Front. Comput. Sci., 2021, 15(3): 153808
https://doi.org/10.1007/s11704-019-9096-y

REVIEW ARTICLE

A survey of Intel SGX and its applications

Wei ZHENG (✉)[1], Ying WU[1], Xiaoxue WU[1], Chen FENG[1], Yulei SUI[2], Xiapu LUO[3], Yajin ZHOU[4]

1 School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710129, China
2 Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney 2007, Australia
3 Department of Computing, The Hong Kong Polytechnic University, Hong Kong 999077, China
4 College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China

Abstract This paper presents a comprehensive survey on the development of Intel SGX (software guard extensions) processors and its applications. With the advent of SGX in 2013 and its subsequent development, the corresponding research works are also increasing rapidly. In order to get a more comprehensive literature review related to SGX, we have made a systematic analysis of the related papers in this area. We first search through five large-scale paper retrieval libraries by keywords (i.e., ACM Digital Library, IEEE/IET Electronic Library, SpringerLink, Web of Science, and Elsevier Science Direct). We read and analyze a total of 128 SGX-related papers. The first round of extensive study is conducted to classify them. The second round of intensive study is carried out to complete a comprehensive analysis of the paper from various aspects. We start with the working environment of SGX and make a conclusive summary of trusted execution environment (TEE). We then focus on the applications of SGX. We also review and study multifarious attack methods to SGX framework and some recent security improvements made on SGX. Finally, we summarize the advantages and disadvantages of SGX with some future research opportunities. We hope this review could help the existing and future research works on SGX and its application for both developers and users.

Keywords Intel SGX, cloud computing, trusted execution environment, TrustZone, AMD SEV

1 Introduction

For a long time, the server platform was considered immune to all types of attacks because the server is kept in a secure computer room and managed by professionals. However, as more and more services are migrated to the cloud, the progressive attack methods and escalating attack types on the data center also arise. A higher level of security is required to avoid potential danger. Security and reliability issues stand out [1].

To address these problems, people associate secure cloud technology with trusted technology and then form a credible cloud technology. A great deal of research has been put forward, including Intel's Trusted Execution Technology (TXT). Some progress has been made [2].

Intel TXT is a set of general hardware extensions for its dedicated processors and chipsets to enhance hardware security and help prevent software-based attacks. The technology is implemented by combining a hardware-based security device and trusted platform module (TPM) [3,4], to provide integrity metrics, sealed storage, protected I/O, protected display buffering and so on. After that, the confidentiality and integrity of the data stored in the user's computers can be improved.

Intel introduced a new technology - SGX (software guard extensions), based on TXT in 2013. SGX is the first group of security extensions introduced in Intel Skylake CPU architecture. By applying a new instruction set and memory access mechanism, it provides users with a trusted execution environment (TEE) for their applications [5].

With the advent of Intel SGX processor, Intel is trying to prove that SGX will eliminate the security problems in the cloud environment with assisted hardware [6,7].

SGX also has obvious shortcomings. The biggest drawback is that the developers need to refactor the code and divide the program into trusted and untrusted parts. Currently, Intel has released the SDK to assist with this work, but the amount of work is still heavy and can easily lead to secret leaks. To address these security insecurities and further improve performance, people have tried different ways to put the program as a whole in the enclave to avoid code refactoring. Important works include Haven, SCONE, and Graphene-SGX [8–10]. In addition, some side-channel attacks can lead to the leakage of sensitive information, e.g., cache attack [11], which also shows that SGX has a big room for improvement.

The research on SGX is still in the stage of development and has not reached a mature level. What's more, it is a challenge for users to be fully skilled in using SGX technology, partly because they lack a deep understanding of it. Our goal is to collect and collate different literature to demonstrate the existence and presentation of a continuous and available research field. By searching, classifying, analyzing and summarizing relevant papers, we give a summary of the application, attack methods and improvements of SGX. We mainly focus on the application

Received March 19, 2019; accepted September 4, 2019

E-mail: wzheng@nwpu.edu.cn

of SGX.

The rest of the paper is organized as follows. Section 2 presents the search sources of papers and classifies them after preliminary reading. Section 3 explains the technical principles of Intel TXT and SGX, we also compare three popular TEEs and list the results meanwhile. Section 4 describes the application of SGX in different fields, we distinguish SGX application papers in terms of unrelated-to-cloud and related-to-cloud aspects, each one has one or more significant highlight in the subject. Section 5 presents some attacking methods of SGX. For the current side-channel attacks against SGX technology, cache attacks and page-table attacks are common. Section 6 concludes several directions of improving SGX, including optimization from different levels like code-level, system-level, hardware-level or development of specific tools. Section 7 gives related work and Section 8 concludes for the whole review. We summarize the advantages and disadvantages of SGX, also introduce our future research directions and research sites in Section 8, which may provide readers with new ideas.

2 Background and literature review

In this section, we describe the process of searching for papers, including which libraries we search for, what keywords we use and the selection criteria.

Search repositories. To obtain a wide enough coverage and few omissions, we collect SGX-related literature from the five most popular academic libraries, they are ACM Digital Library, IEEE/IET Electronic Library, SpringerLink, Web of Science, and Elsevier ScienceDirect.

Terms. We use "SGX" and "enclave(s)" as keywords to search, for the former is the subject of the review and the latter is the prominent feature of itself, which are two certain words for describing. Any paper that mentions these two words in title, abstract or keywords will be considered.

Selection criteria. This paper makes an overview of Intel SGX. Therefore, SGX-themed papers belong to the category we summarize, including papers on SGX research, expansion, and improvement. We have some screening criteria for the retrieved papers, papers with the following conditions would be removed from the collection, that is to say, these papers do not meet our conditions and would not be read in detail:

1) The "SGX" mentioned in some papers' abstracts is not Intel Software Guard Extension but a part of codes or identifiers. This kind of paper that has nothing to do with SGX but containing the letter "SGX" will not be considered.

2) Papers that are unrelated to computer science or do not conform to the subject and papers that do not apply or improve SGX are removed.

3) Duplicate data. A paper can be retrieved in two or even more databases at the same time, in which case the duplicate data is deleted.

Search results. We set the starting time of the retrieval papers as of January 2013, which is the earliest time to find the available papers in the library since SGX was officially proposed in 2013. The collection time of the papers is up to October 2018, and then we start to analyze the collected papers. There are 735 raw records selected from the 5 databases with keywords "SGX" and "enclave(s)".

Table 1 Raw data statistic

Keywords	Refined	Org	Error	Dup.
All	128	735	315	159
"SGX"	124	400	127	30
"enclave(s)"	4	335	188	129

Refined: number of intensive reading papers; Org: original number of papers; Error: including ErrField, ErrTopic, NotPaper; Dup.: duplicate records within a data source or other data sources.

In the first round, we screen the 735 collected papers by reading the abstract and filter out the items that are not related to "Intel SGX". Finally, there are 128 records remained. The statistics of these papers are shown in Table 1, in which column 2 (Refined) presents the 128 articles we selected.

We then conduct the second round review of the 128 papers selected. By reading their title, introduction and conclusion parts, the articles are classified, and the results of the classification can be found in Fig. 1. (Since a paper may cover more than one fields, the total number of papers in the figure is more than 128)

As shown in Fig. 1, we divide the 128 selected papers into the following five categories according to the aspects they focus on.

- Working principle. Some early papers introduced SGX technology and its principles. They include some review papers and related technical documents of Intel, such as [12, 13]. This aspect is beneficial because it can help beginners to learn SGX technology systematically and provide necessary theoretical support for researchers.
- Application. After preliminary analysis, we found that the application of SGX in the field of cloud security accounts for a huge part. For example, build a fully isolated execution environment for cloud applications based on SGX [8], construct a secure and trusted computing environment for cloud-based big data based on SGX [14]. So we divided the application of SGX into two parts: Unrelated-to-cloud and Related-to-cloud. We will discuss them in detail in Section 4 later.
- Attacks. Many papers have studied the security of SGX. Researchers have designed various kinds of attack methods to carry out experiments and found many vulnerabilities in SGX. Here we need to sort out SGX's security issues to provide warnings for users and researchers.
- Improvement. These papers mainly include the design of attack defense and the improvement of SGX function. Improvements are closely related to applications and attacks.
- Others. Several papers are not in the above fields, but they are helpful for us to understand SGX and complete the writing of this paper.

3 Intel SGX and comparison of several TEEs

3.1 Background of Intel TXT and SGX

Intel TXT (Trusted execution technology) is derived from establishing a stable environment from the start of the system by using specific Intel CPU, dedicated hardware and related firmware. And then it provides a variety of methods for system software to achieve a safer system and better data integrity

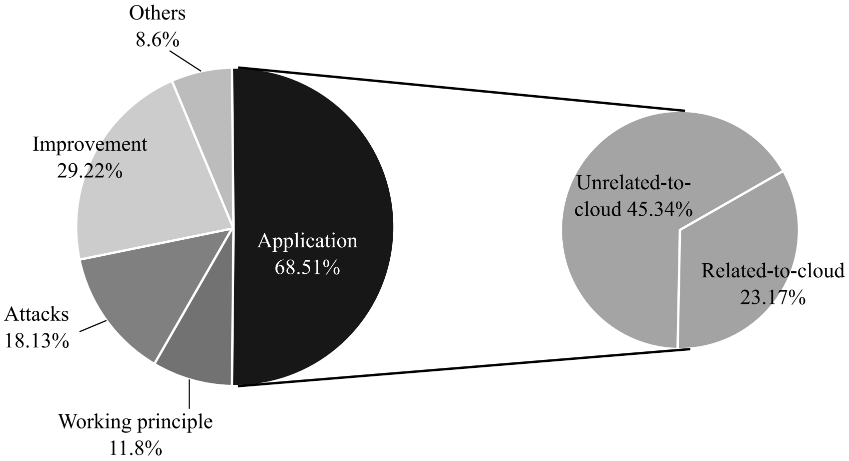


Fig. 1 SGX papers distribution

protection [4, 15].

