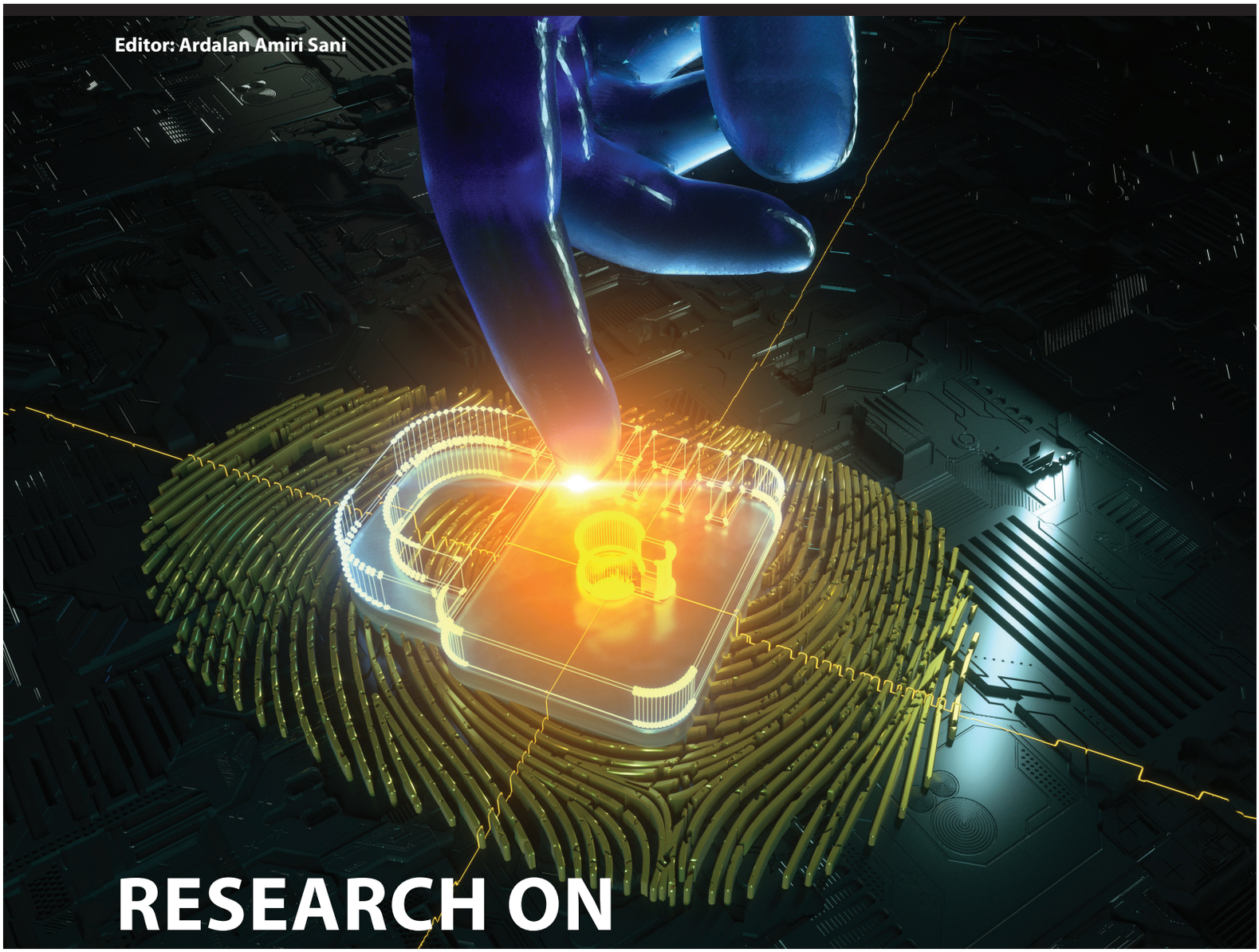
Wenhao Li, Yubin Xia, Haibo Chen *Shaghai Jiao Tong University*

Editor: Ardalan Amiri Sani



RESEARCH ON ARM TRUSTZONE

ARM TrustZone [1] is a hardware-based security feature that can provide software with a high-privilege and isolated execution environment. Such isolation is ensured by hardware, which is usually considered as more trustworthy than software. Thus the execution environment is also known as trusted execution environment (TEE). TrustZone technology was proposed in 2002, but did not get widely used until 2009, when Apple released iPhone 5s. In iPhone 5s, Apple leveraged TrustZone to protect its Touch ID, which ensures that even if the iOS is fully compromised, the user's fingerprint data can still be safe. In 2017, Google made TEE a mandatory requirement on any Android devices with a fingerprint scanner. Nowadays, almost all mobile phones and tablets have TEE deployed. Meanwhile, ARM integrates TrustZone in ARM64 and ARMv8-M to support a broader range of platforms including servers and IoT devices.

