신뢰 실행 환경을 활용한 애플리케이션 프로세서의 실시간 통신 장치 접근 보안

카욘도마틴 [1], 최진명 [2], 백윤흥 [3]
[1,2] 서울대학교 전기정보공학부 석박사과정
[3] 서울대학교 전기정보공학부 교수

kayondo@snu.ac.kr, jmchoi@sor.snu..ac.kr, ypaek@snu.ac.kr

Secure Realtime Communication Peripheral Access on Application Processors with TrustZone

Martin Kayondo[1], Jin-Myung Choi[2], Yun-Heung Paek[1,2]
[1]Dept. of Electrical and Computer Engineering, Seoul National University
[2]Inter-University Semiconductor Research Center (ISRC), Seoul National University

Abstract

Application processors (APs) are increasingly used in mixed-criticality systems (MCS), where components with varying safety and real-time requirements share a platform. In MCS like automotive systems, real-time communication is essential for safety. While APs often handle non-critical tasks, they still rely on real-time data and peripherals. However, their multi-processing nature and often external network connectivity expose them to attacks, which can compromise real-time channels or make the APs a source of malicious data. This paper explores the possibility and challenges of securing real-time communication peripheral access on application processors in MCS environments.

1. Introduction

Application processors (APs) are increasingly integrated into distributed mixed-critical systems (MCS), where components with carrying safety, real-time, and performance requirements coexists on hardware platform connected by networks. In MCS, particularly in domains such as automotive, real-time communication is essential to uphold safety guarantees. Although APs are not typically responsible for hard real-time tasks, they often process workloads that depend on real-time data, necessitating access to real-time peripherals. However, given their multi-processing nature, concurrent execution of diverse applications, and frequent exposure to external networks, APs present a larger attack surface. This exposure raises the risk of compromised or manipulated data entering the system through real-time channels, or the processor itself becoming a source of malicious activity. As such, securing access to real-time communication peripherals as the AP end is critical.

A typical approach to securing real-time communication is the use of cryptography. However, strong algorithms like asymmetric cryptography are often impractical in MCS, as many nodes lack sufficient processing power. Lighter alternatives, such as symmetric cryptography, may not meet the required security standards for complex systems. To address this, hardware security modules (HSMs) or minimal cryptographic engines (HSE, SHE) have been proposed for resource-constrained nodes. While application processors can handle strong cryptographic algorithms, they face challenges in secure key storage, access, and management. One solution is to integrate HSM-like security engines on APs, but this increases implementation costs and underutilizes the APs' capabilities. Fortunately, modern APs now support Trusted Execution Environments (TEEs), enabling trusted execution without the need for additional hardware.

In this paper, we explore the use of Trusted Execution Environments (TEEs) on application processors (APs) to secure real-time communication in distributed mixed-criticality systems (MCS). Assuming cryptographic mechanisms are employed system-wide, with resource-constrained nodes supported by dedicated cryptographic engines, we propose offloading cryptographic operations for real-time communication to a trusted environment on the AP. We outline key requirements and challenges for establishing a secure real-time communication model on APs within MCS. Our solution is based on ARM TrustZone as the TEE, implemented on a Cortex-A53 processor. While the model focuses on the CAN protocol, it is extensible to other interfaces such as SPI for short-range communication and

LIN for lower-criticality tasks.

2. Background

2.1 Mixed-Criticality Systems

Mixed-Criticality Systems (MCS) are computing platforms that execute tasks or applications with different levels of criticality on shared hardware or interconnected network nodes, as in cyber-physical systems (CPS). They integrate components with varying safety, real-time, and performance requirements. In this paper, we focus on automotive systems, which exemplify MCS. These systems typically include brake control systems that are highly safety-critical and require strict real-time operation, infotainment systems that are non-critical and computation-heavy but not time-sensitive, and advanced driver assistance systems (ADAS) that rely on sensor data and algorithms for safe operation and have moderately flexible timing demands. Automotive MCS are powered by heterogeneous computing, combining real-time processors for deterministic control tasks like braking and engine management, application processors such as ARM Cortex-A for high-level non-real-time tasks, GPUs for graphics and ADAS workloads, and a variety of sensors for real-time data collection. These components communicate over different in-vehicle networks, including real-time protocols like CAN for low-latency communication and high-throughput Ethernet for large data transfers. While real-time protocols like CAN are designed for timeliness, they often lack robust security features. Application processors, which are frequently connected to both real-time networks and external interfaces such as wireless or internet connections, pose a significant attack surface. If compromised, an AP can provide a path for attackers to access critical vehicle systems, potentially jeopardizing the entire platform's safety. Therefore, securing AP communication on real-time networks is essential to preserving both the safety and security of automotive MCS.

