Research on Arm TrustZone and Understanding the Security Vulnerability in Its Cache Architecture

Pengfei Guo(✉), Yingjian Yan, Chunsheng Zhu, and Junjie Wang

School of Cryptographic Engineering, Strategic Support Force Information Engineering University, Zhengzhou 450001, China
guoyifei322@gmail.com

Abstract. Arm TrustZone technology is the most widely used system-level security framework, which provides a trusted execution environment (TEE) for embedded system SoC. This paper introduces the technical principle of TrustZone in detail, explains how to extend the security features from CPU to the whole system through various security components, and briefly introduces the secure boot, the application of TrustZone in mobile devices and the variant technology based on TrustZone. At the same time, there are many attacks on TrustZone. According to the cache architecture of TrustZone, each cache line uses a NS bit to indicate whether the line belongs to secure world or normal world. The purpose is to avoid refreshing when switching the two worlds, thus reducing the performance loss. However, it supports cache lines in the two worlds to compete and evict each other, this provides an opportunity for cache attacks, and this article details this security Vulnerability.

Keywords: Secure isolation · Arm TrustZone · TEE · Cache architecture · Cache attacks

1 Introduction

With the development of electronic information technology, mobile devices (such as mobile phones, BYOD) have reached the computing power and network bandwidth comparable to traditional PCs, and have now become an important tool for people to handle daily life and work affairs. The distribution of apps is mostly based on app stores, and there are many untrusted third-party apps. At the same time, even officially certified apps may have the risk of stealing user privacy, because people often perform mobile payment, fingerprint recognition, digital rights management and other data processing involving key sensitive information on mobile devices [1]. In our daily life, we may all have the impression that you have just used search engines (such as Google, Baidu) to search for a book, and when you open a shopping app, the book is automatically recommended to you, indicating that "it" monitors your behavior in the background.

At present, the two most commonly used means to solve information security problems are encryption and isolation [2]. Encryption is a very effective traditional means,

G. Wang et al. (Eds.): SpaCCS 2020, LNCS 12382, pp. 200–213, 2021.
https://doi.org/10.1007/978-3-030-68851-6_14

but in some scenarios, encryption needs the effective cooperation of isolation to give full play to its real power. The core idea of isolation is not to completely identify or completely remove malware or vulnerabilities, but only to provide a trusted execution environment for critical data or services, which can greatly reduce the attack interface. The goal of isolation is to protect sensitive code from attack and destruction in a complex environment where system vulnerabilities cannot be completely avoided.

TrustZone proposed by Arm is a typical representative of system-level isolation scheme, and has been widely recognized and applied. We all know that there is no absolute security in the world, and inevitably, TrustZone has its vulnerabilities. At present, many studies have proven that TrustZone can be successfully cracked [3], and cache attacks are a representative one.

In this article, we mainly focus on two parts, one is the system principle of TrustZone, introduced in Sect. 3, and the other is how to exploit the vulnerability in the TrustZone cache architecture to implement cache attacks, introduced in Sect. 4. The Sect. 2introduces several commonly used security isolation techniques, and the last section, Sect. 5, is about the conclusion.

2 Security Isolation Technology

There are three common isolation mechanisms: hardware isolation, software isolation and system-level isolation.

Hardware isolation mechanisms are usually reflected in the form of external encryption modules, such as the early IBM4758 encryption coprocessor, which is mainly used to verify authorization services, banking and financial systems and industrial control fields. Mobile phone SIM card and set-top box IC card, which are closely related to our daily life, can also be regarded as a kind of hardware isolation module, but it is lightweight application. This kind of card form module is mainly used for identity authentication, secure storage, financial services and so on. The external hardware security module can protect the sensitive data in a relatively secure physical peripheral, and can adopt more advanced technology and physical security protection technology according to the security requirements, but its disadvantage is that it increases the cost of the design, increases the power consumption of the system, and the scope of the external security hardware module is also limited. The limited communication bandwidth between the two domains will also reduce the performance of the system.

Now the popular TPM (Trusted Platform Module) developed and advocated by TCG (Trusted Computing Group) can also be regarded as an external hardware security module, its main task is to establish a hardware-based trust root, from the infrastructure to provide the necessary mechanism for trusted computing environment [4]. With reference to the TPM1.2 architecture, China has launched a self-developed Trusted Cryptography Module (TCM) specification, which changed the module from passive invocation to active control, and also supports national secret algorithms SM1, SM2, SM3, SM4.