The architectural features provided by TrustZone are attractive to researchers. Many researchers try to answer this question: "What is the right way to use TrustZone?" and propose many systems to explore different usages. There are many challenges: first, the system should be easy to use. Some researchers propose integrating TrustZone with the existing system for minimal intrusion to users. Second, the code added to the secure world should be as little as possible. The functionalities offered by TEE should be general to keep a small trusted computing base (TCB). Third, the system should be deployable, which should not require significant modifications to existing systems software or applications. There are also several other challenges like extensibility, attack surface, performance, etc., which will be addressed later.

HARDWARE AND SOFTWARE

ARM TrustZone [1] has been proposed since ARMv6 architecture, which includes security extensions to ARM System-On-Chip (SoC) covering the processor, memory and peripherals. For the processor, TrustZone splits it into two execution environments, a normal world and a secure world (as shown in Figure 1). Both worlds have their own user space and kernel space, together with cache, memory and other resources.

The normal world cannot access the secure world's resources while the latter can access all the resources. Base on this asymmetrical permission, the normal world is used to run a commodity OS, which provides a Rich Execution Environment (REE). Meanwhile, the secure world always uses a secure small kernel (TEE-kernel). The two worlds can switch to each other under the strict supervision of a Secure Monitor running in monitor mode. Typically, a special instruction called "secure monitor call" (smc) is used for worlds switching.

Memory Partitioning: TrustZone divides the memory into two parts: normal part and secure part, which are distributed into normal world and secure world accordingly. Again, TrustZone ensures that the normal world cannot access the secure part of memory while the secure world can access the entire memory. With this feature, two worlds can communicate with each other by using a piece of shared memory. Besides, the memory partition can be dynamically controlled by the secure world, which gives secure services running in the secure world the ability to dynamically protect certain parts of the memory.

Peripheral Partitioning: For I/O devices and interrupts, TrustZone also splits them into two worlds. An I/O device can be assigned to one specific world. TrustZone ensures that the normal world cannot access the secure world's I/O devices while the secure world can control the whole system's devices. For each interrupt, TrustZone can designate the world to handle it. When a secure interrupt arrives, TrustZone will switch the processor to the secure world to handle it. Similar to memory, the partitioning of I/O devices and interrupts can be dynamically configured by the secure world.

Secure Boot: The TEE kernel leverages secure boot to enforce its own integrity. Once a phone is booting, the processor will first enter the secure world and load a TEE image to memory. It then checks the signature of the loaded image to ensure the TEE image has been correctly signed by legal authority (typically the phone vendor). If the check fails, the hardware will stop and hang. The check is done at the very beginning of booting process and cannot be bypassed. After initialization, the security is enforced by the TEE software.

ARCHITECTURAL FEATURES

TrustZone offers several interesting architecture features, which makes it ideal for system security. First, the TEE is isolated from REE, which allows for providing security guarantees without trusting the complex REE OS like Android. Confidential data can be stored and accessed in TEE and can be protected even if malware has taken full control of the REE, like jailbreaking on iOS or rooting on Android.

Second, the TEE has higher privilege than REE, which enables it to monitor and check the REE's running status, like scanning all the REE memory for intrusion detection, integrity enforcement and so on. One challenge is the semantic gap between TEE and REE since TEE can only access the raw memory data without any semantic of the data. The problem is similar as in VMI (VM introspection) and can be mitigated by cooperation between TEE and REE.

Third, TEE can control all the peripherals. Unlike TEE on other platforms like Intel SGX, which focuses on protecting application, TrustZone is a full-system feature that can host both user and system logic, and can control peripheral partition dynamically. This makes it a very good way to protect the I/O path from device to the user, by partitioning both the input and output devices to the secure world.

