Secure Device Access for Automotive Software

Se Won Kim*, Chiyoung Lee*, MooWoong Jeon*, Hae Young Kwon**, Hyun Woo Lee**, and Chuck Yoo*

*Korea University, Seoul, Republic of Korea
**Hyundai Motor Company, Seoul, Republic of Korea
E-mail:{swkim, cylee, mwjeon, hxy}@os.korea.ac.kr, {hykwon, lhwpro}@hyundai.com

Abstract—Current automotive software is evolving to integrate extension software with control software. Software integration increases the complexity of software and can cause system fault that threatens safety of automobile. To solve this problem, extension software should be isolated from control software. However, this isolation alone is not sufficient to prevent extension software from faulting control software due to the lack of secure device access. In this paper, we propose a secure automotive software platform that has secure device access method with TrustZone. Our method restricts a direct access of the extension software and supports multicore processors. Measured overhead of our platform shows less than 1% degradation, and the maximum bandwidth of device access is achieved up to 5MB/s.

Keywords—ARM TrustZone, Automotive Software, Virtualization, Reliable System, Embedded system

I. INTRODUCTION

Automobile software consists of control software that controls ECU (Electronic Control Unit)s and extension software that handles infotainment devices such as AVN (Audio, Video and Navigation) system. Current automobile market trends have shifted to integrate more extension software with control software. However, according to several reports, extension software can cause safety problems because they are not fully verified. Most of the extension software is complex because it has to perform various functions for user convenience such as head-up display and touch interface. The complexity makes it difficult to verify the correctness of extension software. In addition, the extension software’s misbehavior can affect control software, which causes safety problem [1][2]. Therefore, isolating the extension software from control software is required.

For the isolation of the extension software, TrustZone has been used [3]. Originally, TrustZone was designed for the security of mobile platforms. TrustZone provides two virtual execution environments (worlds), called secure world and normal world. These two worlds have CPU states and registers which are independent on the other world. Thus, TrustZone can isolate the software running in each world. Several studies have presented software isolation using TrustZone [4][5][6][7].

Typically, control software runs on RTOS (Real-Time OS) to meet real-time requirements, and extension software runs on GPOS (General Purpose OS) to provide infotainment services. With TrustZone, control software and extension software are isolated by placing RTOS in secure world and GPOS in normal world. However, although control software and extension software are isolated for safety, they also need to share devices such as CAN (control area network) and sensors. For example, digital cluster periodically collects engine RPM and speed information from CAN and renders 3D modeled digital gauges with collected data.

If the digital cluster constantly sends massive requests into CAN (due to the misbehavior of extension software) as shown in Figure 1, other ECUs could not collect proper information to control the automobile and consequently the automobile can be out of control. So the isolation alone (using TrustZone) is not sufficient to prevent extension software from affecting control software when devices are shared between control software and extension software. Specifically, when an extension software in GPOS directly access a shared device, a fault of the extension software can propagate to control software.

Therefore, a secure device access mechanism is necessary in addition to the isolation of the extension software.

This paper presents SASP (Secure Automotive Software Platform) that aims to support the secure device access in TrustZone. SASP provides a device access mechanism to securely isolate control software from the extension software in GPOS. This mechanism restricts a direct access of the extension software in order to ensure the secure device access. In addition, SASP extends the secure device access mechanism to support multicore processor environments.

This paper is organized as follows. Section 2 discusses the related work, and Section 3 briefly introduces TrustZone in the perspective of this paper. In Section 4, we describe the design of SASP and how TrustZone can be utilized for SASP. Section 5 presents experimental results that show the performance overhead of SASP, device access latencies and bandwidth. Finally, a conclusion is drawn in Section 6.
II. RELATED WORK

There have been several researches using TrustZone for the software isolation. [4] describes a secure mobile software platform that isolates SE-Linux from Linux by running the SE-Linux in secure world. In [5], authors isolate a secure kernel in secure world from Linux in normal world. [6] performs DRM operations in secure world to provide secure DRM License access. These researches, however, do not consider the shared device between two worlds.

INTIGRITY Multivisor [12], which also uses TrustZone for virtualization, provides device sharing mechanism. It runs a hypervisor in secure world and more than two guest OS in normal world in order to isolate the hypervisor from the guest OS. However, INTIGRITY Multivisor does not consider secure access for shared device.

SafeG isolates RTOS from GPOS by TrustZone [7][8][9]. In [7] and [8], the authors implement a monitor to switch between worlds and an interrupt mechanism to satisfy the real-time requirements. These RTOS supporting features are also discussed in previous papers [10][11].

