Proceedings of the 2nd International conference on Electronics, Communication and Aerospace Technology (ICECA 2018)
IEEE Conference Record # 42487; IEEE Xplore ISBN:978-1-5386-0965-1

Secure boot of Embedded Applications – A Review

Rashmi R V
M.Tech Student ,SENSE
VIT University, Vellore

Karthikeyan A
Assistant Professor (Senior), SENSE
VIT University, Vellore

***Abstract* :- Trusted Computing focusses on the security of systems by using hardware-software combination to guarantee security. Although there are various techniques to enhance security, new technological advancements bring in new ways for attackers also to inject their malicious software and infections. The modern world uses all smart devices and systems which includes a network with embedded Operating System, sensors and so on. So security becomes a serious issue not only to system Operating System in personal computers but to all these smart devices. Secure boot is a technique which ensures the only the right Operating System gets booted. This paper analyses the process of secure boot done in each of the applications to protect OS kernel from attackers.**

Keywords: Trusted computing, secure boot, embedded Operating System.

I.INTRODUCTION:

In today's world, electronic devices are becoming smart right from refrigerators to uranium centrifuge control systems. Hence security now is not only bounded to PC but also to wide range of embedded smart connected devices. When the security of a device is at threat we can no longer rely on the device for secure data exchange or storage. If electronic banking transactions, implantable medical devices or critical systems such as nuclear plants are hacked, then the global trust would be impacted drastically. Having a complete understanding of security threat and its rootcause is an essential for security engineering of embedded systems. Security examination and configuration must consider the complete scope of the risk angles in order to recognize security necessities, enhance and apply security controls within the confinement of limitations. Consequently in the vision of security of embedded systems, knowing the real reason for the attack and their vulnerabilities the learning causes us to enhance the security of any electronic systems.

Secure boot is a process where the OS boot images and code that are authenticated against the hardware before they are allowed to be used in the boot process. The hardware is set up in such a way that it only authenticates the code generated using trusted security credentials. In short, it ensures the booting of intended OS and it is not tampered with by malicious third parties or malware.

To overcome various security threats faces by smart connected electronic system this paper presents different types of secure boot being done in different applications.

II. SECURE BOOT PATTERNS:

Two ideas of Trusted computing is examined by sHans Lohr et al in [1].They are secure boot and secure storage. Trusted computing conveys a blend of programming and equipment security components to tackle security issues that isn't possible to be comprehended by just programming. At whatever point there is violation of security properties in the product that will be booted, secure boot keeps it from booting subsequently ensuring OS. On the off chance that it is just detected, at that point it is called authenticated boot. Secure storage is a vital prerequisite. Straightforward encryption isn't adequate to ensure information: additionally the aggressors ought not have the capacity to acquire the decoding key. Secure storage handles this issue by utilizing equipment to impose confinements on the put away information. Before get to is allowed to an application, the software integrity is checked.

[1] analyses an example problem and proposes solution of the same. Users who deals with applications that are highly sensitive to security has to make sure of their operational integrity. Any unauthorized changes to the the operating system or application code or may lead to unintentional response that violates security objectives.

Despite the fact that clients have trust on the hardware, they have to ensure that the software which gets stacked on this hardware has not been perveted. Commonly, the hardware begins by stacking a bit of firmware code, which thus stacks the bootloader from a pre-characterized put from main memory. The bootloader loads the operating

system kernel, which thusly stacks the system benefits, the device drivers, and different system level applications.

At any phase of booting procedure, the parts of software can be altered or traded by another client or by any vindictive software. The arrangement proposed is by utilizing roots of trust and measurement. The integrity checks for bring down layer boot modules is done first and then control is being exchanged to the succeeding stage just if the integrity of the specific stage is substantial. Henceforth, each stage represents the duty regarding checking the integrity of the following stage.[9] Checking of integrity can be performed in two courses: correlation of hash values or confirmation of advanced signatures. In the principal case, the hash values are at first figured utilizing a cryptographic hash function and then looked at against reference values.

On the off chance that the processed value and the reference value doesnot coordinate, at that point it appears the binary image has been altered. In the second case, the program binary image is marked with a signature key cryptographically that is just known only to the seller. The signature of the binary can be confirmed to check whether it has been changed since the time of its signature. [2]

In the event that integrity violation gets distinguished at one phase, the execution will be ceased at that stage and the system stops. The succession of all these integrity checks makes the chain of trust. The sequence of boot pattern is as shown in Fig.1.