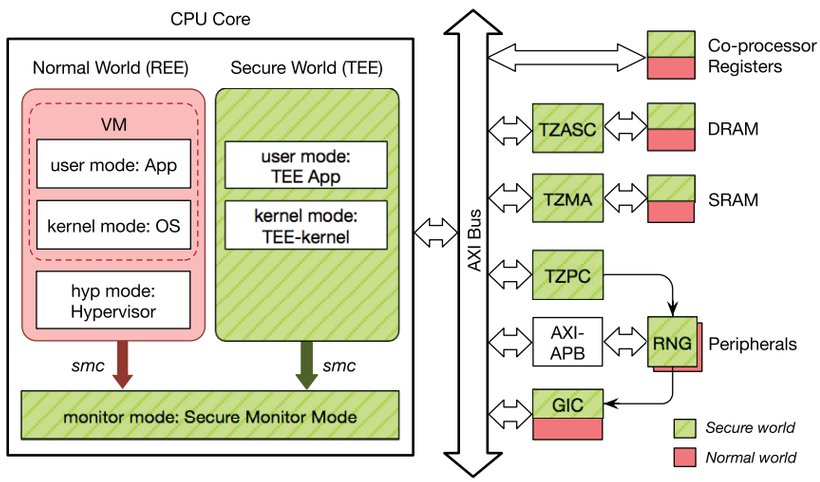


FIGURE 1. The architecture of ARM TrustZone.

RESEARCH ON TRUSTZONE

The features of TrustZone offer new opportunities for enhancing security of systems based on ARM platform, including mobile phones, cloud servers, IoT devices, etc. Currently most of the related researches are on the mobile phone platform, while there are also new work targeting servers and IoT platforms.

AS ARM'S ECOSYSTEM IS EXPANDING FROM MOBILE DEVICES TO IOT, VEHICLES, AND SERVERS, SECURITY IS BECOMING INDISPENSABLE IN MORE AND MORE SCENARIOS

App-Layer: Data Protection

At the very beginning, the typical usage of TrustZone was to host some fixed functionalities, such as biometric authentication, one-time-password, encryption functions, secure storage, etc., which are provided as general secure services of the system. An app in the normal world can invoke these functionalities through API, but cannot install customized logic into the secure world.

TLR (Trusted Language Runtime)[2] enables developers to separate an app's security-sensitive logic from the rest so that it can be isolated in the secure world. It splits the logic of .NET runtime to two parts, one in the normal world and the other in the secure world, and provides four new primitives for developers to use to store and access secret data. The TCB of TLR is 78 times smaller than Mono, an open source .NET runtime. The framework can also be applied to Java runtime.

Rubinov et al. [3] further proposes an approach for automated partitioning of critical Android apps. Specifically, an app is divided to a client code running in the normal world and a TEE commands the part containing the handling of confidential data running in the secure world. It requires the developers to identify the confidential data, and leverages static analysis to extract all the code paths that will access the data, which will further be converted to native code manually.

The challenge of partitioning the app is to define a clear and secure interface between the TEE and REE. First, the code running in the secure world should not depend on the correctness of the code in the normal world. However, such dependency may be app-specific and currently there is no system way to analyze such dependency. Second, an inappropriate interface may increase the opportunities of side channel attacks. For example, consider function-A which maintains a counter. Each time it is invoked it will increase the counter. Assume that the counter is confidential and function-A should be partitioned and run in the secure world. However, an attacker can easily get the counter just by counting the invoking times of the function-A from the normal world. It should be noted that data-driven code partition mechanisms may expose more inner states to the normal world, which increases the attacking surface.

TrustShadow [4] allows an unmodified user application to run in the secure world to isolate the application from the untrusted REE kernel. The TEE kernel does not handle the system calls from the protected application. Instead, it redirects all the system calls to the REE kernel to handle. TrustShadow does not need to consider the partition of app logic and the attack surface now is system calls.

Framework-Layer: UI Protection

Since both the input and display devices can be partitioned to the secure world, it is possible to use TrustZone to protect the process of user input and output, for both privacy and integrity, which is also known as Trust-UI [5]. Trust-UI has an independent UI framework and allows TA direct contact with the user without REE's involvement.