Software isolation technology is mainly to use virtualization technology to achieve the purpose of isolation, such as the use of MMU to achieve the isolation of the address space, the use of privilege management mechanism to achieve the separation of the operating system kernel mode and user mode. In addition, there are software isolation

technologies such as containers, sandboxes, and honeypots. Software isolation technology usually uses the entire core as a trusted computing base (TCB), which has a large attack interface and many vulnerabilities. At the same time, it lacks the root of trust of the hardware, and its own security is difficult to be guaranteed. In addition, DMA and GPU with bus master interface function IP can bypass some software security isolation mechanisms and bring additional security risks.

Both hardware and software isolation mechanisms have their own problems and defects, so many researchers focus on how to enhance the security of the system kernel and applications through the dual-domain execution environment provided by the architecture, and propose a system-level security isolation. The mechanism can also be called the isolation protection mechanism of software and hardware cooperation. TrustZone technology is a typical representative among them. This system-level isolation mechanism can provide considerable isolation and flexibility.

The ultimate goal of the above three isolation mechanisms is to build a trusted execution environment. For ease of comparison, the corresponding execution environment that can provide rich functions is generally called REE. TEE is a trusted environment for secure executing applications. TEE itself does not require that it must be implemented based on hardware. However, in practice, it is found that TEE without hardware support is difficult to ensure its security, therefore, the researchers classify the hardware-based TEE schemes into the following three types, as shown in Fig. 1.

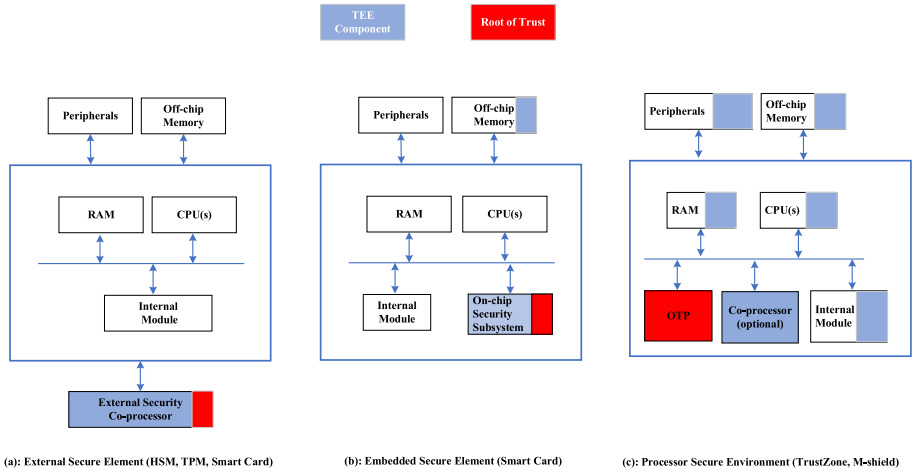

Fig. 1. Implementation scheme of TEE based on hardware.

From left to right, the first scheme is the external security module mentioned above, and the second scheme is the built-in security module, but this scheme does not have the ability to extend the security domain to the whole system, and the third scheme is the system-level solution. In comparison, the third scheme takes into account both flexibility and security, and is mainstream in both industry and academia.

3 Detailed Explanation of TrustZone Technology

3.1 Overview

Arm proposed the TrustZone technology as early as 2004 [5]. This technology realizes the isolation between security environment and normal environment through hardware without affecting the power consumption, performance and area of the system as far as possible. The software provides basic security service interface, and the system security is constructed by the combination of software and hardware. The TrustZone architecture extends the secure isolation mechanism to the entire system instead of a single module [6]. The overall hardware architecture is shown in Fig. 2 [7].