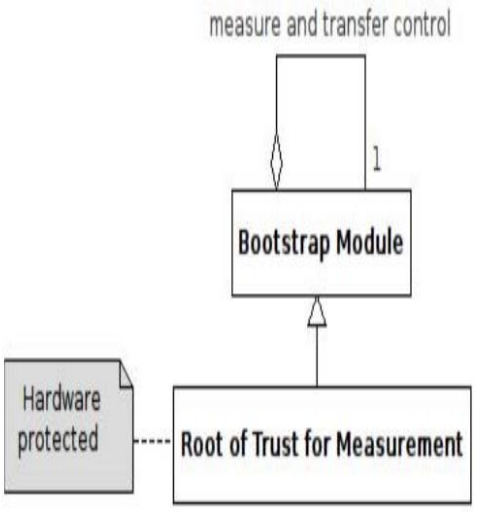

Fig.1 Sequence of Boot pattern

Authenticated boot refers to the mechanism which provides a chance to verify the features of the software which has been booted**.** The main module in the bootstrap chain ie. the hardware is trusted as a matter of course and it gauges the integrity of the following module. The measurements are then stored in protected hardware register and later reported by the security module. The main module in the bootstrap chain ie. the hardware is trusted as a matter of course and it gauges the integrity of the following module.

III. SECURE BOOT OF EMBEDDED LINUX:

Embedded systems is getting flourished in all areas. The world is becoming smart because of advancements in embedded technology. It includes to incorporate an embedded OS (Operating System) inside every system.

In many FPGAs, the Operating System is secured into an external Flash memory and is being run on a processor which is implanted into the FPGA. Because of the remote feature highlight, they can be exposed to any hostile condition. The remote refresh additionally appears to be unavoidable. FPGA embedded processor may now and again process the OS refresh through an unreliable system. Be that as it may, these highlights may offer ascent to security blemishes influencing the system integrity.[5] Integrity can be changed by smind the end goal to infuse malicious code. Correspondingly integrity can be bothered by

using an old configuration in order to downgrade the system

.It is important to secure the processor kernel boot on FPGA. Because of the extensive storage capacity necessity the kernel is put away in an externa Non-Volatile Memory. By and large, this memory is offchip, with the goal that an assailant can adjust the kernel keeping in mind the end goal to present malevolent code.[4] Subsequently to secure the booting process on FPGA, it is required to dynamically trust the distinctive parts amid the system start-up, therefore building up a trusted boot chain from bitstream-to-kernel.

[3] proposed a risk demonstrate. The suspicion here is that FPGA system is presented to an adverse condition where physical however non-intrusive attacks, beside side channel one are feasible. FPGA chip is treated as a trusted zone.

It considers a wide range of attacks including "man in the center attacks", that fundamentally incorporates:

Spoofing, in which the attacker replaces the data or a portion of the data with his own data. Replay, in which the intruder records a particular information and replays it at later time. Of these replays are especially dangerous, on the grounds that the present arrangements proposed by sellers of FPGA to guarantee bitstream encryption and integrity are not adequate to protect against replay attacks. The proposed arrangement is to secure booting of installed Linux on FPGA, in this way keeping an opponent to run is claim possibly malevolent program that is put away in external memory. Fig.2 demonstrates the boot mechanism in XiLinx

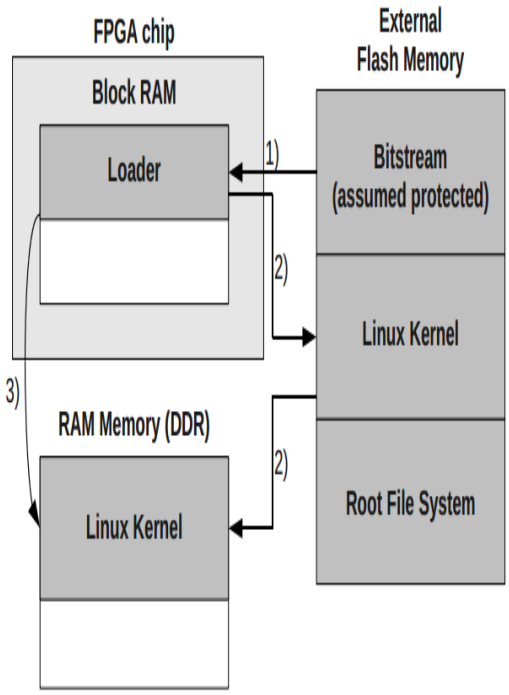

Fig. 2 Boot process in Xilinx

Initially, the bitstream is duplicated from external flash memory to FPGA and began. Likewise RAM of inner FPGA blocks are instated with 'bootloader'. It duplicates the kernel from the external flash to a huge measured RAM memory and afterward bounces to the proper RAM area. Along these lines the kernel begins. This procedure is unsecure in light of the fact that it does exclude integrity verification. The attacker can supplant the kernel by his own particular keeping in mind the end goal to execute his own conceivably malevolent code.

