Countermeasures against a Side-Channel Attack in a Kernel Memory

N.Y. Ahn [1*], D.H. Lee [2**]

[1] Graduate School for Information Security, Korea University, 145, Anam-ro, Seongbuk-gu, Seoul, Korea
[2] Professor at CIST and Graduate School for Information Security, Korea University
[*]humble@korea.ac.kr [**]donghlee@korea.ac.kr

Abstract: We proposed a zero-contention in cache lines a cache policy between REE and TEE to prevent from TruSpy attacks in a kernel memory of an embedded system. We suggested that delay time of data path of REE is equal or similar to that of data path of TEE to prevent timing side-channel attacks.

1. Introduction

TrustZone is hardware-based security built into system-on-chips (SoCs) by semiconductor chip designers. Main theme of TrustZone is the concept of secure and non-secure worlds that are hardware separated, with non-secure software blocked from accessing secure resources directly [1], [2]. Within the processor, software either resides in the secure world (or trusted execution environment TEE) or the non-secure world (rich execution environment REE); a switch between these two worlds is accomplished via software referred to as the secure monitor or by the core logic. This concept of TEE and REE extends beyond the processor to encompass memory, software, bus transactions, interrupts and peripherals within SoC [3]. Generally, TrustZone's software architecture in a mobile device. REE's mobile OS accesses TEE via a TrustZone library and a hardware driver. In the TEE, trusted applications execute on top of a minimal runtime environment, called the trusted OS, which provides a TEE internal application program interface (API) that trusted applications can use for communication with REE applications to access cryptographic operations and secure storage functionality. The trusted OS can enforce access control on trusted applications that attempt to access the secure memory [4], [5], [6]. TrustZone architecture does not define how REE applications access TEE [4].

2. Side-channel Attack in Kernel Memory

Typically, TEE includes three types of hardware models: 1) separated hardware REE and TEE 2) integrated REE and TEE 3) REE and TEE having shared hardware [7]. TrustZone is configured to include the shared hardware. Access of the shared hardware is determined according to the encrypted bit NS-bit. Particularly, REE and TEE share a main memory, for example, Dynamic Random Access Memory (DRAM). That is, DRAM is the memory shared by REE and TEE. NS-bit indicates whether to access one of the nonsecure area of DRAM and the secure area of DRAM [8]. The secure area is a space to store data encrypted by crypto algorithms.

Recently, Ning Zhang, Kun Sun, Deborah Shands, Wenjing Lou, and Y. Thomas Hou addressed the side-channel attack against the kernel memory of TrustZone [9]. This side-channel attack is referred as

TruSpy, which is the first study of timing based cache side-channel information leakage of TrustZone. TruSpy attack exploits the cache contention between REE and TEE as a cache timing side channel to extract sensitive information from the secure world, referring to figure 1.

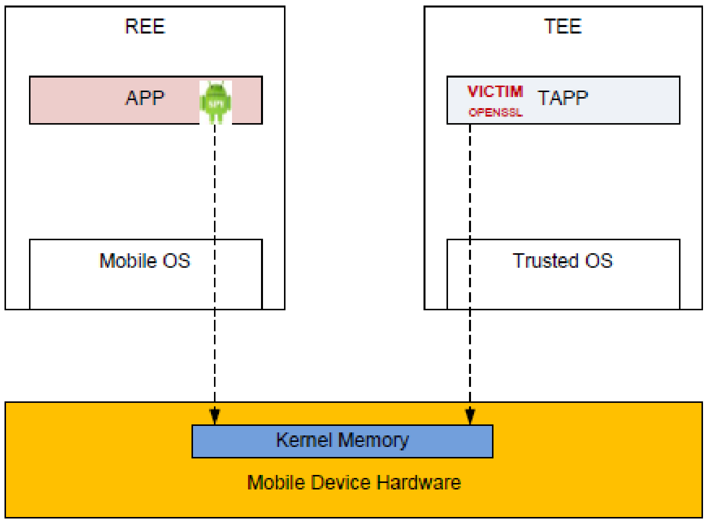

Figure 1 TruSpy Thread Model

There are two attack requirements for the TruSpy attack. Firstly, the attacking process can fill in cache lines at individual cache sets that will cause cache contention between REE and TEE. Secondly, the attacker can detect the state change in the cache lines [9].