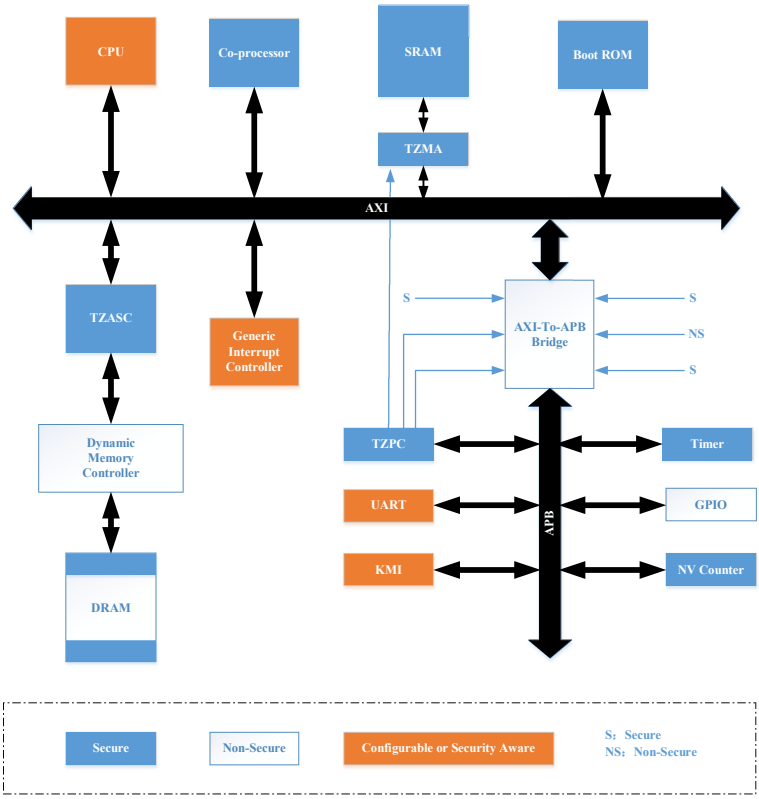

Fig. 2. An example of the overall hardware architecture of TrustZone.

In the early days when this technology was proposed, TrustZone did not get widespread attention and application, maybe people's security awareness is not strong enough, or it is limited by the cost of security design. Fortunately, in the past 10 years, the design concept of TrustZone has been widely recognized and applied on a large scale. To some extent, TrustZone has become a de facto TEE standard reference, especially in the field of embedded systems.

3.2 Security Extension of TrustZone Architecture [8, 9]

CPU. TrustZone technology virtualizes a single physical CPU into two virtual CPU, one is called secure processor core, the other is called normal processor core, the running environment of secure processor is called secure world or security domain, and the running environment of normal processor core is called normal world or normal domain. The two worlds run in a time-sharing manner, and to some extent, the two worlds can be regarded as two big processes. The security extension of CPU is realized through hardware, and the security state of the processor is characterized by the NS bit of SCR registers. NS bits can only be configured in the secure world. At the same time, NS bit can also be extended to each IP through the system-on-chip bus. Therefore, the normal world cannot access the resources unique to the secure world, while the secure world can access all the resources of the system, and CPU will clear the register status of the previous world when switching between the two worlds.

There is a secure "gateway" between the two worlds, which is used to switch between the two worlds and save (restore) the context, known as the monitor. Normal world, which can fall into monitor mode through interrupts, exceptions, and SMC instructions. When the secure world enters the monitor mode, it is more flexible. It can directly write CPSR registers, and of course, it can also be switched to the normal world through interrupts and exception mechanisms.

On-chip Bus. On-chip bus is an important support for secure extension. Two secure extension bits are introduced into AXI (Advanced eXtensible Interface) bus. These two bits can be equivalent to the 33rd address line for read/write transactions, respectively.

AWPROT [1]: bus write transaction control signal, secure write transaction is low level, non-secure write transaction is high level.

ARPROT [1]: bus read transaction control signal, secure read transaction is low level, non-secure read transaction is high level.

The security extension of AHB (Advanced High-Performance Bus) bus can be realized through AXI-To-AHB bridge, in implementation, two AHB buses are usually used to separate secure components and non-secure components.

The APB (Advanced Peripheral Bus) bus is usually used to connect low-speed peripherals, but it does not have the security extension attribute, and the security extension is responsible for by the AXI-To-APB bridge. Each peripheral connected to the bridge has a separate TZPCDECPROT input signal, which is used to determine whether the peripheral is configured as secure or non-secure. These input signals can be fixed and connected during design synthesis, or can be dynamically controlled by the security peripheral TrustZone protection controller (TZPC), so that the security state can be dynamically switched during operation. TZPC itself is also a peripheral on APB, but TZPC is fixed as a security peripheral. In Fig. 1, TZPC, Timer and Non-Volatile Counter are fixed with "0" when designing synthesis, indicating that it is always secure, while GPIO is fixed with "1" when designing synthesis, indicating that it is always non-secure state. The security state of UART and KMI (Keyboard and Mouse Interface) is configurable, and the control signals is given by TZPC.

MMU, TLB and Cache. The processors provide an MMU for each of the two worlds, each corresponding to a virtual processor, which ensures that there is a separate page table structure in the two worlds. Arm does not provide a mandatory isolation standard for TLB, but it is recommended that vendors add a bit to the TLB descriptor according to their specific needs to identify whether the content in the TLB belongs to the normal world or the secure world, so as to avoid refreshing the TLB each time the world is switched. There is also a bit added in the CPU cache to indicate the current security status of the cache accessed by the CPU, so that the caches of the normal world and the secure world can coexist to avoid refreshing the cache during switching. When a cache replacement occurs, the replacement policy does not consider whether the cache line is in a secure state or not, a secure cache line may evict a non-secure cache line, and a non-secure cache line may evict a secure cache line. This provides an opportunity for cache side-channel attacks.