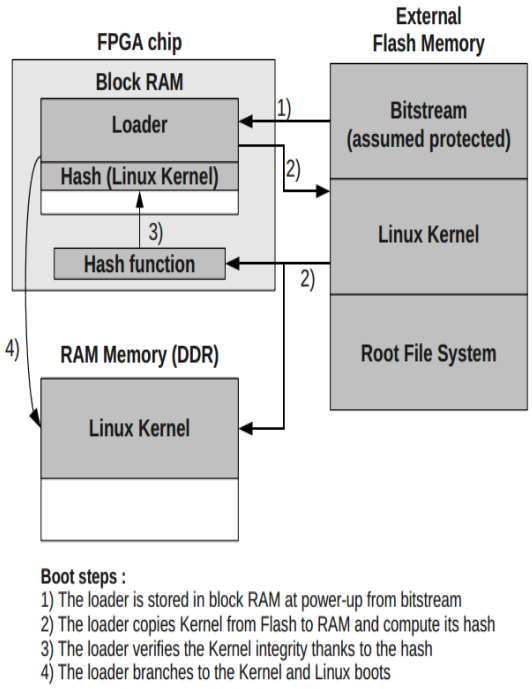

Fig.3 Boot process with integrity verification

Figure 3 demonstrates the boot procedure with integrity verification. The distinction with the typical boot is in the incorporation of integrity verification of the kernel that is replicated. The hash value of the kernel is put away in the boot loader. The hash value calculation will be executed the FPGA chip, which is the confided in region. In this procedure, the kernel is replicated to RAM. The hash value of the kernel produced by hash center is available in the RAM. Presently, the bootloader will hop to the kernel location in the RAM just if the subsequent hash value of the kernel is equivalent to that stored in the bootloader.

The adaptability change is likewise engaged in the following stage which permits to change the kernel, in the external memory, with no adjustment in the bitstream. Keeping in mind the end goal to abstain from putting away a hash value particular to the kernel in block RAM, asymmetric cryptography is utilized.

In this plan, the kernel supplier keeps the private piece of the asymmetric pair of keys in order to sign the diverse future versions of kernel. The public key is put away in the bootloader rather than the hash value to check the signature of the kernel.

There are two stages to boot:

• First, the hash value is created by hash Core

• Then, the mark of the hash value that is stored in the external Flash memory is confirmed by the bootloader with the already created hash value and its public key.

With this concept, the public key that is kept in the blocks RAM will be similar even after updation of the kernel. Consequently, the blocks RAM need not be reinitialized and hence the bitstream need not be changed and reloaded each time. Thus flexibility is increased.

IV. SECURE BOOT ON FPGA BASED IOT DEVICE:

[6] depicts the investigation of secure boot in FPGA based IOT system.

Yuan Liu, Jed Briones, Ruolin Zhou, and Neeraj Magotra [6] has expressed that Intel has assessed that the quantity of connected devices worldwide will ascend from 15 billion of every 2016 to 200 billion by 2020. An assortment of applications in wellbeing, consumer, and military applications are accessible for IoT . Thus it is critical to viably and productively secure and ensure IoT devices. In this paper, Xilinx Zynq-7000 Series System-on-Chip (SoC) ZC706 prototype board to design an IoT gadget was employed. They have contemplated Zynq-ZC706 to

(1) encrypt FPGA bitstream to shield the IoT gadget from bitstream decoding;

(2) encrypt system boot image to upgrade system security; and

(3) guarantee the FPGA works effectively as expected through authentication to abstain from spoofing and Trojan Horse attackand have examined.