Generally speaking, Intel TXT is:

1) A collection of security functions [16].

2) A method that validates these functions is being used.

3) The method of security measurement and verification of system configuration and system code.

4) By setting up policies, establish a metric based trust level.

5) An additional security function for trusted operating systems.

Based on TXT, Intel proposed SGX. Enclave created by SGX can also be understood as a trusted execution environment (TEE). As a TEE system based on hardware, SGX was first known in HASP conference in 2013 [17]. Intel officially released SGX2 in 2016 and introduced a physical CPU (Skylake) that supports SGX for the first time [12], which also led to the development of SGX related papers. SGX uniquely gives an execution environment with confidentiality and integrity protection [13] by allowing an application (or part of it) to execute in a secure container called Enclave. An application is divided into two parts depending on the needs: trusted and untrusted. The users can code specific instructions to create an enclave. Sensitive data and related functions are placed in this enclave, while external access to it is forbidden to keep it away from the potential attacks. Sensitive functions are processed in the clear text inside the enclave. The results of the operations are directly fed back to the outer part while involved data remains in the trusted memory space. The enclave and associated data structure are stored in particular protected CPU memory Enclave Page Cache (EPC), and each EPC is only assigned to a single enclave. Access to code and data within the enclave can only be invoked through a given interface. The CPU prevents any non-enclave from accessing the processor reserved memory, and any access outside from enclave will lead to an aborted transaction. Therefore, the areas exposed to attacks have been greatly reduced, which provides a safer environment for private data. Different from other TEE systems which have large trusted computing base (TCB), SGX provides a small TCB occupying CPU and enclave only.

In conclusion, the security features of Intel SGX include the following three points:

1) The application is divided into two parts: Trusted part and Untrusted part.

2) The security operation of sensitive data is encapsulated in an enclave.

3) The untrusted part needs to create and call the enclave function when it wants to access sensitive data. Only enclave's internal code can view its data and always refuse external access. When the call ends, enclave data is left in protected memory.

The operation process is shown in Fig. 2.

3.2 Comparison of Intel SGX, ARM TrustZone and AMD SEV

When talking about Intel SGX, we can compare it with other TEEs like ARM TrustZone and AMD Secure Encrypted Virtualization (SEV). All of them are popular TEEs at present, and each has unique advantages. Table 2 shows the main features of these three TEEs.

We compare them from three perspectives: Key technology, Division of CPU, Difficulty of application. Key technology describes the name of the specific implementation method. Division of CPU describes the partitioning of CPU and other changes. The difficulty of the application describes the technical difficulties when you want to apply the TEEs.

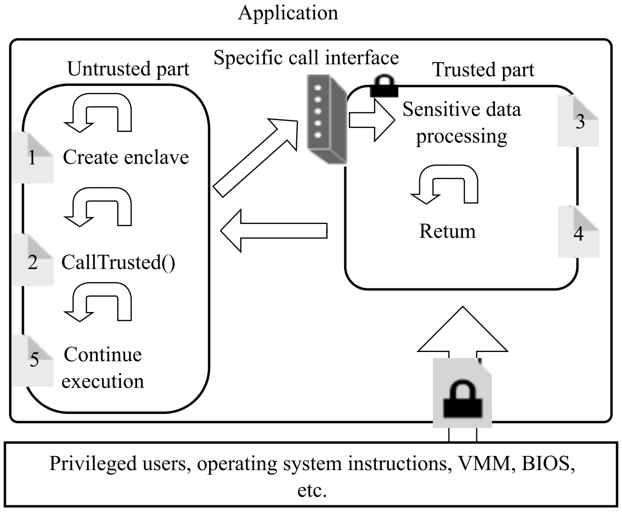


Fig. 2 Execution process of SGX runtime

Table 2 Comparison of Intel SGX, ARM TrustZone and AMD SEV

Domin	Key technology	Division of CPU	Difficulty of application	Description
TrustZone	Monitor Mode	Security and non-security	The world gap between the TrustZone and the monitored Android system makes the TrustZone unable to accurately obtain information about the non-secure world.	TZ divides SoC hardware and software resources into two worlds: Secure World and Normal World. All operations that require confidentiality are executed in the secure world.
AMD SEV	Virtualization SEV SME		The virtual machine stores some data in the main RAM memory, and the main memory's page encryption lacks integrity protection.	SEV protects virtual hosts from malicious cloud service providers by encrypting physical memory data.
Intel SGX	Enclave EPC	Multiple enclaves divide the CPU into different safe locations.	Developers need to refactor the code into trusted and untrusted parts of the program.	SGX encapsulates software security operations in an enclave, protecting them from malware attacks, privileged and unprivileged software cannot access the enclave.

Afterward, we will introduce them separately according to Table 2, and then combine them to analyze their differences in detail.

- ARM TrustZone. TrustZone technology gets along a method of splitting computer resources between two execution worlds, the namely natural world (non-secure world) and the secure world [18, 19], with a mechanism to robustly context switch between them. Being built into System on Chip (SoCs) by semiconductor designers, TrustZone is programmed into the hardware, enabling the protection of memory and peripherals, and also avoids security vulnerabilities caused by proprietary, non-portable solutions outside the core. In this way, the security performance can be maintained without decreasing system capability. The security protection method can be applied into mobile payment or such other applications on the secure kernel [20].
- AMD SEV. The security problem in virtualization technologies is the focus of cloud computing security. The hypervisor could access all the resources of VMs in work, and the resource allocation mechanism of it may introduce unnecessary access to confidential information, which is against the customers' privacy protection requirements [21]. Therefore, AMD chose to provide Secure Encrypted Virtualization (SEV) technology in its processor EPYC to solve this problem by encrypting its memory. AMD has also put forward secure memory encryption (SME) technology to mark a range of memory for encryption by setting a bit in the relevant page-table entries.

TrustZone and SGX have a different mechanism. TrustZone divides the CPU into a secure part and a non-secure one. In the split CPU, the supervisor is responsible for the flow of data between the secure and non-secure part. However, SGX only believes the CPU core. For SGX, one CPU can run multiple security enclaves. Comparing TEE to a public safe box, we can find that TrustZone may contain various objects. It could also include some potentially vulnerable applications. However, in SGX, every application has its unique safe box, and application holds its key. SGX excludes software stacks such as OS and BIOS outside the TCB.

SEV and SGX technologies focus slightly differently, but they all provide a secure, trusted execution environment. SEV protects virtual hosts against malicious cloud service providers by encrypting physical memory data. Although SGX provides fine-grained protection at the application level, it does not include all untrusted elements in enclave. Mofrad et al. [22] compared the security and performance of SGX and AMD. They concluded that SEV is more effective than SGX when encrypting and protecting large-scale data. However, SGX is more reliable when protecting memory data.

However, SGX contains apparent weaknesses in the practical application. For example, developers need to reconstruct the code and divide the program into trusted and non-trusted parts. Although Intel released the SDK to assist in the completion of the reconstruction work, it is still a difficult, complex and vital work. Besides, constructing the enclave could also have an impact on the performance of the system, which is one of the major bottlenecks in using SGX for data processing and security protection. SGX could cause a series of performance loss factors in the processing, such as the cost of the enclave conversion, the system-level function access, large amount leading to EPC page replacement.

4 Application of SGX technology

In the past few years, with the introduction of SGX technology in 2013 and the official released of Intel SGX2 in 2016, the research on its application and improvement have gradually increased. The number of related papers is also proliferating, and the trend seems to be continuing. Due to a large number of papers involved in the application, we do not list all of them. From Fig. 1, we classify papers that we read, which are the application, attacks, improvement, working principle, and others. The unrelated-to-cloud parts are divided into three categories, which are encryption, partition and framework (in Fig. 3). In this chapter, we will discuss papers unrelated-to-cloud in Section 4.1, and those related-to-cloud in Section 4.2.

4.1 Unrelated-to-cloud papers

It has not been long since the concept of SGX was proposed, and the hardware supporting SGX has been introduced. With the maturity of technology, the application of SGX is increasing and deepening. In 2014 and 2015, there were not many papers produced for the SGX study, but SGX research has gradually become popular since 2016. Hardware isolation enforcement and remote attestation are two major security features of SGX. The implementation of hardware isolation is mainly to provide a semi-sandbox mechanism, in which the sensitive and private parts of the application are put into the enclave, such as the non-transparent code. Enclave ensures confidentiality and integrity

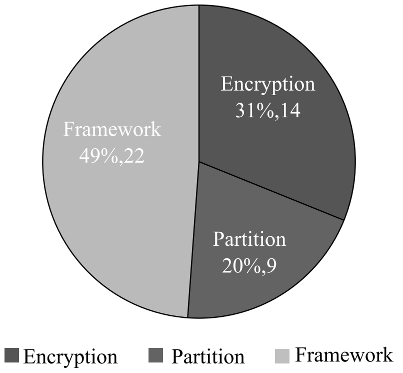


Fig. 3 Unrelated-to-cloud papers distribution

of the trusted part of the application, and it can provide the hardware level of protection. Remote attestation allows the server to determine whether the requesting client is in a secure and trusted environment. The relevant information of the client includes the enclave's information will be sent to the attestation service by the service provider for verification, and the verification result will be returned to the service provider after that. In this way, the service provider can determine whether the client is safe according to the verification results, thus establishing an encrypted channel to facilitate the secure transmission of information.

