Lightweight Secure Sensing Using Hardware Isolation

Mengmei Ye Nianhang Hu Sheng Wei
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115, USA

***Abstract*—This paper develops a new lightweight secure sensing technique using hardware isolation. We focus on protecting the sensor from unauthorized accesses, which can be issued by attackers attempting to compromise the security and privacy of the sensed data. We satisfy the security requirements by employing the hardware isolation feature provided by the secure processor of the target sensor system. In particular, we deploy the sensor in a hardware isolated secure environment, which eliminates the potential vulnerability exposed to unauthorized attackers. We implement the hardware isolation-based secure sensing approach on an Xilinx Zynq-7000 SoC leveraging ARM TrustZone. Our experiments and security analysis on the real hardware prove the effectiveness and low overhead of the proposed approach.**

I. Introduction

Secure sensing plays a significant role in the sensor applications, which refers to the protection of security sensitive sensor data. Without such protection, attackers can leverage the security vulnerabilities of the sensor systems and gain unauthorized access to the sensed data or even falsify the data, which can contribute to adverse influence on security and privacy [1]. Therefore, it is critical to prevent the leakage of secret information and enforce authorized requests to the target sensor systems.

Sensor data encryption is the most widely used technology to date to achieve secure sensing, such as preventing eavesdropping from sensors [1]. Mostly, it uses the existing cryptosystems, such as RSA, AES, and homomorphic encryption to protect the sensed data [2]. However, it often introduces significant timing and resource overhead [3].

To address the secure sensing challenge given the resource constraints, we develop a new lightweight sensor data protection mechanism by employing hardware isolation from the physical layer of the sensor system. Our hardware isolation approach is feasible in the state-of-the-art security-enabled processors, such as ARM TrustZone [4] and Intel SGX [5].

In particular, we deploy the sensor under protection into a hardware isolated "secure world" of the system on chip (SoC), which prevents unauthorized "normal world" applications from accessing the sensor data. On top of the foundational hardware isolation, we implement a secure application in the secure world, which invokes the requested operations on behalf of the software application. It ensures that the sensor under protection only delivers the sensor data to the authorized applications. During the secure sensing with hardware isolation, the only timing overhead is the switching delay between the secure and non-secure modes of the application, which proves to be minimum based on our implementation and evaluation on a real hardware prototype.

II. Related Work

ARM TrustZone [4] is one of the most representative hardware isolation-based techniques. It separates the software and hardware into a secure world and a normal world, which are isolated at the physical bus level. The hardware isolation-based configuration guarantees that the resources in the secure world are not directly accessible by the normal world. Also, a secure monitor controls the context switching between secure and normal worlds, where completely different and isolated memories and registers are used.

ARM TrustZone and other existing hardware isolation techniques have been adopted in many software security applications, such as one-time password [6], kernel monitor [7], and trusted language runtime system [8]. To the best of our knowledge, our work is the first to protect sensor data leveraging hardware isolation.

III. Hardware isolation-based Sensor Framework Design

We develop a hardware isolation-based sensor protection mechanism through employing a hardware isolation feature provided by ARM processors, namely ARM TrustZone [4]. Figure 1 shows the overall framework of our proposed design. We deploy a sensor IP into the secure world in order to isolate it from the potentially compromised applications in the normal world. An normal application is able to request sensor data by issuing a secure monitor call (SMC) to the secure monitor, which in turn switches the application mode from the normal world to the secure world. Then, a secure application invokes the sensor service on behalf of the normal world application. We further deploy a shared memory in the normal world to store the sensor data from the secure world.

Based on the proposed framework, the entire procedure of the sensor can be described as the following:

1) The normal application sends a request to the secure monitor for invoking the sensor IP.
2) The secure monitor switches the application mode from the normal world to the secure world.

978-1-4799-8287-5/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: Rutgers University. Downloaded on October 06,2020 at 01:29:09 UTC from IEEE Xplore. Restrictions apply.

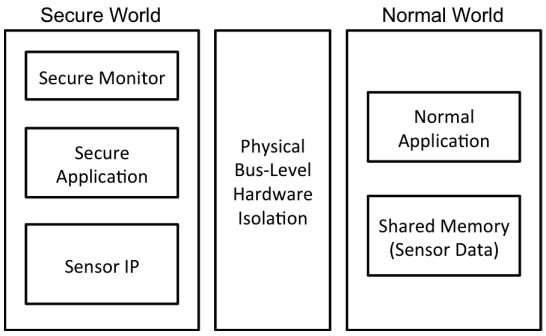

Fig. 1. Overall framework and data flow of the proposed secure sensing approach using hardware isolation.