Xilinx Zynq-7000 Series System-on-Chip (SoC) ZC-706 prototype board to plan an IoT gadget .

outline,. They have contemplated Zynq-ZC706 and possessed the capacity to encrypt FPGA bitstream to shield the IoT gadget from bitstream decoding to safeguard against dangers to FPGA. [7].This major examination is extremely valuable with future extent of securing ZC706 based IoT device through physical unclonable function (PUF) key encryption, authentication, and reverse boot, and so on. Once the system is booted securely, the device IP address can be retrieved as shown in Fig. 4 through the SDK terminal. Both the device and a computer are connected to the same local network. After setting the IP address, the computer can ping the FPGA device.

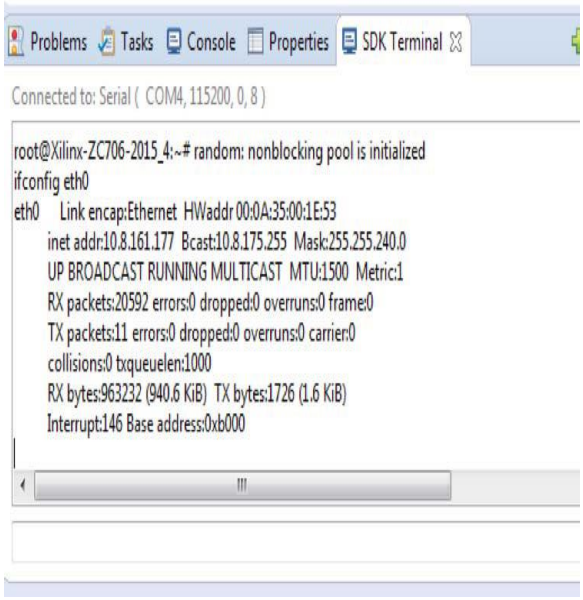

Fig.4 IP Address of the Device

V. SECURE BOOT BASED ON MTM:

[8] investigations the transitive trust of TPM and the secure boot of MTM are examined. In view of that secure boot conspire for embedded system is discussed. In view of embedded Linux and U-Boot, the secure boot capacities for inserted system are executed by

(1) estimation of integrity

(2) storage of integrity

(3) reporting of integrity

This part can improve the security of embedded framework by executing MTM. Also the execution of secure boot is attempted.

One of the security includes in trust registering is Transitive trust. MTM (Mobile Trusted Module) determination characterizes loads of trust roots. Of all, RTM (Root of Trust for Measurement) and RTV .

(Roots of Trust for Verification) are put away as read-only programming modules in ROM. These are executed first when controlled on. These are utilized to start MTM and go about as the beginning stage for trust estimation and check. Two trusted modules characterized by MTM are MRTM (Mobile Remote-Owner Trusted Module) and MLTM (Mobile Local-Owner Trusted Module). MRTM is implied for implanted stages.

Fig.5 demonstrates the procedure of transitive trust in view of MRTM.

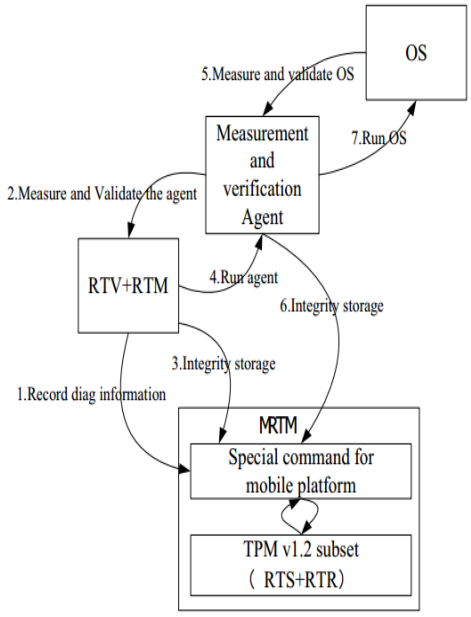

Fig.5 Transitive trust based on MRTM

The mechanism basically includes four entities:

(1) RTM and RTV

(2) The bootloader U-Boot

(3)The Linux kernel image and

(4)The root File System image.

After estimation, storage and confirmation of the Linux kernel image and root FS image, the Linux part is to be booted and the root FS is to be mounted. In this paper, component of boot appropriate for embedded system in view of MTM particular is proposed. The procedure is appeared in Fig.6.

Further, tests have been performed to demonstrate that the plan meets the protected boot prerequisites, the security check for kernel image and root FS image can distinguish the diverse kinds of attacks.