- Encryption

Enclave is a critical technology in SGX to ensure integrity and security. The enclave is considered as a security zone, and it can prevent some malicious attacks by encrypting secret information in the application. Many researchers combine SGX with existing products, which are usually sensitive to attacks, such as "Tor" [23].

On the premise of keeping TCB small, using SGX to encrypt data to achieve code privacy and integrity is an excellent choice. Fisch et al. [24] proposed Iron, a practical and available Function Encryption (FE) system using Intel SGX. FE allows the authorized entity to calculate the encrypted data and display a precise result. Iron is very secure and can run encrypted data at full processor speed. In the high level of Iron's design, the system uses a Key Manager Enclave (KME), which is responsible for the trusted authority of the master key. The permission to set up a standard public key encryption system and signature scheme. The availability of this public key, which can encrypt data, is not limited by the identity of the user. When the client needs to run a particular function on the data, it can request permission from the KME. After the request has been passed, the KME would release a function key for decryption. Also, the client runs a Decryption Enclave (DE) running on an SGX platform. After the decryption DE would output the function applied to plaintext, then the enclave will clear all relevant state from memory. Iron can be applied to complex functions, and it performs better than known solutions in simple applications. The authors also demonstrated security by modeling the FE in the context of hardware elements and proved that Iron has already met the security model.

FE is a universal and practical encryption method at present. However, SGX is also used in specific areas. Tychalas et al. [25] designed an SGX encrypter to protect the IP of various executable files in Windows from malicious static or dynamic analysis.

- Partition

Some applications use SGX to partition software. Considering the number of enclaves or the size of TCB when partitioning, the software will be divided into several parts according to the requirements to ensure the isolation of the secure and nonsecure contents. Atamli-Reinch and Martin [26] used SGX to design different software partitioning schemes for applications. Complex code and architecture of application lead to the possibility of insecure intrusion. Therefore, some technologies use isolated methods to separate sensitive code from the execution of other software. However, it is easy to get confused about how to make a reasonable division. The situation and granularity of division are both need to be considered. The paper proposed a framework composed of four classification schemes based on the integrity and confidentiality of the enclave. The application program was divided according to the framework, from coarse to ultra-fine granularity. The partitioning scheme is dominated by two rules: the number of available enclaves and the TCB size inside each enclave.

Four partitioning schemes are as follows:

1) Whole application. Part of an application is divided into an enclave, and there is no limit to the size of TCB. Enclave may contain code, data, keys, passwords, and credentials, etc.

2) All secrets. In this case, the author uses two enclaves and divides the code into two parts based on the frequency of access to the secret code.

3) Separate secret. This scheme uses multiple enclaves to protect the secret, each of which is in a separate enclave.

4) Hybrid. The scheme takes the division of scheme 3 and optimizes it by considering reducing the number of enclaves, TCB and code duplication. Besides, the authors also propose the evaluating matrix of the partition scheme.

However, it is noteworthy that the above work only proposed a strategy and only validated the feasibility of the strategy in one example. It did not propose a complete method of partitioning applications using these new strategies to balance security and performance. How can we automatically, or efficiently, find the smallest security subset of an application (only this subset is security-sensitive)? A better solution is to propose an automatic program partition framework to automatically extract the trusted parts of the program to achieve the security isolation of the program. Glamdring [27] is a new framework for protecting C programs using Intel SGX. Glamdring partitions the program at the source level to minimize the amount of code in the enclave. It is the beginning of related works.

- Framework

Applications could be combined with SGX on a holistic level to design a prototype framework to achieve the guarding effect such as [28]:

In [28], Erick et al. attempted to apply SGX to computer games. They designed a prototype framework to integrate SGX with computer games. Their work can be used to prevent players from cheating and obtaining unlicensed copies of the software. The authors developed a game protection model, which could be divided into two types: integrity and confidentiality. The former prevents players from making unauthorized changes by selecting certain components to be placed in the enclave. For example, placing important game data (i.e., player position, score.) into the enclave can prevent simple attacks by modifying the value with a hexadecimal editor. Also, placing the modifiable code into the enclave ensures code integrity. The latter implements Digital Rights Management (DRM) by encrypting key components to prevent players from obtaining unlicensed copies of the software.

Inspired by SGX, Beekman et al. [29] proposed an effective network defense mechanism that can be used by web browser clients. They isolated the service code of all network servers in the enclave. Users can only use TLS to establish a secure channel with the server to transfer data. However, instead of validating their mechanism with SGX, they used CloudProxy [30], because SGX was not available at the time. It is worth noting that Behl et al. [31] applied SGX in the field of fault-tolerant control. They took SGX as the primary trusted subsystem of a prototype implementation and proposed a complex hybrid state machine replication protocol, which significantly improved the speed of fault handling. This may provide new ideas for readers in specific fields.

Concerning the inadequacy of existing hardware-based approaches for search over encrypted data, Fuhry et al. [32] contribute HardIDX for improving search efficiency. HardIDX is architected based on hardware approach, utilizing SGX technology, and deployed as a highly performant encrypted database index. The design of HardIDX includes the trusted client, untrusted SGX enabled server and the trusted SGX enclave within the server. Assuming that the data is composed of key and value, the client completes the construction of B + tree for a key and then encrypts the nodes of value and B + tree. These encrypted values and nodes are stored in random order to prevent the leakage of value and key size or other relationships. Then deploy them together to untrusted servers. When searching, enclave reads into the B + tree, decrypts it, and finishes the key search. The result of the search is a pointer to the encrypted value. After obtaining the encrypted value from these pointers, the search process is completed by returning it to the client.

Priebe et al. [33] proposed that SGX can be used to protect data security in databases. They created EnclaveDB, a more secure database engine that maintains secure attributes even when running on an unsafe cloud host, or encountering a malicious database administrator. EnclaveDB puts all the sensitive information, storage and query operations, transaction processing functions in enclave's TCB. Users only need to create and manage a database encryption key to communicate with their database.

Peters et al. [34] applied SGX to protect data security in Bluetooth I/O process. They presented BASTION-SGX: a Trusted I/O architecture for Bluetooth on SGX. It established a secure channel between trusted software and Bluetooth devices. Store trusted software in the enclave, and then create secure Bluetooth I/O channels to communicate using symmetric keys.

These above works have shown us many practical application scenarios of SGX, such as protection of game data, protection of Web browser, protection of data in database and improvement of data encryption technology, protection of data security in Bluetooth I/O process. However, these are not all. More application scenarios include wise medical, autonomous driving, intelligent robots, and other fields of artificial intelligence. In the optimization of block chains, the combination of Intel SGX technology and block chains can help to ensure the security and privacy in the process of data flow.

4.2 Related-to-cloud papers

Among many issues related to cloud security, how to ensure the security of data processing is the key and difficult point. Research on data processing security mainly includes full memory encryption, CPU-based key storage, the encryption protocol, the enclave. The use of enclave enables data to be encrypted in RAM but is available as plain-text in CPU and CPU caches. Intel SGX introduces the concept of the enclave as part of its software protection extension, and enclave has become the most promising hardware-level protection method for data processing security of cloud tenants.

For cloud service, only the excellent integration of software, service, and hardware infrastructure can provide a satisfactory environment. Virtual machine and migration are two critical concepts in cloud computing, many Related-to-cloud papers explained and applied them to provide a more secure environment for users. For example, Yoo et al. [35] proposed a secure compute-VM, which uses SGX to protect big data computing and confidential information. We have a more detailed classification of the papers we have read, as shown in Table 3.

Gu et al. [38] described real-time migration of virtual machines that support SGX. Under the protection of SGX, traditional migration methods are no longer sufficient. Because the virtual hypervisor cannot access the memory of the enclave and the CPU, the operating state of the virtual machine cannot be transmitted. Therefore, the paper provided a method based on the software, which achieves the virtual machine migration while ensures security. During the enclave migration process, the source machine first transfers the running state of an enclave; then the dumped states are transmitted through the network to the target machine; finally, the target machine creates a new enclave and restores the previous operating state. To solve the problem that the interior of the enclave does not allow external transaction access, authors designed a control thread that runs in each separate enclave to assist in the migration. It can transfer the state within the enclave. The control threads on both the source machine and target machine would establish a secure channel during the migration. When the state of the enclave is suspended, the control thread is responsible for generating a checkpoint containing the current state. At the same time, to prevent malicious attacks, this mechanism set up two checkpoints to ensure consistency and improve security. Besides, to further guard against fork attack and rollback attack, the authors designed remote attestation and self-destruction to ensure that each enclave instance does not roll back or generate multiple instances after migration.

Coughlin et al. [42] proposed a potential architecture for using SGX to increase the privacy of Network Function Virtualization (NFV) applications. Applying the cloud to NFV is an enhancement to itself, but the introducing process also brings

Table 3 Classification of related-to-cloud papers

Class 1	Class 2	Class 3	Class 4
The impact of SGX design flaws on cloud services.	Combining SGX with cloud, put forward new application points and expand new areas.	Use SGX to provide more secure memory for cloud computing and big data.	Migration of SGX-related VMs.
Swami [36]	Sfyrakis et al. [37] Chen et al. [39] Silva et al. [41] Coughlin et al. [42] Alansari et al. [44] Bahmani et al. [46] Bhardwaj et al. [48] Lie et al. [50] Duan et al. [52] Han et al. [53]	Yoo et al. [35] Kelbert et al. [40] Schuster et al. [14] Chakrabarti et al. [43] Nguyen and Ganapathy [45] Brekalo et al. [47] Dang et al. [49] Martin et al. [51]	Gu et al. [38]

security and privacy issues. The authors extended the Click modular router to perform packet processing in the enclave. Click can enhance functionality by combining some common network elements. The external library support is needed to enable the clicked elements of the package to be passed to the enclave. Data can be transmitted in this self-built library, and then it would be processed in the enclave. In SGX-protected, encrypted data transmission, each click module needs to be decrypted and re-encrypted before being transmitted to the next element. Therefore, the Aplomb gateway based on the secure channel is used to complete it. During the establishing of the secure channel, the gateway is responsible for remote attestation of the clicked element, while the gateway represents an SGX enabled system. At the same time, [54] has also been inspired by the SGX architecture to define a remote authentication protocol, but unlike SGX, new participants do not necessarily have a priori identity. It is constructed by combining a mandatory authentication protocol with an arbitrary attested computation protocol. However, Duan et al. [52] questioned that the method of [42] only protects application data (i.e., the payload of transport layer data packets), but ignores the protection of various metadata in low-level data packets, such as packet size, timestamp and so on. Disclosure of such information can also have serious consequences [55]. They designed LightBox, a new SGX-Based network middleware system, which is more secure than previous technologies.