VeriUI [6] proposes a TrustZone based login mechanism to protect the process of password input and transmission. During login, a user only interacts with a browser running in the secure world, which will get user's input and send the password to server. After login, a CA can retrieve an OAuth token from the secure world and use the token for further process. The system focuses on protecting the password to defend against attacks like phishing, malware, or even malicious REE OS. Similarly, TruZ-Droid [7] splits the HTTP and SSL protocols and deploys only half part of the two protocols in the secure world to further reduce the TCB size.

On the contrary, SchrodinText [8] is proposed to protect the output process of text, and extends the range of protection to more types of text fields, including messages, verification codes, etc. It decouples the rendering and displaying of text, leverages the REE OS to render some text without accessing the data of the text but only the number of characters, and uses the TEE to reorder pre-rendered text glyph to display. Thus the untrusted OS will never access the data of text.

Besides protecting the text widget, researchers also use TrustZone to protect other UI widgets like ad and button. AdAttester [9] aims to defend against ad fraud by providing two primitives, unforgeable clicks and verifiable display, with TrustZone. It enables a mobile phone to generate attestations for ad display as well as ad clicks (press). The attestations are further checked by the ad server to ensure the integrity of display and user action authenticity. VButton [10] extends the idea to further support attestation on any press operation by combining a preview to a button widget. The preview contains the semantic of user's operation, which is shown to the user to confirm, and will then generate an attestation by TEE. To simplify the processing in the TEE, a preview is generated by the server before being transferred and displayed in the TEE on a mobile phone.

One important question of secure UI is "How can the user tell whether she is interacting with the secure UI or a fake UI?" One simple but effective way is to introduce an LED indicator controlled exclusively by the TEE OS, which only turns on if the secure UI is working. An alternative way that uses software only would be using a user-chosen picture as the indicator, which is easier to deploy but the initialization of the picture should be taken

special care of. Meanwhile, a system also needs to consider the case that an REE OS overlays the display, which can be avoided by letting TEE exclusively control the framebuffer.

OS-Layer: Kernel Protection

SPROBES [11] uses TrustZone to check the code integrity of REE kernel. By instrumenting the REE kernel, a monitor running in the secure world can be notified and perform security checks. TZ-RKP [12] deprives the REE kernel from the ability of several privileged system functions (e.g., operating the MMU) by removing all the privileged instructions, puts these functions in the secure world, and forces the REE kernel to invoke services provided by a monitor in the secure world to do such privileged operations. Thus the monitor gets a chance to interposition these operations and do security check during runtime, instead of checking periodically.

Similarly, PrivateZone [13] leverages TrustZone to exclusively control the MMU and create PrEE, a memory address space isolated from both REE and TEE in the normal world that can run logic in either user or kernel mode. Since the TCB of PrEE is in TrustZone, it allows developers to create their own isolated execution environment without running app-specific logic in the secure world to keep the TCB small.

Hardware-Layer: Peripheral Protection

Since TrustZone can take exclusive control over peripherals, some researchers leverage it to control these hardware devices with minimal TCB or apply mandatory access control [14]. For example, in scenarios such as a private meeting or a battlefield, it is critical to ensure that certain peripheral (i.e., microphone or radios) are disabled. Instead of turning off the entire mobile device, using TrustZone to turn on/off peripheral could be more flexible. There are two ways to operate the control: one is to set policies by local users through protected UI. The other way is to distribute policies through a remote server, which connects with application in the secure world directly by secure channel.

Another way to use TrustZone is to combine it with sensors on mobile platforms like GPS, to protect the value read from the sensors. Based on TrustZone, Liu et al. [15] proposed two software abstractions, named sensor-attestation and sensor-seal, for exposing trusted sensors to mobile applications and cloud services. The attestation can be very useful especially for crowdsourcing applications, where applications upload sensor readings to a cloud, which combines these values to do further analysis. Here, the integrity of the readings can be critical as, otherwise, attackers may upload incorrect or even malicious data to the cloud.

TrustZone can also be used to implement some hardware features. For example, fTPM [16] uses TrustZone to implement TPM 2.0 interface. TPM (Trusted Platform Module) is a widely deployed chip on PCs and laptops as a key enabler of applications like digital rights management. fTPM enables mobile phones to support TPM by pure software, which is backwards compatible and has better performance than a hardware TPM chip.