The TruSpy attack scenario consists of five steps below [9], [10], referring to Figure 2:

The first step is to identify the cache memory to use for cache priming. The key is to find the cache memory that will be filled in cache line that is also used by the victim process in TEE. This step is often accomplished by working out the mapping from virtual address to cache lines [9].

The second step is to fill all cache lines. The spy process fills the cache with its own memory so that each cache line that can be used by the victim is filled with memory contents from the address space of the attacker [9].

The third step is to trigger the execution of the victim process in TEE. When the victim process is running, cache lines that were previously occupied by the attackers are evicted to the cache memory, such as DRAM. As a result, the cache configuration from the attacker's perspective has changed because of the execution of the victim process. Since this step is non-interruptible due to the protection of TrustZone, it is more challenging for this attack to succeed without fine grained information on the victim process cache access [9].

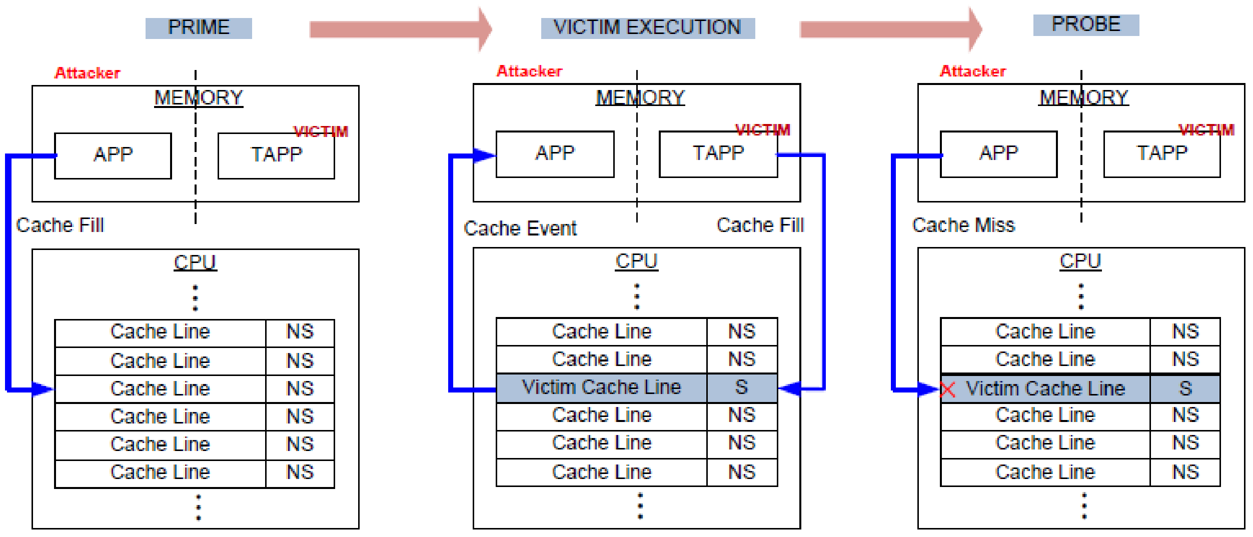

Figure 2. TruSpy Attack Scenario

The fourth step is to measure the state change in cache configuration after the victim finishes its execution in TEE. For each cache line that was previously primed in the second step, the time to execute memory load instruction is measured. If the time it takes to load the cache memory into register is short, then cache lines of which the cache memory is mapped to was not evicted by the victim process. Once the results are recorded for all the cache memory locations that were primed, the attack goes back to the second step and continues to collect more side-channel information [10].

The fifth step is to analyse the collected channel information to recover secret information such as cryptographic keys within the secure domain.

3. Proposed Countermeasures against TruSpy

In this paper, the proposed countermeasure against the TruSpy side-channel attack addresses largely two points. One point is to remove or mitigate cache contention about the kernel memory. Another point is to adjust timing of data paths for non-detecting cache transition.