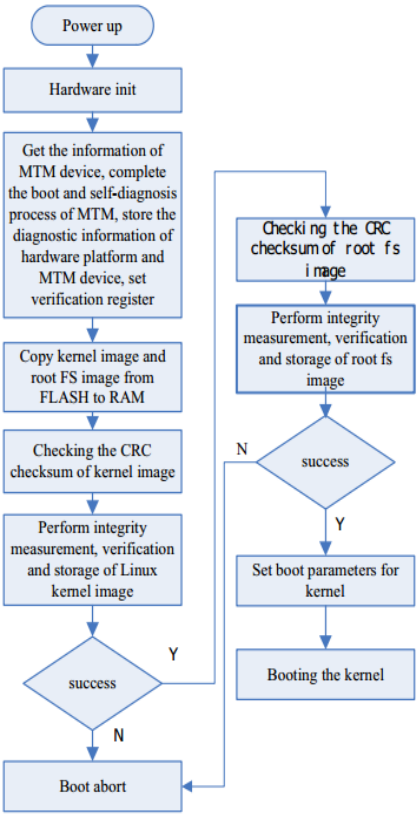

Fig.6 Secure boot flow of trusted embedded system

VI. SECURE BOOT OF WIRESLESS SENSOR NETWORK:

The Wireless Sensor Node fills in as one of the basic parts required for different present day wireless applications and administrations condition. As Wireless Sensor Networks (WSN) turns out to be more mainstream because of their part in transportation, keen home frameworks, military and so on, Security issues develop in light of the fact that these platforms are by and large presented to an excessive number of potential dangers. Regular attacks in WSN framework is amid the underlying boot period of the gadget.[10]. Not just the software based security is adequate in WSN, they likewise require physical security in sensor nodes.[12]proposes a plan to improve the security includes in the sensor node by using ARM11 32 bit processor with on-chip SOC memory and TrustZone.

Fig. 7 demonstrates the ARM TrustZone secure boot conspire. This plan includes cryptographic verification to each phase of the secure boot process. This procedure intends to state the integrity of each image that are executed once it is powered on.

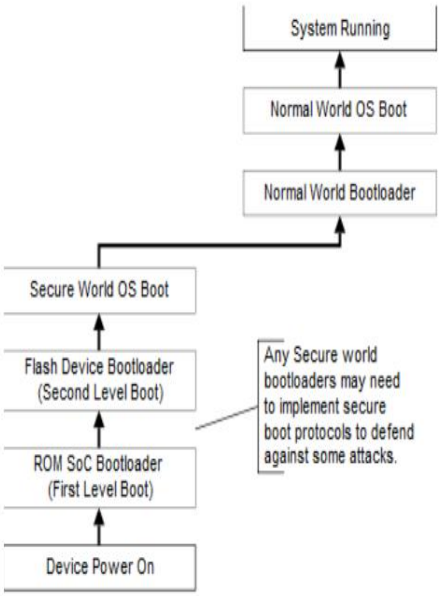

Fig.7 Trust zone Secure Boot

The proposed solution consists of wireless sensor node that focusses on the security features only. The design has main impact on the location of credential informations such as system images and security keys. Based on the above requirement, a design is proposed that consists of wireless sensor networks. It comprises of an embedded processor with TrustZone technology and on-SoC region.

SAMSUNG's S3C6410 is picked as the installed processor of the embedded system to help equipment security and high handling power.

An essential component of S3C6410 is that it gives the equipment security foundation which is only the TrustZone innovation. The innovation means to give a security structure that secures the boot piece of code of the framework from every physical assault since this firmware is the foundation of trust for the framework.

At first booting up from "Boot Rom" location takes place. After the gadget is powered on, the center processor will be booted from secure ROM to enter to secure state since the firmware will instate TZMA to ensure on-chip memory.[11] This procedure will verify the integrity of the secondary bootloader dwelling on outer glimmer memory by estimating the hash estimation of the picture. Prior to the control is given to the secondary bootloader, these estimations are looked at against the reference esteem present in the protected EPROM. After the checking of integrity and verification of the image, the control will be given to the second bootloader. Second bootloader will introduce the fundamental configuration and empower TZASC to divide the protected and non protected region of primary memory. This method expects to check the integrity of the equipment and programming and its confidentiality to keep any malignant code or changed programming from being stacked. The proposed concept and its block diagrams is as shown in Fig.8 and Fig.9.