Martin et al. [51] presented an approach for elastic and secure data processing in cloud environments targeting privacy-sensitive applications. Inadequate elastic mechanisms and the lack of security processing mechanisms hinder the development of cloud computing, such as processing energy consumption from smart grids. The paper proposed to use STREAMMING3G in a cloud environment, an elastic and secure data processing engine combined with SGX. The entire processing mechanism can be divided into two parts: elastic data processing and secure data processing. The former is to create a replica of the operator, and then copy the same state to this replica, making them receive the same data. Moreover, timestamps are added to ensure sequential delivery. The original operator would be deleted after the completion of the copying process, leaving the replica alone. The latter is to run the operator code within the enclave to encrypt the operator instance and entities outside the STREMMING3G. In the process of secure data processing, for applications that use primary of the state, the context of the program needs to be evaluated to address the size limitation of the EPC; the modified libmusl is used to protect the enclave from possible malicious attacks. This method does not cause excessive overhead because the SGX can be used on the Skylake CPU.

Kelbert et al. proposed the SecureCloud method in [40] to use the SGX to handle security issues in the cloud from a full stack perspective, which could be applied to the area of the smart grid. SecureCloud is a layered architecture project, providing important functions like the secure creation and deployment of micro-services, the secure integration of personal micro-services and big data applications, the safe execution of applications in non-secure cloud environments. It consists of a set of micro-services connected by an event bus. The application logic for each type of micro-service is within an enclave, and the author deploys the micro-service by providing a secure container at the top of the non-secure stack. Besides, the authors designed and developed SCONE [10], a secure Linux container environment coupled with SGX to protect existing applications. SCONE can perform an integrity check and copy all memory-based return values inside the enclave before passing the parameters to the micro-service. With SecureCloud, users can deploy micro-services and deliver information.

5 Attacking methods of SGX

SGX is not perfect. Over the years, new methods of attacking SGX have been discovered, and improvements have been made to new SGX versions. We read and analyze SGX-related papers that focus on the attacking measures. In these papers, some of them designed and demonstrated their attack measures, and some gave new methods for attack detection while some were challenges and suggestions.

5.1 Attacking methods

The attack vector is a means of attacking a computer or network server, which can help find possible vulnerabilities in the system. These malicious attacks can cause significant harm to the system and software. We review the SGX-related papers and list some common attack vectors as following Table 4 [56, 57]:

5.1.1 Side-channel attacks

In the group of attacks-related papers, side-channel attacking

Table 4 The attack on side-channel information

1.TLB attack (Shared TLBs and paging-structure caches under HyperThreading)
2.Cache attack (CPU caches are shared between code in the enclave and non-enclave mode)
3.Page-table attack (The page table has access control permission to the enclave page; Some status bits in the page table can indicate the status of the enclave page)
4.DRAM attack (Channels, DIMMs, ranks, banks, are shared between code in the enclave and non-enclave mode)
5.Speculative execution attack (Referenced PTEs are cached as data, or pipeline, Branch Prediction Buffer be used)

is the most common and effective. Because attackers cannot directly control enclave, they try to recover the secret information by side-channel.

Side-channel attack [57, 58] is a common and varied attack method. During the execution of the program, some physical state information closely related to internal operations may be leaked, such as acoustic information, resource consumption, electromagnetic radiation, and running hours. Using cryptographic algorithms combined with statistical techniques, the information of side-channels may be exposed, and the program gets invaded. In recent years, the side-channel technology has gradually penetrated the internal CPU, Cache, branch prediction unit, and so on. Attackers can detect these devices to get sensitive information.

1) Cache attacks

Cache attack [11,59,60] is a new type of side-channel analysis technology. It can implement attack across platforms, CPUs, and secure borders, which poses a great threat to current security technologies. In the SGX environment, the attackers exert the advantage of controlling the resources of the entire system, specifically scheduling the resources, reducing the noise of the side-channel, and increasing the success rate of the attack. In addition, the attacker can use Core Isolation, Cache Isolation, and Uninterrupted Execution to reduce noise, or improve the accuracy of the attack by using high-precision clocks and amplifying time differences.

Intel's SGX whitepapers regard cache-timing attacks as unpractical physical attacks while ignoring the case of software-based side-channel attacks. However, Götzfried et al. [11] raised disagreement on this and pointed out that root-level cache-timing attacks are a promising method of attack against software protected. The practically demonstrated that Intel SGX enclaves are vulnerable against cache-timing attacks for the first time.

Schwarz et al. [59] practically performed a Prime+Probe cache side-channel attack on a co-located SGX enclave running a current RSA implementation. They demonstrated them both in a native environment and across multiple Docker containers. In a semi-synchronous attack, they extracted 96% of an RSA private key from a single track, and in an automated attack, they extracted the full RSA private key from 11 traces. These fine-grained software-based side-channel attacks are the first malware running on real SGX hardware, abusing SGX protection features to conceal itself.

2) Page-table based attacks

Page-table based attack [61, 62] is not a relatively new method. Having access to the enclave page set it inaccessible to trigger a page fault exception for any access. This allows the attacker to distinguish which pages were visited by the enclave.

Xu et al. [61] introduced a new type of side-channel attack named "controlled-channel attacks", which allows an untrusted operating system to extract vast amounts of sensitive information from protected applications on systems like Overshadow, InkTag, and Haven, which is implemented based on Intel SGX. They design concrete controlled-channel attacks against widely used libraries and apply these attacks on Haven and InkTag.

Some of the enclave's privacy data can be inferred in chronologic order. Wang et al. [57] summarized the three page-based side-channel attack vectors for SGX and explained the implementation principle of sneaky page monitoring attacks. However, page-based side-channel attacks are implemented at a page granularity, which allows developers to try to resist them by mixing sensitive code and data into the same page.

Van et al. [56] raised again that enclaved execution environments provided by Intel SGX are known to be vulnerable to a group of controlled-channel attacks. Moreover, they also pointed out that the state-of-the-art defense techniques for suppressing page faults during enclave execution could not cover table-based threats. To analyze and elaborate on this problem, they demonstrated that the untrusted operating system could observe enclave page access without resorting to page faults, by exploiting other side effects of the address translation process. They introduced two new attack vectors, which could infer enclaved memory accesses from page table attributes and caching behavior of unprotected page table memory. Compared to the heuristic defenses applied by Costan et al. [63], their new attack vectors went beyond the former in solving the root cause of page table-based information leakage. Their new program still has an advantage in not requiring interruptions.

3) Speculative execution attacks

Speculative execution is a CPU optimization feature. Since branch instruction execution may require a relatively long memory read time, before the end of branch instruction execution, the CPU predicts which branch would run, extract corresponding instruction code and execute it to improve the performance of CPU instruction pipeline. When a speculative execution finds a prediction error, the result of the speculative execution is discarded, and the state of the CPU is reset. However, the impact of speculative execution on the CPU cache is retained. Through some methods of cache attacks to measure the read memory time difference, the attackers can extract the confidential target data from the cache side channel.

Lee et al. [64] explored a new type of side-channel attack named "branch shadowing". It revealed fine-grained control flows (branch granularity) in an enclave. The branch history is not cleaned while switching from enclave to non-enclave mode, so the traces could be obtained from outside, giving a possible side-channel for branch-prediction.

In addition, the Sgxpectre attack [65], which exploits cache side channel information, has been widely proven to be very useful. The Spectre patch released by Intel does not work very well because attackers can still design variants to bypass these fixes. Similar technologies include Foreshadow [66] and Foreshadow-NG [67], they can read the memory contents of the entire SGX enclave. They are more potent than Sgxpectre because they can extract enclave memory without relying on any code vulnerabilities in the victim enclave, or even without executing the affected enclave. However, Sgxpectre relies on the vulnerable code. The defense of side-channel attack is still the focus of SGX security research.

5.1.2 Enclave execution mechanism attacks

In addition to side-channel attacks, some works showed several other types of attacks, mainly targeting the enclave execution mechanism.

Weichbrodt et al. [68] showed that neutral synchronization bugs in enclave code could turn into sever security that helps an attacker to hijack the enclave's control. In their threat model, they assumed that an attacker has full access to the OS and can start and stop enclaves. They proposed the AsynchShock tool, a semi-automated tool for thread manipulation that exploits the use-after-free and time-of-check-to-time-of-use (TOCTTOU) synchronization bugs inside an enclave. They demonstrated how these bugs could be combined with interrupts which are coming from the malicious OS and create a threat. An example of exploiting the use-after-free bug is that they used objdump command to find where the free function is located in the code page, using AsynchShock's registered signal handler, they manipulated the enclave's page access permissions to trigger a segmentation fault for the free function and then initiated a hijacked process to exploit the bug.

