Secure MCU Platform Techniques to Ensure Security of Cyber-Physical Systems

With the rapid spread of Internet of Things (IoT) devices as key components for the construction of cyber-physical systems (CPS) in recent years, demand has been growing for a microcontroller unit (MCU) for IoT devices with both a secure communication function for connecting to the network and a firmware update function to enhance integrity and availability.

Toshiba Electronic Devices & Storage Corporation has been implementing measures to address this need through the development and introduction of the following techniques: (1) introduction of a software platform technique using the Arm® Mbed™ OS, which is an open-source embedded operating system (OS), to its MCUs with an Arm® Cortex®-M processor in order to achieve secure connection to the network and reduction of the burden on developers, and (2) development of a secure firmware rotation technique using redundant flash memories for firmware updates that can protect IoT devices against cyberattacks in order to achieve a balance between integrity and availability. We have confirmed the effectiveness of the secure firmware rotation technique through experiments on functional prototype hardware and software using a field-programmable gate array (FPGA).

1. Introduction

CPS are rapidly spreading, integrating information technology (IT), factory automation (FA) innovations, and the Internet of Things (IoT) that have evolved separately to date. IoT devices, which serve as an interface between the cyber and physical worlds for CPS, should provide secure network communication and a firmware updating function without compromising either integrity or availability. “Integrity” means guarding against improper modification of firmware while “availability” means ensuring that it remains in an operable state.

In the case of conventional MCUs for embedded devices, firmware development for motors, machine tools, sensors, and other system functions accounts for a large proportion of the total workload. In contrast, MCUs for IoT devices, which require network connectivity, impose a considerable workload on appliance vendors for the development of firmware. Furthermore, security requirements for IoT devices include a high level of availability in addition to the prevention of system malfunctions and the safeguarding of confidential information.

In consideration of these requirements, Toshiba Electronic Devices & Storage Corporation has adopted Mbed™ OS from Arm Limited for its MCUs with a Cortex®-M processor in order to achieve secure network connection and reduce the development workload of appliance vendors. In addition, we have developed a secure firmware rotation technique as a differentiator for the MCU hardware that provides a secure means of firmware updating without compromising availability. This report outlines Mbed™ OS and the newly developed secure firmware rotation technique.

2. Application of Mbed™ OS to MCUs with a Cortex®-M processor

Tasks that require parallel and real-time execution are implemented in the MCUs for embedded devices according to the

objectives of system developers. Therefore, embedded systems are typically designed to run a real-time operating system (RTOS) so as to provide a performance guarantee for task execution. There are many RTOSs with different architectures.

From these RTOSs, we have selected an RTOS called Mbed™ OS that is optimized for MCUs with a Cortex®-M processor. Mbed™ OS provides a useful platform that supports firmware developers in system architecting. The following paragraphs describe Mbed™ OS and its platform.

Mbed™ OS inherits the conventional RTOS architecture as well as basic functions that have been provided by Arm Limited, including inter-task communication and resource-sharing functions. To use Mbed™ OS, firmware developers need to port Mbed™ OS to the target board with a Cortex®-M-based MCU and the DAPLink debug interface and obtain certification from Arm Limited. DAPLink connects the certified board with a firmware development platform available with Arm's cloud services. This firmware development platform is one of the key features of Mbed™ OS.

Another feature of Mbed™ OS is the Arm® Pelion™ IoT Platform, a cloud service that facilitates the connection of IoT devices running Mbed™ OS to a network. Pelion™ IoT Platform provides functions to establish and manage the network connections of IoT devices and to build secure communication links using Arm's Mbed™ TLS (Transport Layer Security) that provides cryptographic communication functions (**Figure 1**). Mbed™ TLS supports the latest cryptographic algorithms and provides basic network security functions, for example, using communication functions compliant with the Transmission Control Protocol/Internet Protocol (TCP/IP).

These two features play a pivotal role in the platform for secure IoT devices requiring a network connection and help to achieve a substantial reduction in the workloads for firmware development and system operation.

Mbed™ OS can be used with our TX and TXZ families of MCUs with a Cortex®-M0, Cortex®-M3, or Cortex®-M4 processor that are suitable for a wide range of applications. The TX and TXZ families include the TMPM46BF10FG with a Cortex®-M4 processor that is well suited to serve as a secure platform for IoT devices (**Figure 2**). The security engine incorporated in the TMPM46BF10FG helps to enhance the throughput of cryptographic communications using Mbed™ TLS.