Zero Contention in Cache Lines

In the above TruSpy modelling, this attack begins due to contention regarding shared cache lines between REE and TEE. Accordingly, if we remove the contention between REE and TEE, TruSpy side-channel attack can be blocked.

1) Hardware Zero Contention Scheme

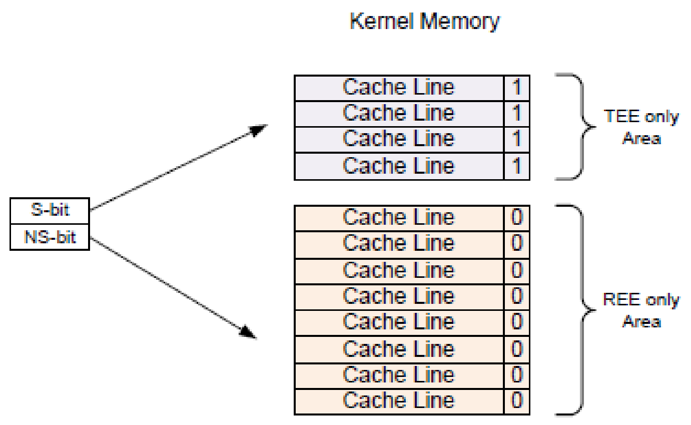

Figure 3. Hardware Separation of Caches

To achieve zero contention between REE and TEE, we suggest a separate cache memory hardwaredly. In an embodiment, the cache memory is divided from fixed REE-only cache lines and fixed TEE-only cache lines to remove or reduce the contention between REE and TEE with respect to the shared cache lines. For example, the fixed REE-only cache lines and the fixed TEE-only cache lines are hardware-implemented in the cache memory to achieve the zero contention, referring to figure 3.

2) Software Zero Contention Scheme

In another embodiment, an allocation of REE addresses and TEE addresses is fixed according to the cache policy. REE addresses and TEE address are separated from each other, referring to figure 4. REE addresses correspond first cache lines in the cache memory, and TEE addresses correspond second physical cache lines. The first cache lines and the second cache lines are not shared, but separated. In this case, the attacker cannot access and fill cache lines corresponding to the TEE logical addresses. But not to limited, various schemes can be introduced about configuration of the cache memory or assignment the logical address for zero contention.

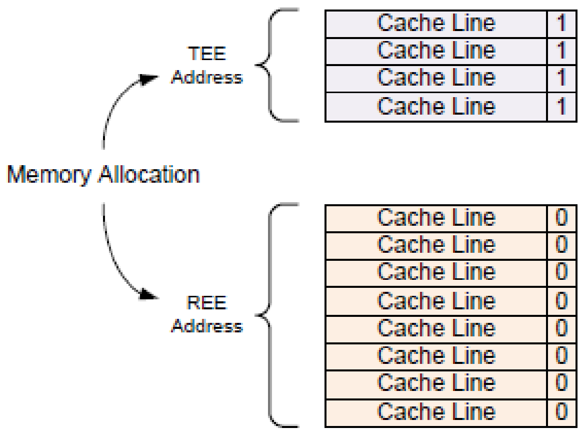

Figure 4. Cache Allocation Policy

Timing Attack Countermeasure

In TruSpy Attack, this timing attack can be performed by sensing state change of a target cache line. Especially, the attacker tries to detect the cache line change by measuring the transition time to load cache data from the cache memory to the register. Then, we must have no difference between TEE load time and REE load time for prevent this timing side-channel attack.

3) REE Data Path Delay Scheme

Generally, TrustZone technology determines REE resource access or TEE resource access according to NS-bit. REE has longer data path latency than that of TEE because TEE performs at least one operation in a cryptography function. Then the attacker performs a timing side-channel attack by using this delay time between REE and TEE. Therefore we suggest that data path latencies of REE and TEE should be set to equal or similar to each other.