Lee et al. [69] designed an attack called "Dark-ROP" to demonstrate the aftermath and harm of memory leaks in enclave code. This technology used return-oriented programming (ROP) to construct a brand-new attack that may ultimately destroy SGX security. The authors created several oracles to tell the attacker the state of the enclave to perform, making the attack feasible with the invisible data and code. Besides, they fully controlled the execution environment by entering the enclave's data into a shadow program. The enclave can perform some operations that are beneficial to the attack, such as reading the SGX encryption key. Dark-ROP may destroy the enclave's memory protection and make internal enclave data unavailable. However, Biondo et al. [70] pointed out the disadvantage of Dark-ROP attack, which requires a constant and non-randomized memory layout. So once developers adopt SGX code randomization techniques, such as SGX-Shield [71], attacks are ineffective. They proposed a new code reuse attack that using two types of gadgets can overcome the SGX-Shield.

5.2 Defense mechanisms

In this category, we list the papers that designed and implement tools for detecting vulnerabilities that SGX could not protect or improving SGX application safety.

5.2.1 Detection mode

Researchers examined enclave's scheduling mode, or they designed tools to detect attacks on SGX to achieve higher levels of security.

Sinha et al. [72] proposed to focus on the vulnerabilities in SGX applications, such as misuse of SGX instructions, memory security errors. So they designed Moat to verify the confidentiality of applications running on SGX. They implemented that from the following aspects: i) formally model primitives for trusted computing, ii) formally verify properties of so-called enclave programs that use them; iii) create formal models of relevant aspects of SGX and develop several adversary models, vi) present a verification methodology for proving that an enclave program running on SGX does not contain a vulnerability that causes it to reveal secrets to the criminal. This paper introduced a technique for verifying information flow properties of enclave programs.

In SGX the shielded execution cannot guarantee software from side-channel attacks [73]. Chen et al. [58] designed a tool for detecting privileged side-channel attacks in shielded execution, which could augment the power of SGX. Moghimi et al. [60] designed a tool named CacheZoom to analyze the memory access of SGX enclave. SGX makes CacheZoom combining an L1 cache Prime+Probe attack with the operating system available, which could get a maximal resolution by the realistic OS of the enclave. This tool could virtually track all memory accesses of SGX enclaves with high spatial and temporal precision. CacheZoom could destroy memory protection and restore AES keys from the enclave.

5.2.2 Enhancement to resist attack

An effort has also been applied to the improvement of SGX applications. The following papers pointed out the possible attacks on SGX and made improvements. Most of them adopt the methods of software separation or hardware isolation, which physically make the environment be trustworthy and not. The researchers came up with their tools and combined them with SGX to make it more secure and less vulnerable to malicious attacks.

Shinde et al. [62] pointed out that SGX is vulnerable to several side-channel capacities of extracting bits of encryption keys from commodity implementations in OpenSSL and Libgcrypt and showed that 27% on average and up to 100% of the secret bits in studied cases could be leaked. They designed a software-only defense that masks page fault patterns by determining the program's memory access behavior for mitigating data leakage by page fault attacks. The approach could be built into a compiler and implemented for a subset of C, which is sufficient to handle the cryptographic routines of their study.

Chandra et al. [74] pointed out that SGX still has shortcomings in defending against side-channel attacks, such as inferring private information through cache access, CPU usage, and other timing channel equilateral channels. They proposed a new randomization method for processing side channel information to avoid being collected by the adversary.

Strackx and Piessens [75] proposed a set of additive security primitives for protected-module architectures like Intel SGX to avoid exploiting vulnerabilities while these systems crash, reboot, or lose power. Their additional security measures ensured that: 1) a protected state can never be rolled back to a previous stale state, 2) accepting an input, the module must eventually finish processing the input or never start processing it, and 3) an unexpected power loss should never cause the system not to start after the reboot. Following these primitives, they proposed a mathematical solution to the state continuity problem which lies behind these primitives and implements Ariadne, a library providing continuous state storage (libariadne). They tackled an essential quirk of SGX technology where SGX enclaves are destroyed when the system is suspended or hibernated. In their threat model, the enclaves are running on top of an untrusted operating system in which its entire software stack is compromised, and the attacker can halt the enclave's execution at any moment in time. They claimed that adding their implementation to the existing SGX instruction set will fix the state-continuity problem of SGX.

Costan et al. proposed a method of attacking SGX [76] and gave out suggestions to improve: a software isolation scheme. Sanctum is a co-design that combines minimal and minimally invasive hardware modifications with a trusted software security monitor that is amenable to formal verification. The authors used conventional building blocks to achieve software isolation without modifying any major CPU building block.

Sasy et al. [77] proposed the shortcomings of SGX vulnerability to side-channel attacks, high context switching costs, and limited memory. The weakness in its architectural limitations and security brings a burden in using. Therefore, they designed and built a memory controller called ZeroTrace, which combined the advantages of RAM and SGX, and became an efficient, flexible, block-level memory controller. ZeroTrace simultaneously achieved a dramatic acceleration of purely encrypted with high performance.

Völp et al. [78] demonstrated how the delayed preemption (DP) simplifies the implementation of the leakage-free code in systems. They illustrated that the construction of leakage-free code for enclaves is impractical when the adversaries have fine-grained control over preemption timing. They used the DP system to provide hardware extensions for a system such as the SGX and achieve efficient leak-preventing measures.

Weiser and Werner [79] proposed another attack point, SGX can protect the security of user applications, but because many traditional I/O devices do not support encrypted transmission, this may lead to data leakage in unsafe input paths. They designed an architecture SGXIO, which is a small SGX-based management program, and established a trusted path to protect users from logical attacks on applications and I/O channels in untrusted operating systems.

In the remaining articles, [80] presented challenges on applying SGX enclaves, [57] presented a systematic review of SGX attack vectors and aimed at inspiring the follow-up efforts to understand better the fundamental limitations of SGX and the ways we can use it effectively and securely.

6 Improvement methods of SGX

We discuss the application of SGX and the attack mode against SGX in Section 4 and Section 5. Both of them indicated that SGX is not in a perfect state, still leaving disadvantages to eliminate and space to improve. The papers covered in this section may be the new application of SGX or the defenses against the attacks. It may also be the improvement of SGX which optimize the function. Improvements could be summarized from the following aspects. As part of the papers have been analyzed in the previous section, the details of the implementations of the program are no longer fully discussed, and we focus only on the improved classification and effectiveness.

6.1 Taxonomy of optimization methods

In the analysis of the paper, the optimization process can be focused on two points: optimization of SGX itself, and optimization of the application.

- Optimization to SGX. Such optimizations are more common in code-focused improvements, where researchers try to improve SGX by modifying its architecture or add more constraints, and specific implementations may include enhancements to algorithms or improvements to methods. Developers chose to use system features to enhance SGX to make it more secure and convenient.
- Optimization of applications. This situation is more common in applications that are combined with SGX, or applications are developed using SGX as a framework. Evolved tools tend to get better results at lower overheads or add new features. Usually, the application of SGX will refer to some software or hardware level isolation. In some instances, the software can be divided into security or non-security part, to prevent the non-security part from interacting with the security part.

Besides, the optimization of SGX can be observed in two directions: automatic performance optimization and auxiliary tool development. The former covers [9, 43, 45], etc. Developers have improved their performance by making improvements to SGX, resulting in higher efficiency with lower overhead. The latter can be seen in [27, 81–83], developers optimize it from a tool perspective and extend the application of SGX.

6.2 Papers about optimization to SGX

The performance problem is one of the main bottlenecks of data processing security protection with SGX. SGX's mechanism determines the performance loss factors generated by its application process, including enclave conversion overhead, the additional overhead caused by system-level function instruction access, and performance overhead caused by EPC page replacement due to large data volume. For security reasons, SGX does not allow system-level function instructions to operate within the enclaves, so each system function call incurs the overhead of additional enclaves in and out of the conversion. Many scholars have analyzed the problem and given optimization strategies.

Tian et al. [84] pointed out that SGX will cause CPU performance loss. The reason is the frequent enclave switch. SGX provides standard functions OCall and ECall for users to get in and out of the enclave. They performed more than 8000 CPU cycles. So researchers used "Switchless Calls" to improve SGX performance by minimizing the number of OCall and ECall calls. Meanwhile, Brenner et al. [85] also proved that SGX improves security while sacrificing processor performance.

For Windows platform applications, Baumann et al. [8] attempted to implement end-to-end SGX protection for existing large systems. To reduce the performance overhead, they adopted in-enclave LibOS, which implemented the Windows 8 API with a set of primitives such as threads, virtual memory, and file I/O. Tsai et al. [9] designed a LibOS called Graphene that supports both single-process and multi-process applications, often with low memory and performance overhead. This design broadens the LibOS paradigm to support secure multi-process APIs.

To some extent, these works alleviated the overhead of SGX protection performance, but still cannot adequately meet the application requirements of SGX in vast and sophisticated systems. LibOS is one way to reduce performance overhead. However, in such an architecture, the application is bundled with a large TCB with millions of lines of code. Within enclave, simulating operating system logic, SGX's EPC memory threshold can result in a loss of page-table replacement performance. Tian et al. [84] proposed the method of switchless system instruction invocation, which has significant performance optimization effect in the case of a single process and a single thread, but the application effect is not ideal in the case of multi-thread, and it is not suitable for multi-process operation mode.

To solve various problems when executing SGX paging in Virtual Machine Monitor (VMM), such as space wasting, the complexity of VMM algorithm, Chakrabarti et al. introduced SGX Oversubscription Extensions in [43], which added additional instructions and virtualization support to SGX architecture. In this architecture three most critical components count: i) determining the customer EPC layout; ii) resolving the conflicting problem of redistributive assignment of enclaves; and iii) preventing the misallocated SECS page from not having children. This solution addressed the extension of paging instructions and solved the problem of the current difficulty of virtualizing SGX memory. These instructions had a significant role in reducing the overhead and complexity of oversubscription of EPCs. This article may provide some new ideas for future researchers to solve the performance loss caused by the EPC memory threshold.