Interrupts and Exceptions. The mode recommended by Arm is to use IRQ as the interrupt source for the normal world and FIQ as the interrupt source for the secure world. If a corresponding interruption occurs in the current world, there is no need to switch the world, otherwise, the world needs to be switched to handle the current interruption. For example, if an IRQ interrupt occurs in the normal world, there is no need to switch the world, if an FIQ interrupt occurs, you need to switch to the secure world to handle the FIQ interrupt.

TZMA (TrustZone Memory Adapter). TZMA divides the on-chip ROM or RAM into security domains. Compared with providing separate memory for the secure world and the normal world, it saves cost and area. TZMA can support a maximum of 2MB memory area division, where the low address area is the security domain, the high address area is a non-secure domain, and the division of the secure domain and the non-secure domain is aligned with 4KB bytes. The size of the security domain is determined by the input signal R0SIZE. This signal can be fixed during design synthesis or can be given by TZPC and can be dynamically configured.

TZASC (TrustZone Address Space Controller). TZASC is usually mounted on the AXI bus, which can divide multiple storage areas of the address space of the AXI bus from the device. The typical application is to place TZASC between the AXI bus and the dynamic memory controller DMC (Dynamic Memory Controller) to divide the DRAM security domain. It is important to note that TZASC can only be used for partition memory mapping devices, not for partition block devices such as NandFlash.

Thus, through the above security components, the security attributes will be extended from CPU to the entire system, which is why TrustZone is called the system-level isolation architecture.

3.3 Secure Boot

Secure boot is the cornerstone of system security. If the security at startup cannot be guaranteed, then system security is a castle in the air. Secure boot depends on two

technologies, one is hardware-based trusted root technology, and the other is trust chain transfer mechanism.

ROM is non-rewritable and has strong resistance to hardware attacks, so the first stage bootloader (FSBL) is usually solidified in ROM as the trusted root of the whole system, and the rest of the code is stored in the on-chip eFlash. If the code is large, the code of the normal world such as the operating system can be stored in mass memory such as NandFlash.

When secure booting, except for the FSBL stored in ROM by default, other program components need to be verified for integrity and authenticity before running. The method is to calculate the hash value of the program code first, and use the Public Key (PuK) of the trusted manufacturer stored in the chip in advance to check the signature attached to the program, compare the two hash values, and pass if consistent, otherwise, stop booting. The transmission of trust chain can use chain transmission or star transmission, or a combination of both.

Secure boot also faces some risks. The traditional method is to store the PuK in ROM. Because all devices use the same PuK, when the Private Key (PrK) is cracked or reversed, it is vulnerable to class-break attacks. To defend against this attack, the system can use OTP (One-Time-Programmable) ROM to store PuK. This enables different devices to have different PuK, to increase the ability to resist class-break attacks. At present, some scholars use PUF technology to build trusted roots [10]. PUF is the abbreviation of Physical Unclonable Functions, also known as chip fingerprint.

3.4 Application of TrustZone Technology in Mobile Terminal

Instead of providing a fixed one-size-fits-all solution, TrustZone provides an overall framework that allows SoC designers to choose, tailor, or add from a range of components that can implement specific functions in a secure environment. At present, TrustZone technology has been widely used, and the typical application on the mobile side are Apple's Secure Enclave, Qualcomm's QSEE (Qualcomm Secure Execution Environment), Samsung's Knox, Huawei's TrustedCore [8]. This technology is mainly used in face recognition, fingerprint recognition and password protection in mobile phones.

3.5 Variants of TrustZone Technology

Dual-Core Architecture. In the dual-core architecture, one physical CPU is replaced by two physical CPU, one CPU is fixed secure and one CPU is fixed non-secure. This architecture can reduce the switching between the two worlds, and does not require CPU to have TrustZone extension properties, which significantly broadens the application field of TrustZone technology.

Multicore Architecture. In the multi-core architecture, whether it is heterogeneous multi-core or homogeneous multi-core, each core can be configured as a secure core or a normal core, or support the switching between the secure world and the normal world at the same time. In practical applications, considering the complexity of the software, usually only one CPU core has the ability to run the secure world, while other CPU can run in the normal world. Sometimes, in order to improve the utilization of multiple cores

as much as possible, normal cores and secure cores can also be switched dynamically, but this scheme will obviously increase the complexity of the software.