2.2 Trusted Execution Environments (TEEs):

Trusted Execution Environments (TEEs) enable the execution of programs or code within a secure, isolated context known as an enclave. Depending on the architecture, the enclave may have exclusive access to a dedicated memory region, which can be restricted either to the code running within it or, in some designs, to a group of trusted applications. TEEs also support binding peripheral devices to the secure world, ensuring that only trusted code can interact with them. On ARM systems, the most widely used TEE implementation is TrustZone. TrustZone divides the system into two worlds: the secure world and the normal world. The secure world has access to isolated memory that the normal world cannot reach, enforced by the TrustZone Address Space Controller (TZASC). Peripheral access can also be restricted to the secure world using the TrustZone Peripheral Controller (TZPC). The secure world and normal world communication via shared memory and secure monitor calls (SMC). Secure monitor calls allow the normal world to hand-over execution to the secure world, a process that results in saving the execution context to which to resume when the call ends.

2.3 Realtime Communication on MCS:

MCS, especially distributed ones such as automotive, which may contain multiple computing nodes are usually connected over networks. For automotives, the in-vehicle network (IVN) connects several computing nodes called electronic control units (ECUs). Among ECUs are microcontrollers for real-time computing connected over networks such as CAN for fast real-time data transmission. A few general-purpose computing nodes such as the infotainment node may also be connected to this network to received and report real-time data of the vehicle or may send messages to ECUs connected to the IVN to control the vehicle. Unfortunately, such general-purpose computing nodes are usually also connected to external networks, making them easy targets for remote attacks, and as such remote attackers can leverage them to inject real-time messages into the IVN and control the vehicle.

3. Securing Realtime Communication on APs with TrustZone

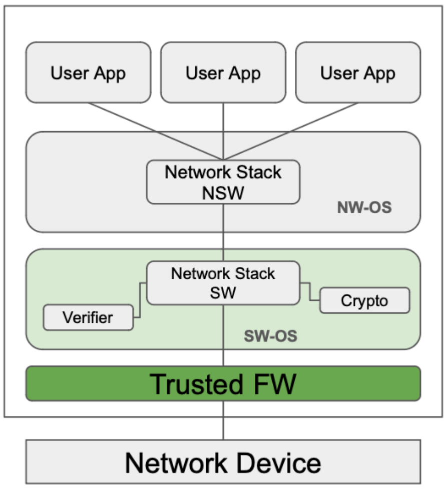

Figure 1: Securing peripheral access using TrustZone-A

As earlier explained, securing real-time communications especially from APs possibly connected to external networks is crucial to maintaining safety and security of the whole system. In this paper, we assume APs based on ARM Cortex-A processors with TrustZone-A.

3.1 Requirements:

By secure communication, we envision a developer wishing to monitor all real-time communication on the APs. This means monitoring all messages from the AP to the IVN and messages from the IVN to the AP. As such, the system requires the following:

- R1: A defined classification of legal and illegal communication. This may be the transmission rate, particular message types (specific CAN IDs, SPI channels, etc.) or a more contextual thoroughly defined firewall combining both.
- R2: A communication verifier and authentication mechanism. This mechanism verifies and authenticates in- and out-bound transmissions, disallowing any illegal ones. This can be a piece of software coupled with a file listing allowed in- and out-bound CAN IDs, and a method for verifying the authenticity of such a list, and facilitation of its persistent storage.
- R3: Communication peripheral access control. Communication peripherals are responsible for transmitting data between the AP and the network bus. For example, a CAN module connects an AP to the CAN bus network. Any software that can access the module has permission to transmit CAN frames and read received CAN frames on the module. For secure communication, access to such a peripheral must be limited only to trusted software, in this case, the verifier and possibly the authentication mechanism.
- R4: Minimal TCB: The trusted computing base, such as the verifier and authentication mechanism must be minimal to avoid bloating the secure world, which may increase the attack surface.
- R5: A communication medium linking the normal world to the peripheral. Since access to the peripheral is controlled, a medium such as shared memory between secure world and normal world may be required to pass on transmission messages from normal world to secure world and received messages from secure world to normal world.

3.2 Secure Communication:

Assuming the developer has a list of legal CAN IDs allowed for transmission, we propose saving such a list in a file on an available secure non-volatile memory (secure NVM). [6] suggests using fTPM with OPTEE's secure non-volatile storage mechanism to facilitate secure NVM using TrustZone-A. Other works such as [3,4] suggest secure storage using a TPM, but that requires additional hardware. In this paper, we anticipate the cost of APs should further be justified by handling such requirement as secure NVM using the available NVM - such as using the fTPM which is open-sourced.

Next, the CAN module can be assigned to the secure world using TZPC, ensuring its access is only from secure world. If the CAN module also has memory-mapped registers, they are mapped to the secure world using TZASC, further enhancing the access control to the module. Any interrupts raised by the peripheral must be handled in the secure world (which is obvious since the peripheral is assigned to the secure world).

3.2.1 Transmission:

During transmission, peripheral drivers from the normal world originally access the peripheral directly, for example to write to transmission buffers of the peripheral device. In this case, they write to the communication medium (R4), and then make an SMC to signal the secure world about the availability of data to transmit.

Handling the SMC call at the secure world end requires invoking the verifier to check the message to be transmitted and either authorized its transmission by writing it to the transmission buffer and making any necessary register settings or denying it if deemed illegal and simply dropping it. The secure world may return a result to the normal world on whether the message was transmitted or dropped (this is a developer's choice). For most peripherals, a successful transmission will raise an interrupt, which allows the secure world to inform the normal world, allowing it to send more messages if available.

3.2.2 Reception:

When a message arrives at the peripheral, most peripherals will raise a reception interrupt. Since the peripheral is configured to the secure world, execution will transition to the reception handler in the secure world, where the verifier or authenticator can be invoked to secure the AP from malicious messages if need be. The message can then be written to the shared memory. At this point, there is no way the normal world is aware of this new message. We propose using a software generated interrupt (SGI) raised by secure world targeting the normal world.

Figure 1. summarizes the basic framework described above.

4. Challenges:

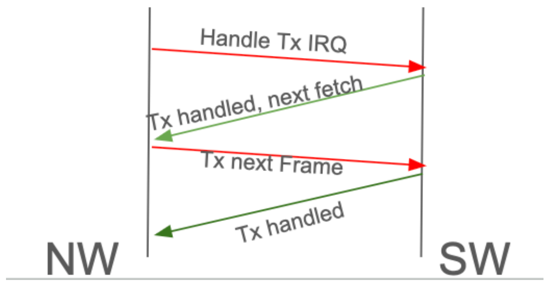

Figure 2: SMC calls required to handle transmission

C1. Performance overhead:

The biggest challenge is balancing between performance requirements of real-time communications and the runtime performance cost of using TrustZone-A. The most obvious source of performance overhead is the number of world switches needed both to transmit or receive a single message. Under TrustZone-A, an SMC call requires saving contexts and sometimes flushing caches for security reasons. A single SMC call requires hundreds and sometimes thousands of CPU cycles. For real-time communication, the AP may miss newly available messages. For some peripherals, such as in CAN, a message may be dropped if another one arrives while the reception buffer is still uncleared. This situation is referred to as a buffer overrun. When multiple buffer overruns occur, the AP may miss critical information, say from sensors, or fail to handle critical situations. For example, in our experiments, when a frame burst occurs, about 7% of the CAN messages are lost due to this performance slowdown. For transmission, since transmitting a single message is costly, the overall throughput is greatly affected. Our experiments show that attempting to transmit multiple messages at tight intervals results in a 33% throughput degradation. Figure 2. shows the world switches that may be required to handle a simple transmission interrupt and before sending another message.