Nguyen and Ganapathy [45] pointed out that the confidentiality of enclaves makes the contents of itself not transparent to cloud providers, which hinders the implementation of the cloud provider's policy in an enclave. Therefore, they designed EnGarde, an enclave inspection library that can meet cloud provider implementation policies. This approach not only ensures that customers' sensitive content is protected by SGX but also ensures that customers can use the services provided by cloud providers in the enclave. It gives a significant boost to the effective integration of SGX with cloud services.

Boneh and Gueron [81] provided surnaming, which can verify digital signatures within a limited validation time, giving the possibility of rapid verification of SGX. This new hash-based approach can replace the current RSA-based method used by SGX and can also be extended to post-quantum security methods.

6.3 Papers about optimization to application

Private Membership Testing (PMT) can eliminate privacy violations caused by malware inspections, but the current PMT solution is still inadequate in a large number of concurrent users and high query arrival rates. Tamrakar et al. [86] proposed a PMT method using a carousel: Circle the entire dictionary with trusted hardware on the cloud server, which can be implemented on SGX. They detected why they experienced a decrease in performance by using a new set of microbenchmarks. Later they designed a new synchronization spin-lock interface of SGX and named it HotCalls. This interface, while being easily integrated into program code, can also provide a 13-27x speedup. Experiments have shown that HotCalls outperforms the original interface in terms of throughput and reduced latency.

At present, the trusted services provided by SGX still lack some application support to protect their state from rollback and forking attacks. To solve this shortcoming, Brandenburger et al. [83] introduced a lightweight protocol, namely LVM. It can detect rollback attacks with less computational overhead, and it complements the simple client.

Lind et al. introduced Glamdring in [27] to solve the problem of excessive TCB generated by placing the whole program into the enclave, which violated the principle of least privilege [87]. The authors used Glamdring, a source-level partitioning framework that secures applications, to partition the application and apply data flow analysis and reverse slicing to ensure data confidentiality and integrity. The Glamdring then placed sensitive features only in the enclave and added running checks and cryptographic operations at the enclave boundary. This type of processing no longer has unacceptable performance overhead, enabling a small TCB.

7 Related work

Much work has been done on studying the Intel SGX technology. For example, researchers' opinions in [4] introduced trusted execution technology. The increasingly popular new technology has attracted more attention, and at the same time, there is a higher demand for security. In this case, TPM is gradually produced. Depending on the service provided by TPM, TXT starts to play its part. Papers like [12, 13, 17–19, 21] respectively introduced several popular TEEs: ARM TrustZone and SGX. They have different strengths in maintaining system security.

As for ARM TrustZone, it is a highly secure system architecture designed through a reasonable combination of hardware and software and has minimal impact on power consumption and performance. This technology has a lot of technical and commercial advantages in improving the security of embedded systems. At present, many security solutions based on TrustZone have been introduced in the industry, and the academic circles have also researched this technology. For example, TrustZone-based security platform and security service are built to meet the security payment, fingerprint identification, and other functions [88–91]. The hardware security isolation advantage of TrustZone can be used to guarantee the security of the operating system [92, 93]. TrustZone hardware security mechanism can be used to implement embedded trusted computing environment [94]. TrustZone can also enhance the security of the existing software virtualization technology and overcome its shortcomings [95]. TrustZone is an effective solution to improve system security, but it also has some shortcomings, such as better defense against software attacks but challenging to prevent physical attacks, as well as side-channel attacks, reverse engineering and other types of attacks [96].

8 Conclusion and future work

8.1 Conclusion

SGX is a credible enforcement technology that is not too trendy, but the research of it is still on the road. This comprehensive paper first analyzes the trusted environment and then compares the SGX technology with several other trusted technologies, such as TrustZone and AMD SEV, giving the features and advantages of SGX. We review the application of SGX and show some typical application scenarios. Subsequently, we carry on a simple classification to the SGX attack methods. We demonstrate the efforts made by researchers in improving and make a classification and summary based on the methods of improvement. We also present the tools that the authors of the papers have implemented in improving. In general, our research paper carries out a comprehensive analysis of SGX from several aspects and provides some reliable information for the research and application of SGX.

8.2 Future research directions

Although this paper has reviewed the research progress of SGX, there are still some open research challenges and problems, and we list as follows:

- A general solution to reduce memory overhead and enhance communication security between enclaves while using SGX technology. The data and code in the Enclave have an overhead of data encryption and decryption as they pass in and out of the processor. Since SGX currently only supports the EPC size of the enclave page cache with a maximum of 128 MByte, if the data size of the Enclave application code exceeds 128 MByte, page missing exceptions will occur, which will also incur considerable overhead. However, different application system types and application scenarios have different effects on performance loss. Performance data can be viewed as a multidimensional function varying with acquisition time. According to queuing theory, the performance of a system is not only determined by the current state but also affected by the past time. Utilizing pattern theory and deep learning, conducting supervised learning on data and studying the relationship between performance data, researchers can propose an SGX performance estimation model to solve the problem of estimating performance loss in SGX protection strategy design.
- As one of the new solutions to cloud security issues, SGX provides a new perspective and strategy for rethinking our approach to cloud security. However, the experimental results of the performance analysis of SGX have shown that the overhead of SGX runtime is enough to affect the performance of cloud applications. Given the considerable performance costs and the complexity of breaking down the application structure into multiple parts, SGX seems ill-suited for today's cloud environments. To summarize, one of SGX's open challenges for large-scale, complex cloud applications is how to map an application to enclaves to provide the best balance of TCB size, performance, and data security. Future work can focus on finding an effective solution to the challenges posed by SGX performance issues in the cloud. It is expected to analyze the balance between security improvement and cost faced by SGX protection and its characteristics, and design a multi-objective optimization algorithm oriented to the characteristics of the problem. Through experiments to improve and optimize the algorithm, future work may achieve a flexible and configurable SGX protection strategy design for cloud tenants.
- Research automation support of SGX technology and apply code automation analysis tools. In the current SGX mode, the code of any application should be divided into two parts: trusted code components and untrusted code components. This feature of SGX requires the application designer to redesign or refactor the application to conform to the guidelines. However, this requirement is unrealistic for broad applications. Therefore, future research needs to focus on how to reduce redundant engineering work as applications migrate to SGX environments. For example, without reconstructing the original application, the code data with high-security sensitivity in the application can be automatically identified and generated to support SGX technical code.
- Achieve seamless integration of SGX technology with services deployed on cloud computing platforms, and leverage SGX to implement more cloud services that are currently technically difficult. For example, although many cloud storage systems allow users to use encryption to protect their data, only a few support collaborative editing of data. One of the main challenges in enabling this collaboration is how to enforce the encryption access control strategy securely and effectively and solve the synchronization and security problems of collaborative editing.
- Solve the problem that SGX technology is easy to be attacked by side-channel. Researchers need to try different defenses for different types of side-channel attacks. Furthermore, the difference in severity between different attack types has not been compared and summarized. Future research work can establish a threat level evaluation model of side channel attack on SGX. Establish a framework for evaluating and validating side-channel attacks that are classified by threat level and allow for automatic evaluation of new attacks.
- Look for some typical application scenarios of SGX technology in real life. For example, the copyright party of multimedia content has stricter security protection requirements for players running on terminal devices. Therefore, it may be possible to use the SGX to secure the broadcast television on-demand system and prevent the attack of the key of the video file.

Acknowledgements This study was supported by Fund of Shaanxi Science and Technology Research and Development Plan Project (2015GY073). Shaanxi Key Research and Development Program (2019GY-057).

References

1. Lou Y, Wang W. The research of trusted technology under cloud environment. In: Proceedings of International Conference on Information Sci-