3) The secure application receives the request, and verifies whether the request is from an authorized application.
4) For requests from authorized normal applications, the secure application further checks the number of continuous requests made by this application, to prevent potential denial of service (DoS) attacks. For unauthorized requests, the secure application will decline them immediately.
5) The secure application invokes the sensor, only if the number of continuous requests is less than the predefined threshold.
6) The secure application writes the sensor data into the shared memory.
7) The secure monitor changes the world state from the secure world to the normal world.
8) The authorized normal world application reads the sensor data from the shared memory.

It is worth noting that secure monitor plays an essential role in the entire secure data flow. During each world switch, the secure monitor saves all the application and CPU states in the stack and will restore them when the context is switched back. The world switches managed by the secure monitor ensure that there is no sharing of data with any internal variables or registers between the two worlds. Instead, the input and output sharing is completed by the shared memory in the normal world, and the trusted secure application ensures that the sensor data can be delivered to the shared memory following the authorized request.

IV. Prototype Implementation

We implement the hardware isolation-based sensor system on Xilinx Zynq-7000 SoC XC7Z020-CLG484-1 (i.e., the ZedBoard). The ZedBoard has a dual ARM Cortex-A9 MPCore processor that supports TrustZone, with a CPU rate up to 667MHz. Also, the Zynq SoC includes a block that has on-chip sensors to monitor the temperature and the voltage, namely the AXI XADC IP.

Figure 2 shows the block design of the XADC IP in the Zynq processing system (PS) using the AXI interconnect, which enables us to obtain the temperature data from this on-chip sensor and associate it with ARM TrustZone. The XADC IP includes 32 16-bit registers to initialize and control the sensor operation, which can be accessed by DRP or JTAG ports on the board [10].

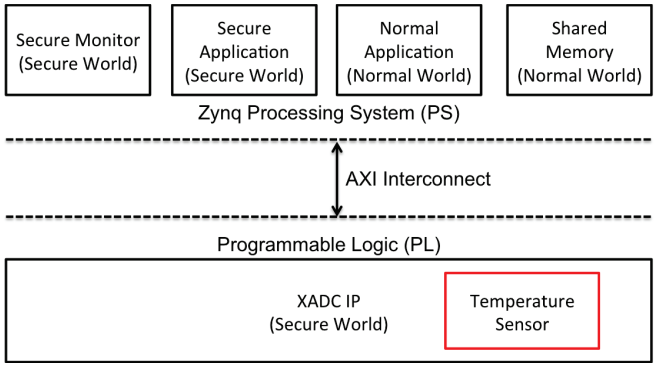

Fig. 2. Block design for hardware isolation-based secure sensing.

We implement the secure monitor and secure application for the secure world, as well as the normal application for the normal world in the PS part of the SoC. In our prototype implementation, we use a white list that stores authorized users in the secure application to approve the sensor service requests. If the request from the normal world is authorized, the temperature sensor data is delivered to the shared memory. If the secure application detects that the service request is from an unauthorized application or the normal world application has issued too many requests (e.g., over 1000 requests), the secure application will decline the service from this normal application.

V. Evaluation Results

A. Security Analysis

Based on the prototype system, we conduct security analysis on the proposed hardware isolation-based secure sensing framework. First, our framework is resilient to software attacks, which avoid unauthorized data access to the sensor. In our prototype implementation, the normal application must issue an SMC to the secure monitor and request a world switch. Then, the secure application takes over and determines whether the sensor data can be delivered to the shared memory. The hardware isolation framework ensures that the secure application never leaks the data to unauthorized applications. Second, our approach can prevent denial of service attacks and reverse engineering attacks, because the secure application limits the number of continuous and frequent requests from the normal applications.

B. Performance Overhead Analysis

We further evaluate the performance overhead of the hardware isolation-based secure sensing approach and compare it with the widely used encryption-based approach, as well as the baseline approach without any security protection. Based on the framework and data flow described in Section III, the delay of the hardware isolation-based approach involves the two world switches during the service call of the temperature

Authorized licensed use limited to: Rutgers University. Downloaded on October 06,2020 at 01:29:09 UTC from IEEE Xplore. Restrictions apply.