Other Platforms: IoT, Drone, Cloud

Besides mobile phones, there are many new types of smart devices, such as smart home assistants, wall-mounted cameras, wearable devices, etc. These devices usually have plenty of sensors, notably cameras and microphones, which collect security sensitive information that should be protected.

Ditio [17] tries to improve the security of IoT devices by recording sensor activity logs with TrustZone and virtualization support. All the access to registers of sensors can be recorded without modification to the REE OS, which will generate a complete log of sensor activities, which will be later be inspected by an auditor and checked for compliance with a given policy. Ditio also provides a tool to ease the process of log analyzing.

PROTC [18] leverages TrustZone on the Drone platform to protect the peripherals. It deploys a monitor in the secure world to enforce secure access control policy for some peripherals of the drone to ensure that only authorized applications can access certain peripherals.

ARM servers are getting popular in cloud computing, which also support virtualization. Currently TrustZone can only provide one secure world, which means that multiple VMs on one server have to share the only secure world. vTZ [19] tries to solve this problem by virtualizing TrustZone to offer each VM an isolated secure world. With vTZ, software in a VM can use SMC instruction to switch between REE and a virtual TEE, just as it does in a non-virtualized environment, and all the features of TrustZone are reserved after virtualization. The implementation requires an REE hypervisor but does not need to trust it.

CHALLENGES

TEE Vulnerabilities

Systems using TrustZone for security rely on one important assumption: the TEE itself is trusted. This assumption came from the fact that less code has fewer bugs. Unfortunately, like most software, the TEE OS and applications can also have vulnerabilities. Back in 2013, it was found that the bootloader of certain Motorola phones can be unlocked to load any system, which is known as CVE-2013-3051. CVE-2016-0825 reveals a vulnerability of the Widevine[1] TA (trusted application) that may leverage TEE kernel to get and leak data stored in TEE's secure storage. Attackers may even use TEE to attack the REE. For example, Boomerang attack [20] leverages bugs of privileged TEE applications to compromise the REE kernel.

How to enforce the security of TEE? The industry is looking for various ways, including stronger isolation, bug finding, formal verification etc., to isolate, find or even eliminate bugs of TEE. For stronger isolation, one important problem to solve is to minimize the attack surface of TEE by authenticating the requests from REE. For example, by enforcing a policy that only one app is allowed to communicate with a certain TA, even if the TA has some bugs, it is hard to trigger since it only accepts requests from one legal app.

Another consideration of TEE security is the update process. An attacker may insert malicious logic to TEE images to be updated, thus the integrity of the images must be checked before applying. Downgrade attack is performed to rollback TEE to some earlier version that is known to contain certain vulnerabilities for further exploitation. Thus, it should be ensured that

[1] https://www.widevine.com

TABLE 1. Available Research Platforms for TrustZone

Hardware Platform	Hardware Features	Documentation	Open source TEE supported	Remarks
Samsung Exynos4412	Cortex-A9	Sufficient (NDA needed)	N.A.	Typical 32bit ARM platform for mobile device.
Freescale i.MX6/i.MX53	Cortex-A9	Sufficient	OP-TEE	Typical 32bit ARM platform for IoT device, such as an automobile.
HiKey 960	Cortex-A73/A53	Sufficient	OP-TEE	Typical 64bit ARM platform with hardware virtualization support.
Arm Juno Versatile Express	Cortex-A53/A72	Sufficient	OP-TEE/Trusty	Typical 64bit ARM platform with hardware virtualization support provided by ARM.
Raspberry Pi2	Cortex-A7	Lack of enough feature description	OP-TEE	Typical 32bit ARM platform for IoT device.

the update process of TEE is monotonic.

Besides TEE vulnerabilities, side channel and physical attack can also be serious threats to the secret data in the secure world, like ARMageddon [21] and cold-boot attack (e.g., Frozen) [22]. Most of the research mentioned do not consider such attacks. Sentry [23] and CaSE [24] propose using SoC storage, e.g., internal SRAM or L2 cache, instead of DRAM to store secret data like private keys to defend against physical attacks. To host data larger than the capacity of SoC storage, these systems use encrypted swapping to swap encrypted data out to/in from untrusted DRAM.