ence and Cloud Computing Companion. 2013, 231–235

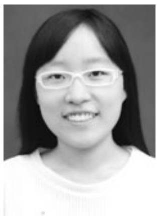
2. Liu C Y, Feng M, Dai X J, Li D Y. A new algorithm of backward cloud. Acta Simulata Systematica Sinica, 2004, 16(11): 2417–2420
3. Hayes B. Cloud computing. Communications of the ACM, 2008, 51(7): 9–11
4. Futral W, Greene J. Intel Trusted Execution Technology for Server Platforms: A Guide to More Secure Datacenters. Apress, 2013
5. Ning Z, Zhang F, Shi W. Position paper: challenges towards securing hardware-assisted execution environments. In: Proceedings of the Hardware and Architectural Support for Security and Privacy. 2017
6. Pei Z, Ruan D, Liu J, Xu Y. A linguistic aggregation operator with three kinds of weights for nuclear safeguards evaluation. Knowledge-Based Systems, 2012, 28: 19–26
7. Meng D, Pei Z. Extracting linguistic rules from data sets using fuzzy logic and genetic algorithms. Neurocomputing, 2012, 78(1): 48–54
8. Baumann A, Peinado M, Hunt G. Shielding applications from an untrusted cloud with haven. ACM Transactions on Computer Systems (TOCS), 2015, 33(3): 8
9. Tsai C C, Porter D E, Vij M. Graphene-SGX: a practical library OS for unmodified applications on SGX. In: Proceedings of USENIX Annual Technical Conference. 2017, 645–658
10. Arnautov S, Trach B, Gregor F, Knauth T, Martin A, Priebe C, Lind J, Muthukumaran D, O'keeffe D, Stillwell M. SCONE: secure linux containers with Intel SGX. In: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). 2016, 689–703
11. Götzfried J, Eckert M, Schinzel S, Müller T. Cache attacks on Intel SGX. In: Proceedings of the 10th European Workshop on Systems Security. 2017
12. McKeen F, Alexandrovich I, Anati I, Caspi D, Johnson S, Leslie H R, Rozas C. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In: Proceedings of the Hardware and Architectural Support for Security and Privacy. 2016
13. Xing B C, Shanahan M, Leslie H R. Intel® software guard extensions (Intel® SGX) software support for dynamic memory allocation inside an enclave. In: Proceedings of the Hardware and Architectural Support for Security and Privacy. 2016
14. Schuster F, Costa M, Fournet C, Gkantsidis C, Peinado M, Mainar R G, Russinovich M. VC3: trustworthy data analytics in the cloud using SGX. In: Proceedings of IEEE Symposium on Security and Privacy. 2015, 38–54
15. Shepherd C, Arfaoui G, Gurulian I, Lee R, Markantonakis K, Akram R, Sauveron D, Conchon E. Secure and trusted execution: past, present and future – a critical review in the context of the internet of things and cyber-physical systems. In: Proceedings of IEEE Trustcom/BigDataSE/ISPA. 2016, 168–177
16. Wang J, Hong Z, Zhang Y, Jin Y. Enabling security-enhanced attestation with Intel SGX for remote terminal and IoT. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(1): 88–96
17. Hoekstra M, Lal R, Pappachan P, Phegade V, Del Cuvillo J. Using innovative instructions to create trustworthy software solutions. HASP@ ISCA, 2013, 11
18. Ngabonziza B, Martin D, Bailey A, Cho H, Martin S. Trustzone explained: architectural features and use cases. In: Proceedings of the 2nd IEEE International Conference on Collaboration and Internet Computing. 2016, 445–451
19. Platform G. Global platform made simple guide: trusted execution environment (tee) guide. Derniere Visite, 2013
20. Kobayashi T, Sasaki T, Jada A, Asoni D E, Perrig A. SAFES: sand-boxed architecture for frequent environment self-measurement. In: Proceedings of the 3rd Workshop on System Software for Trusted Execution. 2018, 37–41
21. Du Z H, Ying Z, Ma Z, Mai Y, Wang P, Liu J, Fang J. Secure encrypted virtualization is unsecure. 2017, arXiv preprint arXiv:1712.05090
22. Mofrad S, Zhang F, Lu S, Shi W. A comparison study of intel SGX and AMD memory encryption technology. In: Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy. 2018
23. Kim S, Han J, Ha J, Kim T, Han D. SGX-Tor: a secure and practical tor anonymity network with SGX enclaves. IEEE/ACM Transactions on Networking, 2018, 26(5): 2174–2187
24. Fisch B, Vinayagamurthy D, Boneh D, Gorbunov S. Iron: functional encryption using Intel SGX. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. 2017, 765–782
25. Tychalas D, Tsoutsos N G, Maniatakos M. Sgxcrypter: IP protection for portable executables using Intel's SGX technology. In: Proceedings of the 22nd Asia and South Pacific Design Automation Conference. 2017, 354–359
26. Atamli-Reineh A, Martin A. Securing application with software partitioning: a case study using SGX. In: Proceedings of International Conference on Security and Privacy in Communication Systems. 2015, 605–621
27. Lind J, Priebe C, Muthukumaran D, O'Keeffe D, Aublin P L, Kelbert F, Reiher T, Goltzsche D, Eyers D, Kapitza R. Glamdring: automatic application partitioning for Intel SGX. In: Proceedings of USENIX Annual Technical Conference. 2017, 285–298
28. Bauman E, Lin Z. A case for protecting computer games with SGX. In: Proceedings of the 1st Workshop on System Software for Trusted Execution. 2016, 1–6
29. Beekman J G, Manferdelli J L, Wagner D. Attestation transparency: building secure internet services for legacy clients. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. 2016, 687–698
30. Manferdelli J, Roeder T, Schneider F. The cloudproxy tao for trusted computing. Technical Rep. UCB/EECS-2013-135, 2013
31. Behl J, Distler T, Kapitza R. Hybrids on steroids: SGX-based high performance BFT. In: Proceedings of European Conference on Computer Systems. 2017, 222–237
32. Fuhry B, Bahmani R, Brasser F, Hahn F, Kerschbaum F, Sadeghi A R. HardIDX: practical and secure index with SGX. In: Proceedings of IFIP Annual Conference on Data and Applications Security and Privacy. 2017, 386–408
33. Priebe C, Vaswani K, Costa M. EnclaveDB: a secure database using SGX. In: Proceedings of IEEE Symposium on Security and Privacy. 2018, 264–278
34. Peters T, Lal R, Varadarajan S, Pappachan P, Kotz D. BASTION-SGX: bluetooth and architectural support for trusted I/O on SGX. In: Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy. 2018, 1–9
35. Yoo S, Kim H, Kim J. Secure compute-VM: secure big data processing with SGX and compute accelerators. In: Proceedings of the 3rd Workshop on System Software for Trusted Execution. 2018, 34–36
36. Swami Y. Intel SGX remote attestation is not sufficient. IACR, Cryptology ePrint Archive, 2017
37. Sfyrakis I, Gross T. UniGuard: protecting unikernels using Intel SGX. In: Proceedings of IEEE International Conference on Cloud Engineering. 2018, 99–105
38. Gu J, Hua Z, Xia Y, Chen H, Zang B, Guan H, Li J. Secure live migration of SGX enclaves on untrusted cloud. In: Proceedings of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. 2017, 225–236
39. Chen F, Wang C, Dai W, Jiang X, Mohammed N, Al Aziz M M, Sadat M N, Sahinalp C, Lauter K, Wang S. PRESAGE: privacy-preserving genetic testing via software guard extension. BMC Medical Genomics, 2017, 10(2): 48
40. Kelbert F, Gregor F, Pires R, Köpsell S, Pasin M, Havet A, Schiavoni V, Felber P, Fetzer C, Pietzuch P. SecureCloud: secure big data processing in untrusted clouds. In: Proceedings of the Conference on Design, Automation & Test in Europe. 2017, 282–285
41. Silva L V, Barbosa P, Marinho R, Brito A. Security and privacy aware

data aggregation on cloud computing. Journal of Internet Services and Applications, 2018, 9(1): 6