TABLE I
PERFORMANCE OVERHEAD ANALYSIS ON DIFFERENT SECURE SENSING APPROACHES. THE DATA INDICATE THE DELAY WITH DIFFERENT NUMBER OF REQUESTS FOR EACH APPROACH.

Number of Requests	No Protection (us)	Hardware Isolation-Based (us)	Encryption-Based (us)
16	20,872	20,876	23,214
32	41,742	41,748	46,202
48	62,612	62,620	69,190
64	83,482	83,492	92,179
80	104,352	104,364	115,167
96	125,222	125,235	138,155
112	146,092	146,108	161,143
128	166,961	166,979	184,131

sensor, one from the normal world to the secure world, and the other from the secure world to the normal world. During each switch, the secure monitor must detect from which world the SMC was issued, save the states of the source world, and restore the states of the destination world. Table I shows the delay of the three approaches for different number of requests. We implement the encryption-based approach by encrypting the sensor data after it is captured and decrypting it before use. The cryptosystem we employ is AES in CBC mode by using the Tiny AES128 library [9]. Then, we time the delay of executing the sensor and gaining the data. Also, for each approach, the application keeps requesting the XADC sensor service to acquire the temperature data many times (i.e., 16 to 128 times).

Furthermore, we calculate the disparity rates on each group of request number, so that we are able to compare the performance of different approaches. The disparity rates show the delay differences between the corresponding secure sensing approach (i.e., hardware isolation-based or encryption-based) and the baseline approach (i.e., no protection), which can be expressed as Equations (1) and (2).

$$Disparity\ Rate\ 1 = \frac{Isolation\ Delay - NoProtection\ Delay}{No\ Protection\ Delay} \quad (1)$$

$$Disparity\ Rate\ 2 = \frac{Encryption\ Delay - No\ Protection\ Delay}{No\ Protection\ Delay} \quad (2)$$

Based on the results in Table I, the average $Disparity\ Rate\ 1$ is 0.0127%, and the average $Disparity\ Rate\ 2$ is 10.5%. Therefore, the delay of hardware isolation-based approach is only slightly higher than the delay of no protection, which indicates minimum performance overhead. Also, the hardware isolation-based approach takes much less time than the encryption-based approach, which represents significant performance savings.

VI. CONCLUSION

We developed a hardware isolation-based secure sensing mechanism, leveraging ARM TrustZone, to protect security and privacy sensitive sensor data. Our hardware isolation framework ensures that the sensor data is strictly isolated in the secure world with no exposure to potentially malicious normal world applications. We implemented the proposed approach on Xilinx Zynq SoC and analyzed its security enhancement and performance overhead.

REFERENCES

[1] H. Chan and A. Perrig, "Security and Privacy in Sensor Networks", IEEE Computer, pp. 103-105, 2003.

[2] F. Li, B. Luo, and P. Liu, "Secure Information Aggregation for Smart Grids Using Homomorphic Encryption", Proceedings of the First IEEE International Conference on Smart Grid Communications, pp. 327-332, 2010.

[3] H. Shafagh, A. Hithnawi, A. Droescher, S. Duquennoy, and W. Hu, "Talos: Encrypted Query Processing for the Internet of Things", Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, pp. 197-210, 2015.

[4] "ARM Security Technology: Building a Secure System Using TrustZone Technology", http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/index.html

[5] "Intel Software Guard Extensions", https://software.intel.com/en-us/isa-extensions/intel-sgx

[6] H. Sun, K. Sun, Y. Wang, and J. Jing, "TrustOTP: Transforming Smartphones into Secure One-Time Password Tokens", Proceedings of the 22nd ACM Conference on Computer and Communications Security, pp. 976-988, 2015.

[7] A. Azab, K. Swidowski, R. Bhutkar, J. Ma, W. Shen, R. Wang and P. Ning, "SKEE: A Lightweight Secure Kernel-Level Execution Environment for ARM", Proceedings of the 2016 Network and Distributed System Security Symposium, 2016.

[8] N. Santos, H. Raj, S. Saroiu, and A. Wolman, "Using ARM TrustZone to Build a Trusted Language Runtime for Mobile Applications", Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 67-80, 2014.

[9] "Tiny-AES128-C", https://github.com/kokke/tiny-AES128-C.

[10] "7 Series FPGAs and Zynq-7000 All Programmable SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide", UG480 (v1.8), Xilinx, 2016.

Authorized licensed use limited to: Rutgers University. Downloaded on October 06,2020 at 01:29:09 UTC from IEEE Xplore. Restrictions apply.