Ecosystem: Openness VS. Security

Currently, the ecosystem of TEE is still not open, which means that all the TAs are controlled by the vendors and are pre-installed on devices before selling. In order to enable REE applications to better leverage TrustZone, it should be allowed to install TAs dynamically. Some TEE products already offer TA management as a service using cloud, e.g., MyTAM [25], which enables an application to request the phone to download a customized TA and deploy it in the TEE. The phone will verify the signature of the downloaded TA before installation. However, it also brings new security challenges to the TEE since now the attacking surface of TEE is larger.

Remote attestation is also an important feature of TEE. A mobile phone can be remotely attested to determine whether it has deployed a valid TEE. The phone manufacturers will embed a private key in each TEE before selling the devices. The public keys are usually managed by the vendors. Recently, ARM has established Open Trust Protocol (OTrP) alliance [26], which is building a certificate authority-based trust architecture to further ease the process of attestation of trusted applications.

For every TEE-equipped device, a per-device key pair and a unique device ID will be generated and securely saved in the secure storage of the device. It is stored either in the efuse or the eMMC partition with RPMB (Replay-Protected Memory Block) in a controlled environment (e.g., on the production line of device factory) before shipping to the market. The public key of the device and the device ID are uploaded to the backend server in the production line for future device attestation. The efuse storage is relatively expensive and the available space is usually very small. The RPMB partition in the eMMC, on the other hand, contains several megabytes for storing data in an authenticated and replay-protected manner.

PLATFORMS FOR RESEARCHING

Currently, although almost all of ARM application processors support TrustZone, and billions of mobile phones on the market have deployed TEE, most of them are not available for research, either due to lack of documentation and firmware support, or due to being close with secure boot. Still, there are several hardware and software platforms available for research. Hardware platforms that could be easily used by researchers include Samsung Exynos 4412, Freescale i.MX6/i.MX53, HiKey 960, ARM Juno Versatile Express development board and Raspberry Pi 2. Open source TEE, which is still actively maintained, includes Linaro OP-TEE and Google Trusty. Note that although TrustZone architecture is proposed by ARM, many SoCs provide customized protection mechanisms on memory and peripheral. Samsung Exynos 4412, Freescale i.MX6/i.MX53 and Raspberry Pi 2 are suitable for IoT and low-end mobile device. ARM Juno Versatile Express development board is recommended by ARM targeting at ARMv8 software prototyping with a flexible hardware expansion. Both HiKey 960 and ARM Juno Versatile Express development board are good alternatives for TrustZone and hardware virtualization research.

CONCLUSION

As ARM's ecosystem is expanding from mobile devices to IoT, vehicles, and servers, security is becoming indispensable in more and more scenarios. TrustZone and TEE provide a key-enabling technology for protecting, monitoring and isolating various up-layer applications, which will play a more important role on multiple platforms in the near future. There is a trend that the code running in the secure world is increasing. Fine-grained isolation within the TEE will be essential. It is also critical to enforce the correctness of TEE's design as well as its implementation by technologies like formal verification. Meanwhile, the management and deployment of TA should also be much more strict. ■

Wenhao Li is a PhD candidate at Shanghai Jiao Tong University. He received his master's degree from the same university. His research interests include system and mobile security. His research focuses on leveraging new hardware features to improve the dependability and security of systems, building and optimizing high performance platforms that benefit from system security.

Yubin Xia is an associate professor at Shanghai Jiao Tong University. His research interests include operating systems, system virtualization, and computer architecture. Currently, he is focusing on building a new system software suitable for isolation in new execution environments, like TEE, VM, enclave, etc.

Haibo Chen has been a full professor at Shanghai Jiao Tong University since 2011. He received a bachelor's degree and a PhD in Computer Science, both from Fudan University. He currently leads the Institute of Parallel and Distributed Systems, and works with members to improve the performance and dependability of computer systems.

REFERENCES

[1] Tiago Alves and Don Felton. 2004. *TrustZone: Integrated hardware and software security.* ARM white paper 3, 4 (2004), 18–24.

[2] Nuno Santos, Himanshu Raj, Stefan Saroiu, and Alec Wolman. 2014. "Using ARM TrustZone to build a trusted language runtime for mobile applications." In *ASPLOS*. ACM, 67–80.