42. Coughlin M, Keller E, Wustrow E. Trusted click: overcoming security issues of NFV in the cloud. In: Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization. 2017, 31–36
43. Chakrabarti S, Leslie-Hurd R, Vij M, McKeen F, Rozas C, Caspi D, Alexandrovich I, Anati I. Intel® software guard extensions (Intel® SGX) architecture for oversubscription of secure memory in a virtualized environment. In: Proceedings of the Hardware and Architectural Support for Security and Privacy. 2017
44. Alansari S, Paci F, Sassone V. A distributed access control system for cloud federations. In: Proceedings of the 37th IEEE International Conference on Distributed Computing Systems. 2017, 2131–2136
45. Nguyen H, Ganapathy V. EnGarde: mutually-trusted inspection of SGX enclaves. In: Proceedings of the 37th IEEE International Conference on Distributed Computing Systems. 2017, 2458–2465
46. Bahmani R, Barbosa M, Brasser F, Portela B, Sadeghi A R, Scerri G, Warinschi B. Secure multiparty computation from SGX. In: Proceedings of International Conference on Financial Cryptography and Data Security. 2017, 477–497
47. Brekalo H, Strackx R, Piessens F. Mitigating password database breaches with Intel SGX. In: Proceedings of the 1st Workshop on System Software for Trusted Execution. 2016
48. Bhardwaj K, Shih M W, Agarwal P, Gavrilovska A, Kim T, Schwan K. Fast, scalable and secure onloading of edge functions using airbox. In: Proceedings of IEEE/ACM Symposium on Edge Computing. 2016, 14–27
49. Dang H, Purwanto E, Chang E C. Proofs of data residency: checking whether your cloud files have been relocated. In: Proceedings of the ACM on Asia Conference on Computer and Communications Security. 2017, 408–422
50. Lie D, Maniatis P. Glimmers: resolving the privacy/trust quagmire. In: Proceedings of the 16th Workshop on Hot Topics in Operating Systems. 2017, 94–99
51. Martin A, Britoy A, Fetzer C. Elastic and secure energy forecasting in cloud environments. 2017, arXiv preprint arXiv:1705.06453
52. Duan H, Yuan X, Wang C. Lightbox: SGX-assisted secure network functions at near-native speed. 2017, arXiv preprint arXiv:1706.06261
53. Han J, Kim S, Ha J, Han D. SGX-Box: enabling visibility on encrypted traffic using a secure middlebox module. In: Proceedings of the 1st Asia-Pacific Workshop on Networking. 2017, 99–105
54. Barbosa M, Portela B, Scerri G, Warinschi B. Foundations of hardware-based attested computation and application to SGX. In: Proceedings of IEEE European Symposium on Security and Privacy. 2016, 245–260
55. Coull S E, Dyer K P. Traffic analysis of encrypted messaging services: apple imessage and beyond. ACM SIGCOMM Computer Communication Review, 2014, 44(5): 5–11
56. Van B J, Weichbrodt N, Kapitza R, Piessens F, Strackx R. Telling your secrets without page faults: stealthy page table-based attacks on enclaved execution. In: Proceedings of the 26th USENIX Security Symposium. 2017, 1041–1056
57. Wang W, Chen G, Pan X, Zhang Y, Wang X, Bindschaedler V, Tang H, Gunter C A. Leaky cauldron on the dark land: understanding memory side-channel hazards in SGX. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. 2017, 2421–2434
58. Chen S, Zhang X, Reiter M K, Zhang Y. Detecting privileged side-channel attacks in shielded execution with Deja Vu. In: Proceedings of the ACM on Asia Conference on Computer and Communications Security. 2017, 7–18
59. Schwarz M, Weiser S, Gruss D, Maurice C, Mangard S. Malware guard extension: using SGX to conceal cache attacks. In: Proceedings of International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. 2017, 3–24
60. Moghimi A, Irazoqui G, Eisenbarth T. CacheZoom: how SGX amplifies the power of cache attacks. In: Proceedings of International Conference on Cryptographic Hardware and Embedded Systems. 2017, 69–90
61. Xu Y, Cui W, Peinado M. Controlled-channel attacks: deterministic side channels for untrusted operating systems. In: Proceedings of IEEE Symposium on Security and Privacy. 2015, 640–656
62. Shinde S, Chua Z L, Narayanan V, Saxena P. Preventing page faults from telling your secrets. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. 2016, 317–328
63. Costan V, Devadas S. Intel SGX explained. IACR, Cryptology ePrint Archive, 2016, 2016(086): 1–118
64. Lee S, Shih M W, Gera P, Kim T, Kim H, Peinado M. Inferring fine-grained control flow inside SGX enclaves with branch shadowing. In: Proceedings of the 26th USENIX Security Symposium. 2017, 16–18
65. Chen G, Chen S, Yuan X, Zhang Y, Lai T H. SgxPectre attacks: leaking enclave secrets via speculative execution. 2018, arXiv preprint arXiv:1802.09085
66. Van B J, Minkin M, Weisse O, Genkin D, Kasikci B, Piessens F, Silberstein M, Wenisch T F, Yarom Y, Strackx R. Foreshadow: extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In: Proceedings of the 27th USENIX Security Symposium. 2018
67. Weisse O, Van B J, Minkin M, Genkin D, Kasikci B, Piessens F, Silberstein M, Strackx R, Wenisch T F, Yarom Y. Foreshadow-NG: breaking the virtual memory abstraction with transient out-of-order execution. Technical Report, 2018
68. Weichbrodt N, Kurmus A, Pietzuch P, Kapitza R. AsyncShock: exploiting synchronisation bugs in Intel SGX enclaves. In: Proceedings of European Symposium on Research in Computer Security. 2016, 440–457
69. Lee J, Jang J, Jang Y, Kwak N, Choi Y, Choi C, Kim T, Peinado M, Kang B B. Hacking in darkness: return-oriented programming against secure enclaves. In: Proceedings of USENIX Security Symposium. 2017, 523–539
70. Biondo A, Conti M, Davi L, Frassetto T, Sadeghi A R. The guard's dilemma: efficient code-reuse attacks against Intel SGX. In: Proceedings of the 27th USENIX Security Symposium. 2018, 1213–1227
71. Seo J, Lee B, Kim S M, Shih M W, Shin I, Han D, Kim T. SGX-shield: enabling address space layout randomization for SGX programs. In: Proceedings of Network and Distributed System Security Symposium (NDSS). 2017
72. Sinha R, Rajamani S, Seshia S, Vaswani K. Moat: verifying confidentiality of enclave programs. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 2015, 1169–1184
73. Buhren R, Hetzelt F, Pirnay N. On the detectability of control flow using memory access patterns. In: Proceedings of the 3rd Workshop on System Software for Trusted Execution. 2018, 48–53
74. Chandra S, Karande V, Lin Z, Khan L, Kantarcioglu M, Thuraisingham B. Securing data analytics on sgx with randomization. In: Proceedings of European Symposium on Research in Computer Security. 2017, 352–369
75. Strackx R, Piessens F. Ariadne: a minimal approach to state continuity. In: Proceedings of the 25th USENIX Security Symposium. 2016, 875–892
76. Costan V, Lebedev I A, Devadas S. Sanctum: minimal hardware extensions for strong software isolation. In: Proceedings of the USENIX Security Symposium. 2016, 857–874
77. Sasy S, Gorbunov S, Fletcher C W. ZeroTrace: oblivious memory primitives from Intel SGX. In: Proceedings of Symposium on Network and Distributed System Security. 2017
78. Völp M, Lackorzynski A, Decouchant J, Rahli V, Rocha F, Esteves V P. Avoiding leakage and synchronization attacks through enclave-side preemption control. In: Proceedings of the 1st Workshop on System Software for Trusted Execution. 2016, 1–6
79. Weiser S, Werner M. SGXIO: generic trusted I/O path for Intel SGX. In: Proceedings of the 7th ACM on Conference on Data and Application Security and Privacy. 2017, 261–268
80. Strackx R, Piessens F. Developing secure SGX enclaves: new challenges on the horizon. In: Proceedings of the 1st Workshop on System Software for Trusted Execution. 2016

81. Boneh D, Gueron S. Surnaming schemes, fast verification, and applications to SGX technology. In: Proceedings of Cryptographers' Track at the RSA Conference. 2017, 149–164
82. Weisse O, Bertacco V, Austin T. Regaining lost cycles with HotCalls: a fast interface for SGX secure enclaves. ACM SIGARCH Computer Architecture News, 2017, 45(2): 81–93
83. Brandenburger M, Cachin C, Lorenz M, Kapitza R. Rollback and forking detection for trusted execution environments using lightweight collective memory. In: Proceedings of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. 2017, 157–168
84. Tian H, Zhang Q, Yan S, Rudnitsky A, Shacham L, Yariv R, Milshten N. Switchless calls made practical in Intel SGX. In: Proceedings of the 3rd Workshop on System Software for Trusted Execution. 2018, 22–27
85. Brenner S, Behlendorf M, Kapitza R. Trusted execution, and the impact of security on performance. In: Proceedings of the 3rd Workshop on System Software for Trusted Execution. 2018, 28–33
86. Tamrakar S, Liu J, Paverd A, Ekberg J E, Pinkas B, Asokan N. The circle game: scalable private membership test using trusted hardware. In: Proceedings of ACM on Asia Conference on Computer and Communications Security. 2017, 31–44
87. Saltzer J H, Schroeder M D. The protection of information in computer systems. Proceedings of the IEEE, 1975, 63(9): 1278–1308
88. Pirker M, Slamanig D. A framework for privacy-preserving mobile payment on security enhanced ARM TrustZone platforms. In: Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. 2012, 1155–1160
89. Smalley S, Craig R. Security enhanced (SE) Android: bring flexible MAC to Android. In: Proceedings of the 20th Annual Network and Distributed System Symposium. 2013, 20–38
90. Zheng C. Overview of security Enhanced Android's security architecture. In: Proceedings of the 2nd International Conference on Teaching and Computational Science. 2014
91. Liu R, Srivastava M. PROTC: PROTeCting drone's peripherals through ARM trustzone. In: Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications. 2017, 1–6
92. Fitzek A, Achleitner F, Winter J, Hein D. The ANDIX research OS-ARM TrustZone meets industrial control systems security. In: Proceedings of the 13th IEEE International Conference on Industrial Informatics. 2015, 88–93
93. Ying K, Ahlawat A, Alsharifi B, Jiang Y, Thavai P, Du W. TruZ-Droid: integrating TrustZone with mobile operating system. In: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services. 2018, 14–27
94. Winter J. Trusted computing building blocks for embedded linux-based ARM trustzone platforms. In: Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing. 2008, 21–30
95. Jia L, Zhu M, Tu B. T-VMI: trusted virtual machine introspection in cloud environments. In: Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. 2017, 478–487
96. Cho H, Zhang P, Kim D, Park J, Lee C H, Zhao Z, Doupé A, Ahn G J. Prime+ count: novel cross-world covert channels on arm trustzone. In: Proceedings of the 34th Annual Computer Security Applications Conference. 2018, 441–452

Wei Zheng is an associate professor in the School of Software and Microelectronics at University of Northwestern Polytechnical University, China. His current research interest focused on cloud computing, big data security, and software quality assurance. He has published more than 60 papers, including many of the top papers in the field of software engineering (such as TOSEM, FSE, ICSE, etc.).



Ying Wu received the BS degree from the school of Software and Microelectronics, Northwestern Polytechnical University, China. She is currently studying for a master's degree in this faculty. Her research interests include information security and software engineering theory.

Xiaoxue Wu is currently a PhD candidate at Department of Automaiton, University of Northwestern Polytechnical, China. She received the MS degree in Software Engineering from University of Northwestern Polytechnical, China. Her main research interests are security testing and interactive machine learning. She has pubished over 10 articles in journals, conferences, and book chapters.

Chen Feng is a master degree candidate in the School of Automation at University of Northwestern Polytechnical University, China. Her current research is in software security testing, big data security and machine learning.

Yulei Sui is a faculty member at University of Technology Sydney (UTS), Australia. He is broadly interested in the research field of software engineering and programming languages, particularly interested in static and dynamic program analysis for software bug detection and compiler optimizations. He has been awarded an ICSE Distinguished Paper and a CGO Best Paper and an Australian Discovery Early Career Researcher Award (DECRA) 2017-2019.

Xiapu Luo is an assistant professor with the Department of Computing and an associate researcher with the Shenzhen Research Institute, The Hong Kong Polytechnic University, China. He received the PhD degree in Computer Science from The Hong Kong Polytechnic University, China, and was a post-doctoral research fellow with the Georgia Institute of Technology, USA. His current research focuses on smartphone security and privacy, network security and privacy, and Internet measurement.

Yajin Zhou received the PhD degree in computer science from North Carolina State University, USA. He is currently a ZJU 100 Young Professor with the Institute of Cyber Security Research and the College of Computer Science and Technology, Zhejiang University, China. His research mainly focuses on smartphone and system security.