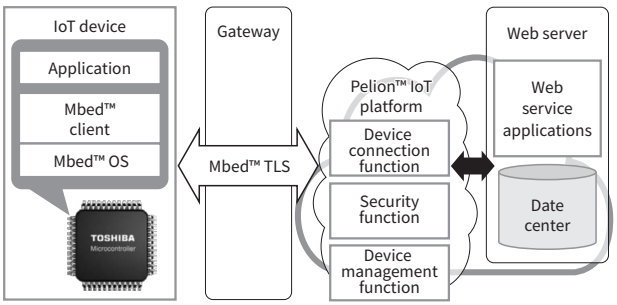
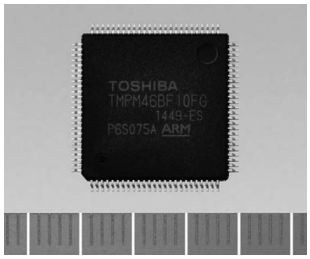
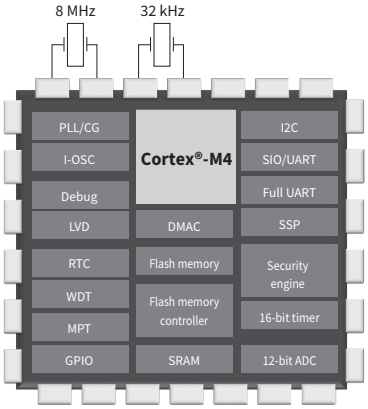


Figure1. Configuration of software platform applying Mbed™ OS

The use of Mbed™ OS for the MCUs for IoT devices makes it possible to improve the efficiency of firmware development and IoT device management in tandem with the cloud services from Arm Limited.



(a) TMPM46BF10FG



(b) Block diagram

PLL/CG : phase-locked loop/clock generator
I-OSC : internal oscillator
LVD : low-voltage detect
RTC : real-time clock
WDT : watchdog timer
MPT : multi-purpose timer
GPIO : general purpose input/output
DMAC : direct memory access controller
SRAM : static RAM
I2C : inter-integrated circuit
SIO/UART : serial input output/universal asynchronous receiver transmitter
SSP : synchronous serial port
ADC : analog-digital converter

Figure2. Block diagram of TMPM46BF10FG MCU with Cortex®-M4

The TMPM46BF10FG incorporates various functions necessary to create secure IoT devices, including computing units, various input/output(I/O) interfaces, flash memory, and a security engine.

3. Overview of the secure firmware rotation technique

3.1 Security threats to and constraints on IoT devices

As described in Section 2, the use of Mbed™ OS for an existing MCU facilitates the development of IoT devices with a network security function. However, there are many security issues to be addressed regarding IoT devices, including countermeasures for software vulnerabilities. It is not enough for critical control systems to simply use Mbed™. In-depth security protection is also important for these systems. The workload of firmware developers can be further reduced by incorporating such additional security measures into an MCU as one of the elements of a platform.

Since IoT devices interact with the physical world, their malfunctions could cause personal or property damage. Access control and other security functions are essential to protect IoT devices from unauthorized access and thereby a malfunction. IoT devices also require adequate levels of availability and reliability. However, a function to block unauthorized access should not be added inadvertently because it might reduce the availability of IoT devices if use cases are missed. Examples of security functions for non-IoT devices include login lockout for PCs, a feature that locks a PC after a given number of unsuccessful password attempts. Although this feature is useful to prevent security threats arising from the use of stolen PCs, its downside is that it reduces a PC's availability if the user forgets its password and enters the wrong password several times consecutively. Therefore, a login lockout feature must be accompanied by a mechanism to unlock the locked PC due to failed user authentication and issue a new password.

One of the reasons that make it difficult for the firmware of IoT devices to achieve high levels of integrity and availability lies in the need to obtain firmware updates via a network. Remote firmware updating is a fundamental requirement for IoT devices not only to enhance their functionality and correct bugs but also to fix software vulnerabilities and thereby maintain their security over the long term. Since there are public guidelines for firmware updating for smart meters and PC peripherals[(1)(2)], they are increasingly equipped with a firmware updating function.