Multi-domain Architecture. Ferdinand Brasser et al. [11] proposed a multi-domain architecture called SANCTUARY. The innovation of this architecture is that the isolation zone is placed in the user's normal world instead of the security world. Multiple isolation zones called SANCTUARY can be instantiated. SANCTUARY can achieve two-way isolation from the normal world and the secure world, the architecture effectively reduces the trusted computing base TCB, and can tolerate malicious SA (SANCTUARY APP). SA is a program that runs in the SANCTUARY zone, which reduces the security risk of malicious TA to the system, and facilitates the secondary development of TrustZone by third parties. But the architecture needs to rely on Arm's latest TZC-400 memory controller and is only applicable to multi-core SoCs.

TrustZone for ARMv8-M. With the evolution of the Cortex-A series instruction set, Arm has also brought TrustZone to the Cortex-M series to build a trusted execution environment on the MCU. The TEE here has the low power consumption of the MCU, and the switching speed of the secure world and the normal world is faster than that of the A series CPU. The introduction of TrustZone on Cortex-M is mainly used for secure boot, firmware security, creating a root of trust, and at the same time can control secure peripherals, such as independent secure storage, random number generator, secure clock, etc. [8]. Although the isolated security world is introduced on the M series, it will not reduce the real-time performance of the service (compared to the deterministic delay of MCUs without TEE). Currently, some manufacturers have produced TrustZone M series chips, such as MicroChip's SAML10 and L11 (based on CortexM23).

4 The Security Vulnerability in TrustZone Cache Architecture

4.1 Memory Hierarchy of TrustZone

The memory hierarchy of TrustZone is shown in Fig. 3. Each cache line extends the NS bit, which specifies the security state of the cache line. The original intention of this design is to use the NS bit to distinguish between the cache line of the normal world and the cache line of the secure world, thereby reducing the system overhead of refreshing the cache during world switching. Any cache line can be evicted to make room for new data regardless of its security status. In other words, secure cache line filling can evict non-secure cache lines, and vice versa. This design is a key factor in launching cache side-channel attack on TrustZone.

4.2 Introduction to Cache Side-Channel Attack

In the field of cryptanalysis, the method of using the side channel information leaked during the implementation of the cryptographic algorithm to attack the key is called the side channel attack. The side information usually refers to power consumption, electromagnetism, sound, fault, photon, time, cache and so on [13].

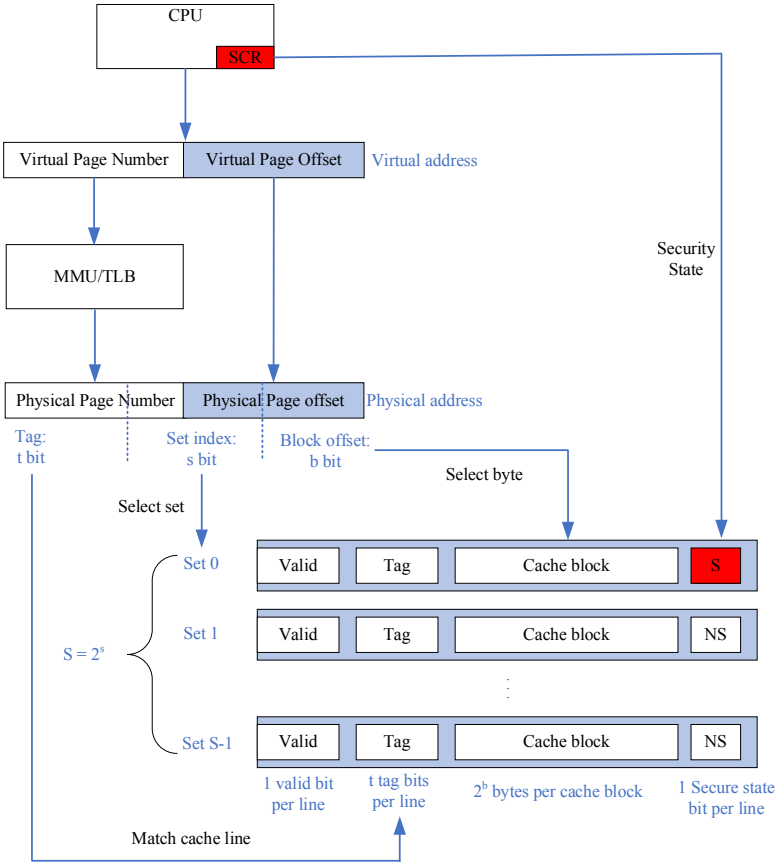

Fig. 3. Memory hierarchy of TrustZone [12].