In [9], to access a shared device, SafeG proposes the re-partitioning mechanism that dedicates the device to a particular OS whenever it is needed. If RTOS sends UNPLUG event to GPOS, GPOS could not use the device until receiving the PLUG event. It allows RTOS access a device whenever RTOS wants to use the device, regardless of GPOS.

However, the device access mechanism of SafeG does not address how to protect the shared device from the extension software’s misbehavior. It cannot prevent failure of the device caused by the GPOS accesses because the mechanism only reclaims the access permission of the device from the GPOS when the RTOS requires the device.

III. BACKGROUND

A. TrustZone Architecture

TrustZone provides two independent software execution environments on ARM processor: secure world and normal world. Secure world is able to access to all system resources, whereas normal world cannot access the memory and devices of secure world. To support the asymmetry access of two worlds, the TZPC (TrustZone Protection Controller) partitions hardware resources including memory and peripherals [13].

TrustZone needs a monitor software to switch between two worlds. When the world is switched, the hardware context, including cache, TLB and MMU, is automatically changed. Thus, the two worlds have completely isolated views of hardware. One exception is general-purpose registers (r0–r12), and those registers are shared between the worlds. To switch between worlds, TrustZone provides an instruction called SMC (Secure Monitor Call), and an execution mode, monitor mode. When SMC instruction is executed, hardware context of the current world is automatically flushed so that it cannot be read by the other world. Then, TrustZone enters the monitor mode, and the monitor software switches the general-purpose registers between two worlds.

TrustZone utilizes prioritized interrupt model. This interrupt models support two kinds of interrupts: FIQ and IRQ. FIQ is used to deliver device interrupt of secure world, and IRQ is used for normal world. Because priority of FIQ is higher than IRQ, RTOS in secure world can process interrupts without interference of other interrupts of normal world so that FIQ is used to support real-time.

B. Inter-processor interrupt for multicore processor

Previous researches are based on single core processor and propose device access method using SMC. However, SMC does not address the following situation. In multicore processor, RTOS and GPOS can run on separate cores. If RTOS runs on core 0 and GPOS runs on core 1, SMC cannot deliver messages between RTOS and GPOS because SMC is designed to switch worlds on the same core.

To overcome the limitation of SMC, we use IPI (Inter Processor Interrupt). IPI is a communication method among cores in multicore processors. However there is a restriction of IPI in TrustZone. Secure world can send IPI to any world in any core, but normal world can send IPI to only normal world, not secure world.

IV. DESIGN AND IMPLEMENTATION OF SASP

In this section, we describe the architecture of SASP and how to implement secure device access mechanism. Then, we extend the proposed secure device access mechanism to support multicore processor.

Figure 2. SASP Architecture

A. Overall architecture

Figure 2 shows the architecture of SASP that has four components: para-TrustZone GPOS (GPOS in Fig. 3 and 4), para-TrustZone RTOS (RTOS in Fig. 3 and 4), V-Monitor, and V-Bridge. Para-TrustZone GPOS is a modified GPOS to use devices allocated in secure world. It includes device stub to initiate access to the device. Para-TrustZone GPOS also handles IRQ, not FIQ, and provides memory mapping for the memory allocated to normal world.
Para-TrustZone RTOS is a modified RTOS to run on secure world of TrustZone. The modification is needed to support devices allocated to secure world. For example, CAN device is allocated to secure world, and we modify RTOS for the CAN driver on secure world. Also, the modification includes changes for interrupt handling: when a device generates interrupt, TrustZone translates the interrupt into FIQ, and para-TrustZone RTOS handles FIQ, including ones from CAN.

Before describing V-Monitor and V-Bridge, we first describe our security model. We divide security model into admission control and behavior control, and they are performed by V-Monitor and V-Bridge respectively. The admission control means that V-Monitor checks whether GPOS has access right to the device. Admission control can block illegal access requests. In addition, V-Monitor has functions that manage RTOS and GPOS (Guest OS Manager in Figure 2), distributing interrupts (Interrupt Router), managing shared memory (Buffer Manager) and transferring IPI of GPOS to RTOS (IPI Stub).

The behavior control means that V-Bridge evaluates the legitimacy of request. Even though GPOS has a proper access right, it may have only read right, not write right. In this case, V-Bridge rejects write request of GPOS. V-Bridge receives device access request from device stub in GPOS, and forwards device access request to destination device driver via SMC instruction.

B. Secure device access

Our secure device access scheme is based on shared memory managed by V-Monitor. Figure 3 shows a single core device access scenario. When the GPOS wants to use the device in secure world, (1) device stub prepares device access request, and (2) V-Bridge copies the request to the shared memory. Then, (3) V-Bridge in GPOS notifies V-Monitor for device access using SMC. When V-Monitor is notified, (4) V-Monitor switches the world from normal world to secure world. Then, (5) V-Monitor copies the request from the shared memory to V-Bridge of RTOS in Fig. 3. Finally, (6) the V-Bridge invokes device driver to handle the request.

