Watson, G. J., R. Safavi-Naini, M. Alimomeni, M. E. Locasto, and S. Narayan. (2012). "LoSt: Location Based Storage". In: *Proceedings of the 2012 ACM Cloud Computing Security Workshop (CCSW 12)*. Association for Computing Machinery. 59–70.

Werner, M., T. Unterluggauer, L. Giner, M. Schwarz, D. Gruss, and S. Mangard. (2019). "Scattercache: Thwarting cache attacks via cache set randomization". In: *Proceedings of the 28th USENIX Security Symposium (USENIX Security 19)*. 675–692.

Williams, P. and R. Sion. (2008). "Usable PIR". In: *Proceedings of the 2008 ISOC Network and Distributed System Security Symposium (NDSS 08)*.

Williams, P. and R. Sion. (2012). "Single Round Access Privacy on Outsourced Storage". In: *Proceedings of the 2012 ACM SIGSAC Conference on Computer and Communication Security (CCS 12)*. Association for Computing Machinery.

Williams, P., R. Sion, and B. Carbunar. (2008). "Building Castles out of Mud: Practical Access Pattern Privacy and Correctness on Untrusted Storage". In: *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 08)*.

Williams, P., R. Sion, and A. Tomescu. (2012). "PrivateFS: A Parallel Oblivious File System". In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS 12)*.

Wu, C., Z. Wang, and X. Jiang. (2013). "Taming Hosted Hypervisors with (Mostly) Deprivileged Execution." In: *20th Annual Network and Distributed System Security Symposium (NDSS 2013)*. San Diego, CA.

Wu, Z., Z. Xu, and H. Wang. (2012). "Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud". In: *21st USENIX Security Symposium (USENIX Security 12)*.

Xiao, J., Z. Xu, H. Huang, and H. Wang. (2013). "Security Implications of Memory Deduplication in a Virtualized Environment". In: *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks. DSN '13*.

Xiao, Y., X. Zhang, Y. Zhang, and R. Teodorescu. (2016). "One bit flips, one cloud flops: Cross-vm row hammer attacks and privilege escalation". In: *Proceedings of the 25th USENIX security symposium (USENIX security 16)*. 19–35.

Xu, Z., H. Wang, and Z. Wu. (2015). "A measurement study on co-residence threat inside the cloud". In: *Proceedings of the 24th USENIX Security Symposium (USENIX Security 15)*. 929–944.

Yang, J. and K. G. Shin. (2008). "Using Hypervisor to Provide Data Secrecy for User Applications on a Per-page Basis". In: *Proceedings of the 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2008)*. Seattle, WA. 71–80.

Yao, A. C. (1986). "How to Generate and Exchange Secrets". In: *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society. 162–167.

Yarom, Y. and K. Falkner. (2014). "FLUSH+ RELOAD: A high resolution, low noise, L3 cache side-channel attack". In: *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 719–732.

Yu, J., L. Hsiung, M. E. Hajj, and C. W. Fletcher. (2019). "Data Oblivious ISA Extensions for Side Channel-Resistant and High Performance Computing". In: *Proceedings of the Annual Network and Distributed System Security Symposium (NDSS 19)*.

Zhang, F., J. Chen, H. Chen, and B. Zang. (2011a). "CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization". In: *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP 2011)*. Cascais, Portugal. 203–216.

Zhang, T., Y. Zhang, and R. B. Lee. (2016). "Cloudradar: A real-time side-channel attack detection system in clouds". In: *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer. 118–140.

Zhang, Y., A. Juels, A. Oprea, and M. K. Reiter. (2011b). "Homealone: Co-residency detection in the cloud via side-channel analysis". In: *Proceedings of the 32nd IEEE Symposium on Security and Privacy (S&P 11)*. IEEE. 313–328.

Zhang, Y., A. Juels, M. K. Reiter, and T. Ristenpart. (2012). "Cross-VM side channels and their use to extract private keys". In: *Proceedings of the 2012 ACM conference on Computer and communications security (CCS 12)*. 305–316.

Zhang, Y., A. Juels, M. K. Reiter, and T. Ristenpart. (2014). "Cross-tenant side-channel attacks in PaaS clouds". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 990–1003.

Zhang, Y. and M. K. Reiter. (2013). "Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud". In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS 13)*. 827–838.

Zhou, Z., M. K. Reiter, and Y. Zhang. (2016). "A software approach to defeating side channels in last-level caches". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 871–882.

Zhou, Z., V. D. Gligor, J. Newsome, and J. M. McCune. (2012). "Building verifiable trusted path on commodity x86 computers". In: *Proceedings of the 33rd IEEE symposium on security and privacy (IEEE S&P 12)*. 616–630.

Zhou, Z., M. Yu, and V. D. Gligor. (2014). "Dancing with Giants: Wimpy Kernels for On-Demand Isolated I/O". In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP 2014)*. San Jose, CA. 308–323.

Zhu, M., B. Tu, W. Wei, and D. Meng. (2017). "HA-VMSI: A Lightweight Virtual Machine Isolation Approach with Commodity Hardware for ARM". In: *Proceedings of the 13th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2017)*. Xi'an, China. 242–256.