For firmware updating, it is essential to use a digital signature or other means of identification so as to verify the authenticity of the downloaded firmware update and to determine, prior to execution, that the program in flash memory has not been tampered with. This is required to eliminate the possibility that a software vulnerability might be exploited by malware to tamper with the software program in flash memory after an authentic firmware update is received. The term "software vulnerabilities" collectively refers to a type of system weakness that can be exploited for security attacks. There are various types of software vulnerabilities such as arbitrary code execution. Arbitrary code execution is triggered when a computer receives unintended parameters from an attacker that are not usually used, causing such data to be executed as part of an authentic software program. The malware delivered over a network might rewrite data in a region of the flash memory in which a firmware update is to be written. To remove such software vulnerabilities, it is necessary to remove all the effects of malware prior to the execution of the firmware update and determine that it has not been tampered with.

It should be noted that, even in the event of an arbitrary code execution exploit, a computer recovers from its effects when it is rebooted, as long as the tampered firmware remains in a static RAM (SRAM). However, the effects of the arbitrary code execution exploit persist if it alters the firmware update in flash memory. To counter this threat, it is important to determine when and how to perform a final verification of the firmware update to ensure that it is completely free from tampering.

To prevent a malfunction, if even one bit of firmware has been altered, such firmware should not be executed. This is because this bit might indirectly affect an important conditional branch if it holds a parameter that determines whether to branch to a different code path. An appropriate use of digital signature technology for firmware makes it possible to detect and eliminate such tampering.

However, an attacker's intent might be to reduce the availability of the target computer, or in other words, to make it unusable (by destroying software). In that case, rigorous tampering verification makes it easier for the attacker to accomplish his/her purpose. It is therefore difficult to realize firmware that combines high levels of integrity and availability.

3.2 Issues and requirements for firmware updating

This section discusses the requirements for a firmware updating

function.

As described in Section 3.1, firmware updating is necessary to address software vulnerabilities in IoT devices. In consideration of maintainability, the first requirement is the ability to remotely update firmware without relying on on-site service personnel.

In small IoT devices, firmware is stored in and executed from flash memory. This means an IoT device needs to provide a mechanism to rewrite its flash memory. Therefore, the second requirement is to address the threat of the exploitation of this mechanism by attackers. It is necessary to prevent not only the execution of malware sent by attackers but also the destruction of software as described in Section 3.1.

There are also requirements that should be satisfied because of MCU constraints. PCs generally execute a software program from dynamic RAM (DRAM) after it is copied from flash memory to DRAM whereas MCUs use execute-in-place (XIP), i.e., a method of executing firmware word by word directly from flash memory rather than copying it into SRAM. This is because the per-bit cost of SRAM is much higher than that of flash memory. Many kinds of firmware are executed without an OS or as a single integrated image of an OS kernel and an application without a filesystem. Therefore, firmware updating involves the rewriting of a set of system programs. This is the third requirement.

The fourth requirement is a consideration for the lack of a sophisticated privilege control function in low-end MCUs, i.e., an ability to switch between privileged and non-privileged modes of operation at arbitrary timing during execution.

The fifth requirement is to ensure that a secure state can be recovered even in the event of a vulnerability in an old version of firmware being exploited for malware execution. This is possible by guaranteeing secure firmware updating compliant with the second requirement.

When firmware is remotely updated in an environment that satisfies all the above requirements, it is essential to verify the firmware received via a network. A simple updating method directly rewrites the firmware in flash memory while it is being executed. However, if an IoT device is rebooted before the firmware updating is completed, the IoT device will not function properly after reboot. In case this occurs, many IoT devices retain one or multiple firmware backups.

3.3 Development approach

In order to satisfy the five requirements described in Section 3.2, the newly developed secure firmware rotation technique: (1) allocates two firmware storage regions in flash memory; (2) provides a support software program called the ROM monitor that is specifically designed for firmware verification and contained in the on-chip mask ROM of an MCU; and (3) uses an MCU with a simplified hardware access mechanism[3]. This technique is characterized by the ROM monitor contained in an untamperable ROM region and an access control method that write-protects a storage region for the verified firmware until the ROM monitor completes the verification of the firmware update.