[3] Konstantin Rubinov, Lucia Rosculete, Tulika Mitra, and Abhik Roy Choudhury. "Automated partitioning of android applications for trusted execution environments." In ICSE. 2016.

[4] Guan, Le, et al. "TrustShadow: Secure execution of unmodified applications with ARM trustzone." *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 2017.

[5] Li, Wenhao, et al. "Building trusted path on untrusted device drivers for mobile devices." *Proceedings of 5th Asia-Pacific Workshop on Systems*. ACM, 2014.

[6] D. Liu and L. P. Cox, "Veriui: Attested login for mobile devices," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014.

[7] Kailiang Ying, et at. TruZ-Droid: "Integrating TrustZone with Mobile Operating System." MobiSys, 2018.

[8] A. Amiri Sani, "Schrodintext: Strong protection of sensitive textual content of mobile applications," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 2017, pp. 197–210.

[9] W. Li, H. Li, H. Chen, and Y. Xia, "Adattester: Secure online mobile advertisement attestation using trustzone," in MobiSys, 2015.

[10] Wenhao Li, Shiyu Luo, Zhichuang Sun, Yubin Xia, Long Lu, Haibo Chen, Binyu Zang, Haibing Guan. "VButton: Practical Attestation of User-driven Operations in Mobile Apps." MobiSys, 2018.

[11] Xinyang Ge, Hayawardh Vijayakumar, and Trent Jaeger. Sprobes: "Enforcing kernel code integrity on the trustzone architecture." MOST, 2014.

[12] Ahmed M Azab, Peng Ning, Jitesh Shah, Quan Chen, Rohan Bhutkar, Guruprasad Ganesh, Jia Ma, and Wenbo Shen. "Hypervision across Worlds: Real-time kernel protection from the ARM TrustZone Secure World." CCS, 2012.

[13] Jang, Jinsoo, et al. PrivateZone: Providing a Private Execution Environment using ARM trustzone." *IEEE Transactions on Dependable and Secure Computing*, 2016.

[14] Brasser, Ferdinand, et al. "Regulating arm trustzone devices in restricted spaces." *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 2016.

[15] Liu, He, et al. "Software abstractions for trusted sensors." *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012.

[16] Raj, Himanshu, et al. fTPM: "A software-only implementation of a TPM chip." USENIX Security Symposium. 2016.

[17] S. Mirzamohammadi, J. A. Chen, A. Amiri Sani, S. Mehrotra, and G. Tsudik, "Ditio: Trustworthy auditing of sensor activities in mobile & IoT devices." In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 2017.

[18] Liu, Renju, and Mani Srivastava. "PROTC: PROTeCting drone's peripherals through ARM trustzone." *Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*. ACM, 2017.

[19] Zhichao Hua, Jinyu Gu, Yubin Xia, Haibo Chen, Binyu Zang and Haibing Guan. vTZ: "Virtualizing ARM trustzone." Usenix Security Symposium, 2017.

[20] Aravind Machiry, Eric Gustafson, Chad Spensky, Chris Salls, Nick Stephens, Ruoyu Wang, Antonio Bianchi, Yung Ryn Choe, Christopher Kruegel, and Giovanni Vigna. "BOOMERANG: Exploiting the semantic gap in trusted execution environments." NDSS, 2017.

[21] Lipp, Moritz, et al. "ARMageddon: Cache attacks on mobile devices." USENIX Security Symposium, 2016.

[22] Halderman, J. Alex, et al. "Lest we remember: cold-boot attacks on encryption keys." *Communications of the ACM*, 2009.

[23] Colp, Patrick, et al. "Protecting data on smartphones and tablets from memory attacks." *ASPLOS*, 2015.

[24] Zhang, Ning, et al. "Case: Cache-assisted secure execution on arm processors." S*ecurity and Privacy (SP)*, 2016.

[25] Intercede Inc. MyTAM, https://www.intercede.com/news/intercedes-mytam-enables-enhanced-trust-for-android-apps-to-protect-against-hackers-and-malware.

[26] ARM Inc. OTrP, https://www.electronicsweekly.com/news/arm-aims-to-build-trust-in-iot-security-2016-07.