Cache is designed to solve the mismatch between computing speed and memory speed of CPU, and it is an important means of processor design optimization at present. Table 1 [14] enumerates the access latency at all levels of memory in modern computer systems.

4.3 Cache Basics [15]

Cache Organization. There are three mapping modes between cache and main memory, namely, associative mapping, direct mapping, set-associative mapping. Direct mapping can be regarded as 1-way set-associative mapping, associative mapping can be regarded as a set-associative mapping with only one set, set-associative mapping combines the advantages of direct mapping and associative mapping. Modern caches are usually organized with N-way associative.

Table 1. The ubiquity of caching in modern computer systems.

Type	What cached	Where cached	Latency (cycles)	Managed by
CPU registers	4-byte or 8-byte words	On-chip CPU registers	0	Compiler
TLB	Address translations	On-chip TLB	0	Hardware MMU
L1 cache	64-byte blocks	On-chip L1 cache	4	Hardware
L2 cache	64-byte blocks	On-chip L2 cache	10	Hardware
L3 cache	64-byte blocks	On-chip L3 cache	50	Hardware
Virtual memory	4-KB pages	Main memory	200	Hardware + OS
Disk cache	Disk sectors	Disk controller	100,000	Controller firmware

Cache Replacement Strategy. There are three common replacement strategies for cache, namely least-recently used (LRU), pseudo-random replacement strategy and first-in first-out replacement strategy. Intel processors often use LRU replacement strategies, while Arm processors usually use pseudo-random replacement strategies in order to reduce power consumption and complexity. It is worth noting that the pseudo-random replacement strategy will bring additional difficulty to cache side channel attacks [16].

Addressing Modes. Both the index and tag in the cache address field can use physical addresses or virtual addresses, and there are four addressing modes by arrangement. They are virtually-indexed virtually-tagged (VIVT), Physically-indexed physically-tagged (PIPT), physically-indexed virtually-tagged (PIVT), and virtually-indexed physically-tagged (VIPT). The specific mode is related to the processor.

Inclusiveness. There are three types of inclusiveness in multi-level cache, namely Inclusive cache, Exclusive cache, and Non-inclusive cache. Take two-level cache as an example, Inclusive cache refers to L1 cache is a subset of L2 cache, exclusive cache refers to the data in memory can only exist in a certain level of cache, and cannot exist in multi-level cache at the same time, Non-inclusive cache refers to the above two properties not satisfied. It is worth mentioning that cross-core attacks require an inclusive multi-level cache architecture.

4.4 Cache-Attack Technology

The underlying principle of cache attacks is simple, which is to use the difference in access time caused by whether the data is in the cache. To implement cache attacks, the following conditions need to be met:

- The attacker and the victim share the cache, or at least share a part of the cache.
- The attacker and the victim will have a competitive use of the cache, that is, they can evict each other's cache line.
- After the attacker competes with the victim to use cache, the state of cache can be retained, in other words, the state of cache cannot be refreshed.
- Different states of cache line will lead to differences in execution time, and this time difference can be accurately distinguished by timing means.

The three most common attack techniques are described below [15].

Evict + Time. Osvik et al. [17] put forward the attack method of Evict + Time, which indirectly measures the total execution time of the victim to observe the total number of cache hits and misses. This method is relatively simple to implement, but the signal-to-noise ratio is very low and its practicability is not strong, so it is seldom used at present.

Prime + Probe. Prime + Probe is also proposed by Osvik et al. [17], which is more efficient than Evict + Time, and this method does not require the attacker process and the victim process to share memory. The attack flow is shown in Fig. 4. This method is mainly used in cache attacks on TrustZone.

Flush + Reload. Flush + Reload [18] is a more efficient method of attack, but there are two prerequisites, one is that the CPU needs to have cache refresh instructions, such as the cflush instruction of the Intel processor, and the other is that the attacker process and the victim process share memory. If there is no flush instruction, you can use Evict + Reload instead. Because the two worlds of TrustZone are isolated and it is impossible to share memory, this approach is not suitable for attacks on TrustZone.

The common timing method is to use precise Performance Interface, such as x86 processor's rdtsc instruction and Arm's performance monitor register, but the use of these performance interfaces may require kernel permissions. In addition, you can also use unprivileged system call, POSIX function, dedicated thread timer and other methods to implement timing [15].