Therefore, the developer must devise a smarter way to improve both transmission throughput and reception speed to avoid or minimize frame loss. [5] suggests reserving a whole CPU core and running a real-time OS (RTOS) on it to handle communication requests from other AP CPU cores. Using TZPC, the communication peripherals are assigned to the RTOS CPU core, such that the other CPUs cannot access it. The challenge with this design is that it may degrade the overall AP performance, especially if real-time communication is only required at particular periods, not all the time during execution. A workaround may involve scheduling the RTOS dynamically on one of the CPU cores as an optimization for when repeated transmission is needed or a reception burst occurs. In this design, however, the developer is further tasked with designing and defining the conditions under which the RTOS can be scheduled or paused. Another alternative is using batched transmission as done by [1] and [2]. However, it is important to note that not all peripherals support batched I/O.

C2: Message Verification:

In automotive systems, messages can be authenticated using message authentication codes (MAC). However, most existing research proposes authenticating messages at the computing node level. This means that a message can only be authenticated as originating or intended to be received by a particular AP. A more realistic solution must enforce finer-grained authentication, verifying and authenticating transmission messages based on the application intending to send the message. This way, it is easy to know block malicious applications on the AP from sending unauthorized messages. More challenging, however, is the fact that most real-time communication protocols such as CAN or SPI do not include sender application information in the message, requiring the developer to devise a means to identify the normal world sender application at the verifier level at the secure world end.

5. Conclusion

In this paper, we present a basic framework for securing real-time communication on application processors in distributed mixed criticality systems, taking automotive systems as an example, using TrustZone. For our reader, we also outline challenges that may be encountered in realizing this framework in a real-world system. Our framework relies on TrustZone which is readily available on commonly used ARM Cortex-A processors and does not require additional hardware such as TPMs. For future works, we plan on devising solutions to the outlined challenges to reduce the transmission throughput overhead and reception frame loss.

6. Acknowledgements

This research was supported by Korea Planning & Evaluation Institute of Industrial Technology (KEIT) grant funded by the Korean Government (MOTIE) (No. RS-2024-00406121, Development of an Automotive Security Vulnerability-based Threat Analysis System(R&D)). This work as also supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) under the artificial intelligence semiconductor support program to nurture the best talents (IITP-2023-RS-2023-00256081) grant funded by the Korea government (MSIT). Additionally, this work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00277326). Furthermore, this work was supported by the BK21 FOUR program of the Education and Research Program for Future ICT Pioneers, Seoul National University in 2025. Finally, this work was supported by Inter-University Semiconductor Research Center (ISRC).

참고문헌

[1] Röckl, Jonas, Nils Bernsdorf, and Tilo Müller. "TeeFilter: High-Assurance Network Filtering Engine for High-End IoT and Edge Devices based on TEEs." Proceedings of the 19th ACM Asia Conference on Computer and Communications Security. 2024.

[2] Schwarz, Fabian. "TrustedGateway: TEE-assisted routing and firewall enforcement using ARM TrustZone." Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses. 2022.

[3] Plappert, Christian, and Andreas Fuchs. "Secure and Lightweight Over-the-Air Software Update Distribution for Connected Vehicles." Proceedings of the 39th Annual Computer Security Applications Conference. 2023.

[4] Plappert, Christian, and Andreas Fuchs. "Secure and lightweight ecu attestations for resilient over-the-air updates in connected vehicles." Proceedings of the 39th Annual Computer Security Applications Conference. 2023.

[5] Kim, Se Won, et al. "Secure device access for automotive software." 2013 International Conference on Connected Vehicles and Expo (ICCVE). IEEE, 2013.

[6] Raj, Himanshu, et al. "{fTPM}: A {Software-Only} Implementation of a {TPM} Chip." 25th USENIX Security Symposium (USENIX Security 16). 2016.