C. Multicore support

Because SMC can be used only in single core processor, we need a different notification mechanism when SASP runs on multicore. To support safe device access in multicore processor, IPI is used to implement device access as shown in Fig. 4. However, as mentioned in Section 3, normal world cannot send IPI to secure world directly because TrustZone restricts normal world from sending IPI to secure world. To resolve this problem, we additionally introduce IPI stub to the V-Monitor.

Figure 4 shows how the normal world OS sends IPI to the secure world OS. Firstly, the sender, GPOS (V-Bridge) in the Figure, copies requests from its own memory to the shared memory. Secondly, the GPOS executes SMC instruction, and the SMC handler in the V-Monitor catches the SMC call. Thirdly, the V-Monitor invokes the IPI stub. Because the IPI stub runs in secure world, now it can send IPI to another core in secure world. Finally, RTOS (V-Bridge) in the Figure, receives IPI and copies requests from the shared memory to the local memory, and the request is processed. Since the shared memory can be read and written by both worlds at any core, V-Monitor maintains a queue per core for synchronization.

V. EVALUATION

For evaluation, we implement SASP on the NVIDIA Tegra3 application processor based on the ARM Cortex-A9 MPCore. It has four cores and each core’s clock speed is 900MHz. We modify AUTOSAR 2.0 as RTOS to run on secure world and SMP Linux as GPOS on normal world. Since AUTOSAR 2.0 does not support multicore processors, core 0 is dedicated to AUTOSAR 2.0. On the other hand, SMP Linux runs on the remaining cores.

A. Software isolation overhead

To show the overhead of SASP, we measure the execution time of LMBench3 [14] workload in the normal world when the secure world is idle. Figure 5 is the normalized execution time of arithmetic workload of LMBench3. The result shows the overhead caused by SASP is negligible as the measured performance loss is within 1%.
Figure 6 depicts the execution time of system call in the normal world. System call benchmark shows the performance of switching privileged level (user and kernel) by access to a block I/O device. Unlike arithmetic workload, the system call benchmark shows that SASP causes 5% of the performance loss, for instructions except open/close instruction. However, the performance degradation is much smaller than the software isolation using hypervisor based on software.

To evaluate access latency to the shared device, we measure the latency when accessing shared device. Figure 7 presents the latency of request and response. In SASP, the request is transmitted from normal world to secure world, and the response is measured in the opposite direction. As shown Figure 7, the average latency of the request and response is as small as 658 to 733 nanoseconds, respectively. This means that the overhead of request and response is negligible because the overhead is much smaller than the time spent for data copy from shared memory to world's memory that is normally 1-2 ms for 10KB.

But, there is asymmetry in request and response times in Figure 7. Figure 8 illustrates the execution flow of SMC and IPI handling for the secure device access. The dashed line represents “Request” from Linux and the black line means “Response” from AUTOSAR. Linux sends the request using SMC and IPI that is handled by FIQ vector in AUTOSAR, whereas AUTOSAR responses using IPI that is handled by IRQ vector in Linux. So this means that FIQ processing time is shorter than that of IRQ.

Figure 9 shows bandwidth between V-Bridge from Linux to V-Bridge of AUTOSAR according to buffer size of shared memory. In this experiment, buffer size is equal to the size of data which is transmitted at once. So, if the buffer size is...
increased, data copy overhead is also increased. It means that the transfer rate should be regulated by buffer size. As shown in Figure 9, the peak bandwidth achieves 5MB/s when buffer size is 20Kbytes. This result presents that our secure device access mechanism satisfies the bandwidth requirement of CAN because the maximum bandwidth of CAN standard is 1MB/s. The result is the same in both “Secure to Normal” and “Normal to Secure”.

VI. CONCLUSION

This paper presents a secure device access method for automotive software based on TrustZone. From previous studies, hardware partitioning provided by TrustZone helped to develop software integration. However, device access methods, that are proposed by other researches, do not consider security issues. Additionally, their device access method works only on single core processor. Thus, it is difficult to apply prior works to multicore processors. By those reasons, we propose SASP that has secure device access method using V-Monitor and V-Bridge. To access device securely, V-Monitor provides admission control and V-Bridge offers behavior control. From performance evaluation, SASP has about 1% performance degradation in arithmetic operations and under 5% degradation in system call operations except open/close system calls. The measured bandwidth of V-Bridge is up to 5MB/s.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No.2010-0029180) with KREONET.

REFERENCES