4.5 Cases of Cache Attacks on TrustZone

Zhang Ning et al. [12] proposed an attack called TruSpy, the first study of timing-based cache side-channel information leakage of TrustZone. TruSpy used Prime + Probe cache attacks technology to successfully implement the attack in the user mode of the normal world and the kernel mode of the normal world. The difference is that in the user mode, the attacker cannot access virtual-to-physical address translation and high precision timers, which will cause great trouble for cache attacks. TruSpy devise a novel method that uses the expected channel statistics to allocate memory for cache probing, and shows how to implement timing with less accurate performance time interfaces. The attack model is described as follows [12]: Running OpenSSL in the secure world as a security service, fast software-based AES implementation in the OpenSSL 1.0.1f, T-Tables uses precomputed look-up tables. In the OS-based attack, the attacker has full control of the

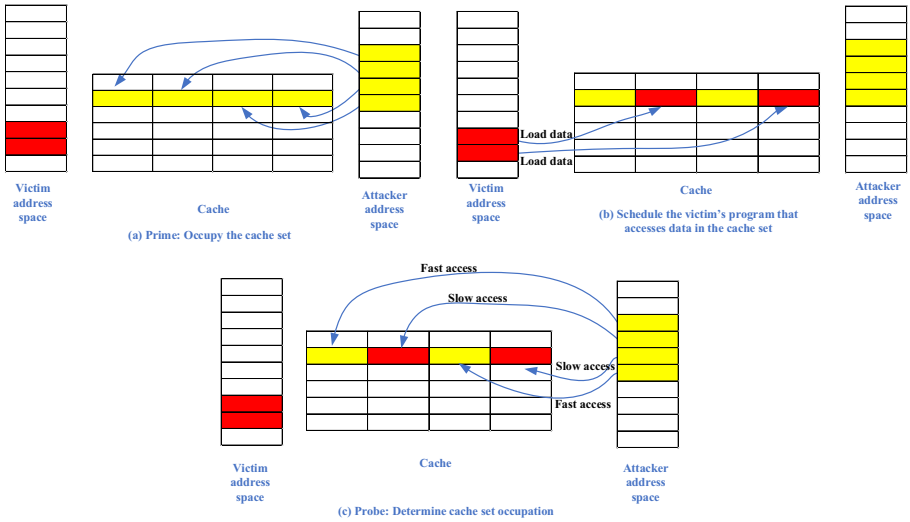

Fig. 4. Illustration of the Prime + Probe attack by means of a 4-way (columns) cache with 6 sets (rows).

normal world, in the app-based attack, the attacker has only user privileges. The goal is to obtain the AES key. To successfully carry out an attack, two conditions need to be met. First, the attack process must be able to fill the cache lines of each cache set, which will cause cache contention with the victim process. Second, the attacker must be able to detect changes in the cache state [19].

TruSpy focuses on the last round of the AES, the last round key can be recovered by taking the XOR of the T-table entry and the cipher text value. There are certain differences in the access time between cache hit and miss. The Prime + Probe attack technology can be used to get which cache line (set) is accessed by the victim, further, we can figure out which entry of the T-Tables the victim accessed (provided that it is known the mapping between the T-table and cache), according to the fast software-based AES algorithm implementation, If we know which entry of T-Tables is accessed, we know the result of $k_i \oplus P$i, in the case of known-plaintext attack, Ki can be determined, and the original key can be deduced from the last round key [20].

MoritzLipp et al. [21] use prime + probe technology to distinguish whether the provided key is valid based on Alcatel One TouchPop2 (with QSEE platform running on it). The key master trustlet on the Alcatel One Touch Pop 2 provides an interface to generate hardware backed RSA keys. In addition, it can be used for the signature creation and verification of data inside the TrustZone.

Ben Lapid et al. [22] only need 1 min to recover the AES-256 key using laptop GPUs for parallel optimization analysis. The attack platform is Samsung's Galaxy S6, which deployed TrustZone framework.

5 Conclusions

Little attention was paid to TrustZone technology when it was first proposed, but now it has been widely used and popular, which not only shows the recognition of this technology by industry and academia, but also shows that people pay more and more attention to security issues. TrustZone technology provides a general, flexible and secure framework for embedded system chips, which can be tailored and customized according to specific needs. In this paper, the technical principle of TrustZone is introduced in detail, and some variants based on TrustZone technology are introduced.

At the same time, we also notice that there are more and more attacks on TrustZone [3], and the means are more and more abundant, in which the cache side channel attack makes use of the vulnerable in the cache architecture, and the attack can be successfully implemented without the help of any external physical devices. Just like Spectre and Meltdown, this kind of attacks belong to the scope of micro-architecture attacks, which have a wide range of influence.

The common countermeasure against cache side channel attacks are still applicable in TrustZone architecture, mainly including cache partition, access randomization, removing high resolution timers, runtime attack detection, cache flushing, cache prefetching [15], but these countermeasures will more or less affect the performance of the system, which requires users to make tradeoffs.