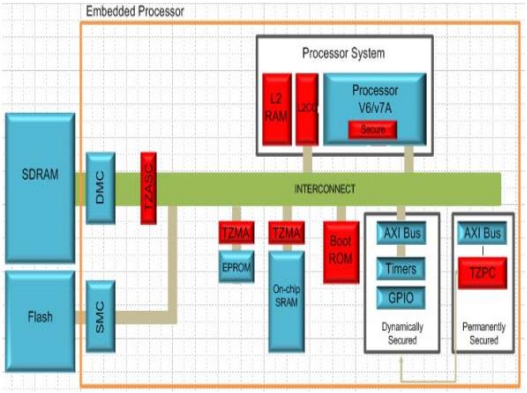

Fig.8 Block Diagram

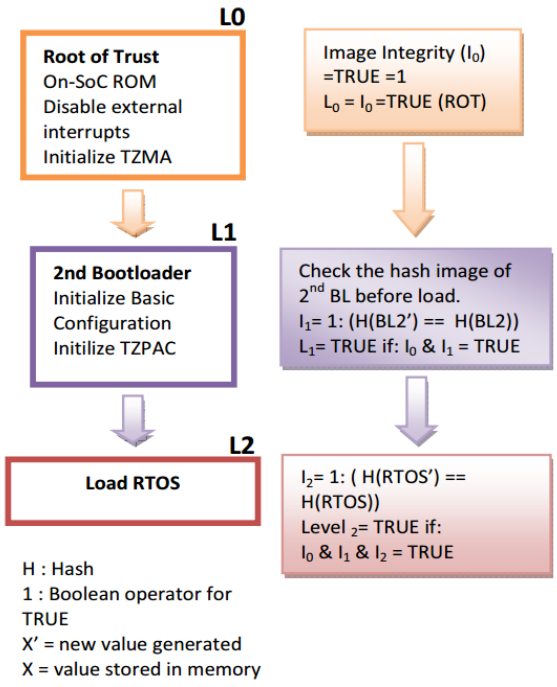

Fig. 9 Secure Boot in WSN

VII. CONCLUSION:

The secure boot which prevents malicious software on different applications is analysed in detail. Any design methodology nowadays requires the security aspects to be taken into consideration for reliability and stability. So the analysis forms the basis for any secured embedded system design.

REFERENCES:

[1] Hans Löhr, Ahmad-Reza Sadeghi, Marcel Winandy, "Patterns for Secure Boot and Secure Storage in Computer Systems" , International Conference on Availability, Reliability and Security, 2010.

[2] Trusted Computing Group (TCG), TPM Main Part 1.2.3 Design Principles Specification Version 1.2, 2011.

[3] Florian Devic, Lionel Torres, Benoît Badrignans,"Securing Boot of an Embedded Linux on FPGA", IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, 2011.

[4] F. Devic, B. Bradrignans, L. Torres, "Secure FPGA Configuration Architecture Preventing System Downgrade", FPL'10: IEEE International Conference on Field Programmable Logic and Applications, pp. 179-182, September 2010.

[5] Nicolas Sklavos, Ioannis D. Zaharakis, Achilles Kameas, Angeliki Kalapodi, "Security & Trusted Devices in the Context of Internet of Things (IoT)", Euromicro Conference on Digital System Design, 2017.

[6] Yuan Liu, Jed Briones, Ruolin Zhou, and Neeraj Magotra, "Study of Secure Boot with a FPGA-based IoT Device", 60th International Midwest Symposium on Circuits and Systems (MWSCAS), 2017.

[7] L. Sanders, "Secure boot of zynq-7000 all programmable soc,"April 2015, v2.0.

[8] Kurt Dietrich, Johannes Winter,"Secure Boot Revisited", The 9th International Conference for Young Computer Scientists,2008.

[9] D. Schutz, "Boot loader," Proceedings of the 11th European Conference on Pattern Languages of Programs (EuroPLoP 2006), 2006.

[10] L.H Adnan, Y.M. Yussoff, H. Hashim, "Secure Boot Process for Wireless Sensor Node", International Conference on Computer Applications and Industrial Electronics (ICCAIE 2010), December 5-7, 2010.

[11] ARM Ltd. TrustZone Technology Overview. Introduction available at "http://www.arm.com/products/processors/technologies/trustzone.php."

[12] J. Winter, "Trusted computing building blocks for embedded linux-based ARM trustzone platforms," in Proceedings of the ACM Conference on Computer and Communications Security,2008, pp. 21-30.