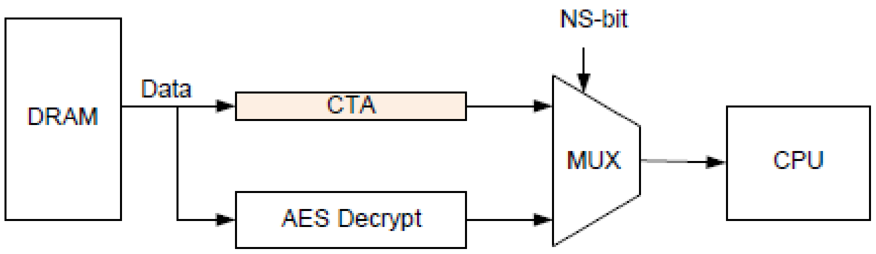

Figure 5. Read Path with CTA

In figure 5, data path of REE is configured to include a countermeasure circuit for timing attack CTA. For example, CTA may be a delay circuit (eg. flip-flop configuration) to have almost no difference in data path latency between TEE and REE.

We supposed that the kernel memory of TrustZone is DRAM. Firstly, in a read operation, read data from DRAM may be one of secure data and non-secure data. The secure data is transferred to TEE via TEE data path and non-secure data is transferred to REE via REE data path. The read data from DRAM includes NS-bit, which indicates TEE data path or REE data path. MUS selects whether one of REE data path and TEE data path is connected to CPU according to the NS-bit. Referring to Figure 5, the shown REE data path includes CTA. Accordingly data latency of REE data path is equal or similar to that of TEE data path until the read data arrive at CPU. For example, decryption time by AES decrypt is equal or similar to delay time by CTA.

Secondly, in a write operation, NS-bit indicates REE data path or TEE data path. Similarly, data latency of REE data path is equal or similar to that of TEE data path until the write data arrive at DRAM, referring to figure 6. For example, encryption time by AES encrypt is equal or similar to delay time by CTA.

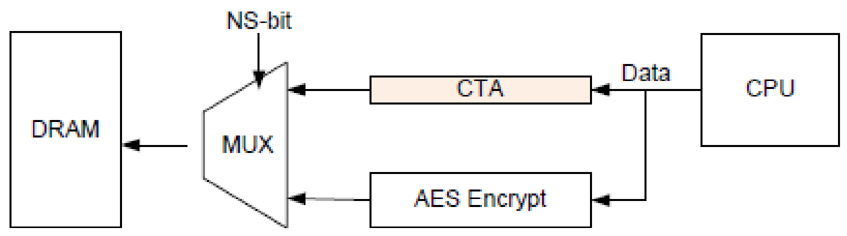

Figure 6. Write Path with CTA

4) TEE Parallel Data Path Scheme

Also, the intended data path delay in REE may have a bad effect. To solve this problem, we suggest parallel data paths in TEE. For example, due to implemented parallel crypto circuits, we can screen time difference between REE data path and TEE data path. In read operation, non-secure data from DRAM is transferred to CPU of REE via REE data path, referring to figure 7. Also, secure data from DRAM is transferred to CPU of TEE via TEE data path having parallel AES Decryptors. AES Decryptors perform parallel decryption operation on the read data from DRAM. MUX selects one of non-secure data of REE data path and secure data of TEE data path according to NS-bit. Data of TEE data path are output from AES Decryptors. Accordingly data latency of REE data path can be same or similar to that of TEE data path until the read data arrives at CPU.