Figure 3 shows an outline of the secure firmware rotation technique. The left-hand side of Figure 3 indicates the MCU condition before firmware updating while the right-hand side shows the MCU condition after firmware updating. The flash memory in which firmware is stored is partitioned into two regions, A and B. Prior to updating, firmware1 in region A is being executed. Since region A is write-protected, firmware1 cannot be modified. On the other hand, region B, which is reserved for firmware

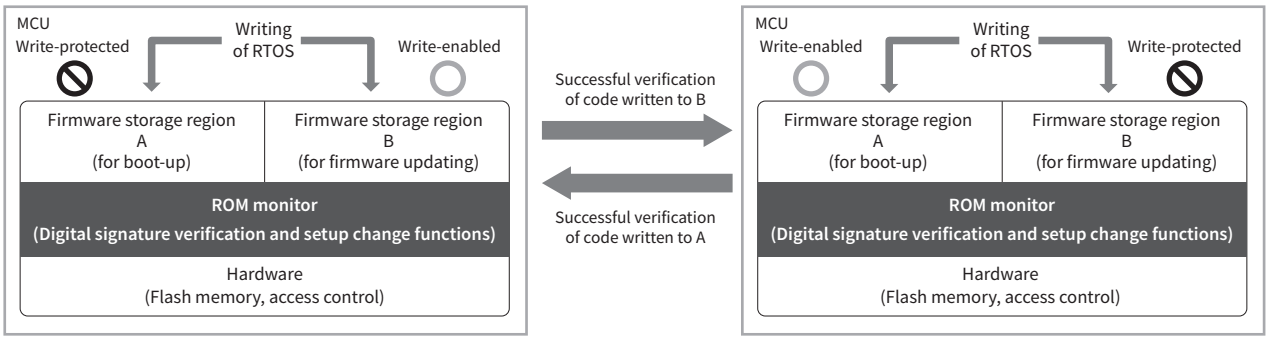


Figure 3. Outline of secure firmware rotation technique

The secure firmware rotation technique uses two firmware storage regions in flash memory. One of these regions is write-protected for bootup while the other one is write-enabled for firmware updating. The ROM monitor switches these regions only when it is determined that the firmware written into the updating region is authentic.

updating, is write-enabled so that a firmware update can be written freely into region B. While firmware1 is being executed, a digital signature is received via a network to guarantee the authenticity of the firmware update. Following appropriate verification, the firmware update is written into region B as firmware2.

After this write operation is completed, firmware1 sets the Updated flag in flash memory to issue a reset and reboot the IoT device. The ROM monitor is invoked upon reboot. At this time, the ROM monitor has a privilege to freely access the entire flash memory. When the ROM monitor determines that the Updated flag is set, it verifies the digital signature in region B for firmware updating. If the verification of the digital signature is successful, the ROM monitor changes the access control settings for the flash memory. Specifically, it write-protects region B and write-enables region A to execute firmware2, i.e., the updated version of the firmware, in region B. The right-hand side of Figure 3 shows this state. When a need for firmware updating arises the next time, the roles of the two flash memory regions revert to the state shown in the left-hand side, rotating the regions in which the latest firmware resides.

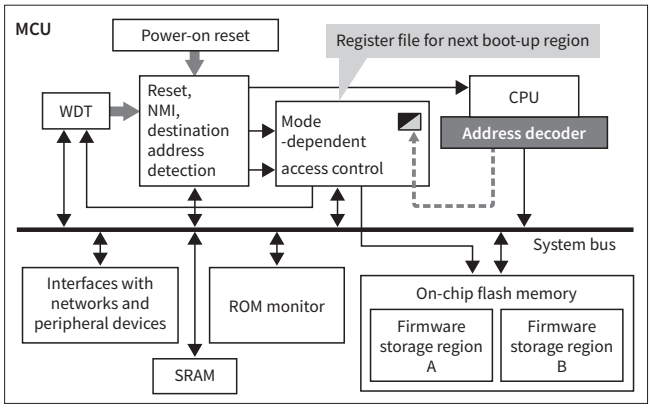
During the firmware updating process, the ROM monitor, which is invoked after a reset, controls the selection of the version of firmware to be verified and executed. For example, even if an attack against a software vulnerability causes the firmware to malfunction temporarily, the possibility of malware execution can be eliminated by rebooting an IoT device. Since the ROM monitor and firmware rotation are not executed concurrently, the secure firmware rotation technique can be applied to low-end MCUs without privileged modes.