References

1. Enck, W., Ongtang, M., McDaniel, P.: Understanding android security. IEEE Secur. Priv. Mag. **7**, 50–57 (2009)
2. Ravi, S., Raghunathan, A., Kocher, P.C., Hattangady, S.: Security in embedded systems: design challenges. ACM Trans. Embed. Comput. Syst. **3**, 461–491 (2004)
3. Cerdeira, D., Santos, N., Fonseca, P., Pinto, S.: SoK: understanding the prevailing security vulnerabilities in TrustZone-assisted TEE systems. In: IEEE Symposium on Security and Privacy (SP), pp. 1416–1432 (2020)
4. Feng, W., Feng, D., Wei, G., Qin, Y., Zhang, Q., Chang, D.: TEEM: a user-oriented trusted mobile device for multi-platform security applications. In: International Conference on Trust and Trustworthy Computing (TRUST), pp. 133–141(2013)
5. Alves, T.: TrustZone: integrated hardware and software security. Inf. Q. **3**, 18–24 (2004)
6. Ngabonziza, B., Martin, D., Bailey, A., Cho, H., Martin, S.: TrustZone explained: architectural features and use cases. In: International Conference on Collaboration and Internet Computing (CIC), pp. 445–451 (2016)
7. Jang, J., et al.: PrivateZone: providing a private execution environment using ARM TrustZone. IEEE Trans. Dependable Secure Comput. **15**, 797–810 (2018)
8. Pinto, S., Santos, N.: Demystifying ARM TrustZone. ACM Comput. Surv. **51**, 1–36 (2019)
9. Mukhtar, M.A., Bhatti, M., Gogniat, G.: Architectures for security: a comparative analysis of hardware security features in intel SGX and ARM TrustZone. In: International Conference on Communication, Computing and Digital systems (C-CODE), pp. 299–304 (2019)
10. Zhao, S., Zhang, Q., Hu, G., Qin, Y., Feng, D.: Providing root of trust for ARM TrustZone using on-chip SRAM. In: Proceedings of the 4th International Workshop on Trustworthy Embedded Devices (TrustED 2014), pp. 25–36 (2014)

11. Brasser, F., Gens, D., Jauernig, P., Sadeghi, A., Stapf, E.: SANCTUARY: ARMing TrustZone with user-space enclaves. In: Network and Distributed Systems Security Symposium (NDSS), p. 15 (2019)
12. Zhang, N., Sun, K., Shands, D., Lou, W., Hou, Y.T.: TruSpy: cache side-channel information leakage from the secure world on ARM devices. Cryptol. ePrint Arch. **2016**, 980 (2016)
13. Zhou, Y., Feng, D.: Side-channel attacks: ten years after its publication and the impacts on cryptographic module security testing. Cryptol. ePrint Arch. **2005**, 388 (2005)
14. Bryant, R.E., O'Hallaron, D.R.: Computer Systems: A Programmer's Perspective. 3rd edn. 650 (2016)
15. Lyu, Y., Mishra, P.: A survey of side-channel attacks on caches and countermeasures. J. Hardware Syst. Secur. **2**, 33–50 (2018)
16. Zhang, L., Tang, B.: A cost-effective cloud storage caching strategy utilizing local desktop-based storage. In: International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage (SpaCCS) Workshops, pp. 382–390 (2016)
17. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: the case of AES. Cryptol. ePrint Arch. **2005**, 271 (2005)
18. Yarom, Y., Falkner, K.: FLUSH+RELOAD: a high resolution, low noise, L3 cache side-channel attack. In: USENIX Security Symposium, pp. 719–732 (2014)
19. Zhang, N., Sun, K., Shands, D., Lou, W., Hou, Y.T.: TruSense: information leakage from TrustZone. In: IEEE Conference on Computer Communications (INFOCOM), pp. 1097–1105 (2018)
20. Apecechea, G.I., Inci, M.S., Eisenbarth, T., Sunar, B.: Wait a minute! A fast, cross-VM attack on AES. In: International Symposium on Research in Attacks, Intrusions and Defenses (RAID), pp. 299–319 (2014)
21. Lipp, M., Gruss, D., Spreitzer, R., Maurice, C., Mangard, S.: ARMageddon: cache attacks on mobile devices. In: USENIX Security Symposium, pp. 549–564 (2016)
22. Lapid, B., Wool, A.: Cache-attacks on the ARM TrustZone implementations of AES-256 and AES-256-GCM via GPU-based analysis. Cryptol. ePrint Arch. **2018**, 621 (2018)