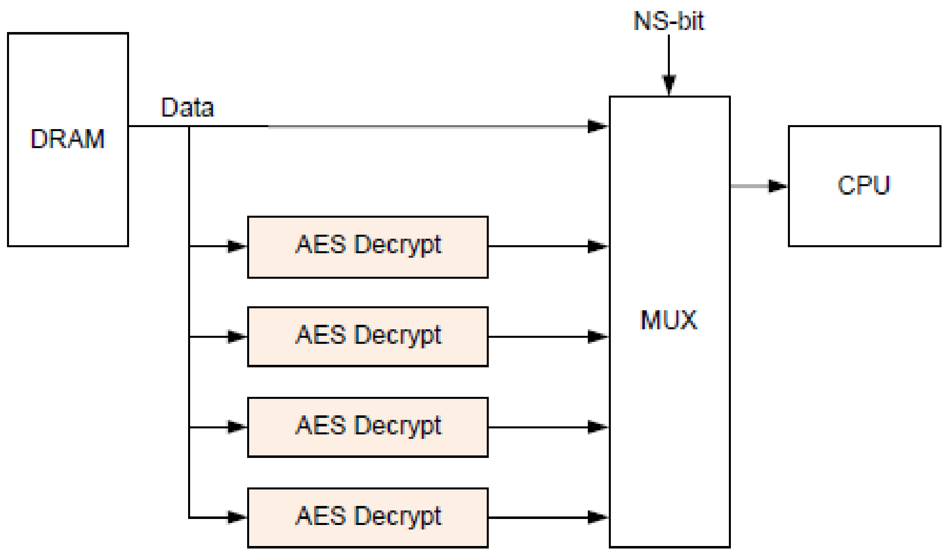

Figure 7. Parallel Decryption in Memory Read

In write operation, non-secure data from CPU is transferred to DRAM of REE via REE data path, referring to figure 8. Also secure data from CPU is transferred to TEE via TEE data path h having parallel AES Encrypts. AES Encryptors perform parallel encryption operation on the write data from CPU. MUX selects one of non-secure data of REE data path and secure data of TEE data path according to NS-bit. Write data of TEE data path are outputted from AES encrypts. Accordingly data latency of REE data path can be equal or similar to that of TEE data path until the write data arrive at CPU.

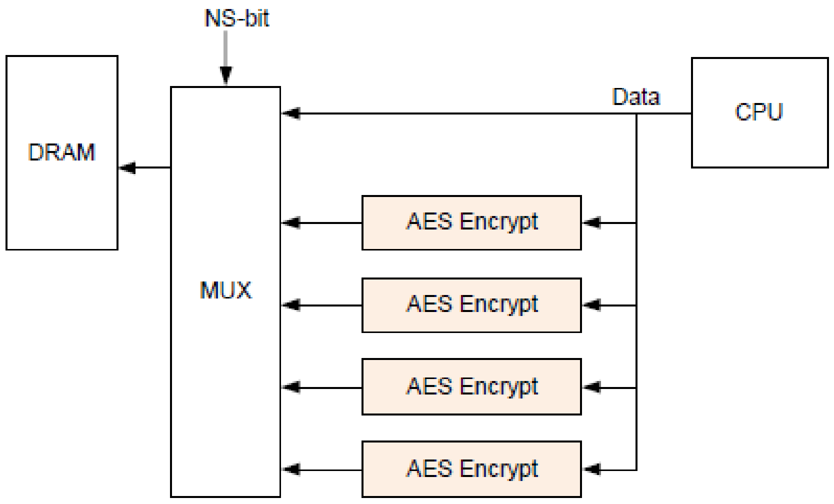

Figure 8. Parallel Decryption in Memory Write

TEE Integrity Enhancement using Clark-Wilson Model

Recently, Kaiqiang Li, Hao Feng, Yahui Li, and Zhiwei Zhang established the model of information flow control, and designed information flow control policies of MMR and Guard, which make all the communication data pass security audit and retransmission controlled by information flow control mechanism and credible components to meet the multilevel security requirements. The model and method effectively guarantee the confidentiality of the information flow in MILS using Bell-LaPadula (BLP) model [11].

The Clark-Wilson model was described by David D. Clark and David R. Wilson [12]. This Clark-Wilson model introduces a way to formalize the notion of information integrity, especially as compared to the requirements for multi-level security (MLS) systems. The Clark-Wilson security model is based on reserving information integrity against the malicious attempt of tampering data. The security model maintains that only authorized users should make and be allowed to change the data, unauthorized users should not be able to make any changes, and the system should maintain internal and external data consistency [13]. We assumed that the mobile device having TrustZone is a kind of a multi-level security (MLS) system.