Furthermore, the secure firmware rotation technique allows secure firmware updating even in the event of a zero-day attack, i.e., an exploit that adversely affects an IoT device before its vendor learns of the vulnerability or releases a firmware update. Since the secure firmware rotation technique uses an independent ROM monitor for firmware verification, it is not affected even when the old version of firmware is attacked. If the vulnerability is fixed in the updated version of firmware, the vulnerability is removed once an IoT device is rebooted following the firmware verification by the ROM monitor. Since the firmware region containing the old version of firmware is write-protected until successful verification of a firmware update, the old version of firmware is protected from destruction even if it has any vulnerabilities. The rebooting of an IoT device and the acquisition of a firmware update can be retried any number of times until the firmware update is successfully verified by the ROM monitor. It is true that remote firmware updating requires a longer time per IoT device than on-site firmware updating, particularly under influence of a cyberattack. However, the number of on-site firmware updates that can be processed in parallel is constrained by the number of available servicepersons whereas the number of remote firmware updates is constrained by the network and server capacities. Remote firmware updating is more efficient in cases where numerous sensor nodes or smart meters are connected to a network.

3.4 Functional prototyping

We created functional hardware and software prototypes incorporating the secure firmware rotation technique using a single-chip MCU with a Cortex®-R4 processor from Arm Limited and on-chip flash memory (**Figure4**). We utilized a logic synthesis tool to map the entire hardware design, including the Cortex®-R4 processor, to a FPGA. A function equivalent to the ROM monitor and the firmware (i.e., an RTOS and an application) were implemented as software. The ROM monitor employs digital signatures using the RSA-2048 and SHA-256 hash functions to verify the integrity of the firmware update. The footprint of the ROM monitor was roughly 18 Kibytes (Kibyte: kibi (2^{10}) bytes), including digital signatures and data. Electronic devices designed for long-term use have been increasingly adopting RSA-3072 and SHA-384 with long keys. It is estimated that the use of these cryptosystems will cause an increase of only a few Kibytes in the size of the ROM monitor.

The functional prototypes demonstrated that the newly developed secure firmware rotation technique works properly.



NMI : non-maskable interrupt (interrupt that cannot be masked by software)
I/F : interface

Figure4. Block diagram of prototype MCU incorporating secure firmware rotation technique

A prototype MCU with a Cortex®-R4 processor and on-chip flash memory was implemented as an FPGA to verify the function of the secure firmware rotation technique.

4. Conclusion

This report described our initiatives for MCUs for IoT devices that provide an interface between the cyber and physical worlds interlinked by CPS. The application of Mbed™ OS to our existing MCUs makes it possible to establish and manage the network connections of IoT devices easily and securely. The secure firmware rotation technique, a differentiator for the MCU hardware, enables secure firmware updating without compromising the availability of IoT devices. We are currently working on its commercialization.

In addition to the security of IoT devices discussed herein, digital authentication and other trust services are important for the realization of secure IoT devices. We have concluded a business alliance with Cybertrust Japan Co., Ltd. that has a proven track record in digital authentication and commenced studies toward the establishment of a comprehensive trust service platform using an MCU that will be newly developed as a key component[(4)].

To contribute to the enhancement of the security of CPS, we will continue to develop MCU-related technologies that meet market requirements and provide MCUs designed to protect the integrity and availability of IoT devices.

References

(1) Japanese Electrotechnical Standards and Codes Committee. 2016. JEAG1101-2016. “Guidelines for Smart Meter System Security.”

(2) National Institute of Standards and Technology (NIST). 2018. NIST Special Publication 800-193:2018. “Platform Firmware Resiliency Guidelines.”

(3) Hashimoto, M. et al. 2016. “Security technologies for non-volatile memory in IoT devices: Mechanisms for on-chip MCU and off-chip memory system.” IEICE technical report. **116**(240): 37–42.

(4) Toshiba Electronic Devices & Storage Corporation, Cybertrust Japan Co., Ltd. 2019. “Toshiba Electronic Devices & Storage Corporation and Cybertrust Japan Co., Ltd. Conclude a Deal on Trust Services for IoT Equipment (in Japanese).” News release / Products / Public Information. Accessed July 12, 2019. https://toshiba.semicon-storage.com/jp/company/news/news-topics/2019/07/micro-20190709-1.html.