We proposed security information flow control model to enhance the integrity of the information content between REE and TEE based on the Clark-Wilson model, and built the information flow control mechanism using TrustZone driver and Authentication Tokenization Program (ATP). Generally an integrity level of a component (ex. APP) in REE is lower than that of a component (ex. TA) in TEE. Naive APP on REE cannot access a secure memory.

The proposed ATP may generate a token by communication with the TA in TEE, and transfer the token to APP. Then APP can access the secure memory using the token. For example, if APP wants to write secure data in the secure memory based on a request, APP transforms data to be written using the token in the secure memory. For example, the transformed data may be obtained by performing a first XOR operation on data and the token. Then APP transfers the transformed data and the token to ATP.

The transformed data correspond to UDI (unconstrained data item) of Clark-Wilson Model. ATP verifies the transformed data using the token. If the verification is passed, ATP performs a second XOR operation on the transformed data and the token to obtain the data to be written in the secure memory. The data correspond to CDI (constrained data item) of Clark-Wilson Model. Then the data from ATP is transferred to the secure memory via TZ driver of Mobile OS and Trusted OS.

Also, the token is generated by TA of TEE and is stored in the secure memory via TA. ATP compares the token transferred from APP with the token stored in secure memory in the above verification operation. Also, the token may include time expiration information. ATP may perform the verification operation as to whether there is the token or whether the token is valid based on the time expiration information.

Also, ATP verifies whether the token received from APP on REE is available by comparing the token from REE with the stored token in TEE. Also, ATP can manage the token via the trusted application (TA) on TEE.

4. Conclusion

In this paper we introduced countermeasures against side-channel attacks in the kernel memory of TrustZone. We proposed a zero-contention cache memory or a cache policy between REE and TEE to prevent TruSpy attacks in TrustZone. And we suggested that delay time of data path of REE is equal or similar to that of data path of TEE to prevent timing side-channel attacks.

References

[1] https://www.arm.com/products/security-on-arm/trustzone

[2] A. Fitzek, F. Achleitner,J. Winter and D. Hein, "The ANDIX research OS — ARM TrustZone meets industrial control systems security" 2015

[3] IEEE 13th International Conference on Industrial Informatics, 2015.

[4] Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan, “The Untapped Potential of Trusted Execution Environments on Mobile Devices”, IEEE Security & Privacy, 2014

[5] Mohamed Sabt, Mohammed Achmlal, and Abdelmadjid Bauabdallah, “Trusted Execution Environment: What It Is, and What It Is Not,” IEEE Trustcom/BigDataSE/ISPA, 2015

[6] S.Pinto, D. Oliverira, J. Perira, N. Cardoso, M. Ekpanyapong, J. Carbral and A. Tavares, “Towards a Lightweight Embedded Virtuallization Architecture Exploiting ARM TrustZone”, IEEE Emerging Technology and Factory Automation (ETFA), 2014

[7] Mobile Platform Security Trusted Execution Environments Summer School”, 2014 N. Asokan Aalto University and University of Helsinki
David Kaplan, Jeremy Powell, Tom Woller, “AMD MEMORY ENCRYPTION”April 21, 2016,

[8] Ning Zhang and Kun Sun and Deborah Shands and Wenjing Lou and Y. Thomas Hou, “TruSpy: Cache Side-Channel Information Leakage from the Secure World on ARM Devices”, http://eprint.iacr.org/2016/

[9] D. A. Osvik, A. Shamir and E. Tromer, "Cache Attacks and Countermeasures: the Case of AES", in Topic in Cryptology-CT-RSA, 2006

[10] Kaiqiang Li, Hao Feng, Yahui Li, and Ziwei Zhang, “An improved method of access control based on BLP model In MILS,” IEEE, 2014 Tenth International Conference on Computational Intelligence and Security, 2014.
[11] Clark, David D. and Wilson, David R. “A Comparison of Commercial and Military Computer Security Policies”, in Proceedings of the 1987 IEEE Symposium on Research in Security and Privacy (SP'87), May 1987, Oakland, CA; IEEE Press, pp. 184–193

[12] https://en.wikipedia.org/wiki/Clark-Wilson_model

[13] http://www.drchaos.com/clark-wilson-security-model