Study of Secure Boot with a FPGA-based IoT Device

Yuan Liu, Jed Briones, Ruolin Zhou, and Neeraj Magotra
Department of Electrical and Computer Engineering
Western New England University

***Abstract*—Internet of Things (IoT) is network connected "Things" such as vehicles, buildings, embedded systems, sensors, as well as people. IoT enables these objects to collect and exchange data of interest to complete various tasks including patient health monitoring, environmental monitoring, system condition prognostics and prediction, smart grid, smart buildings, smart cities, and do on. Due to the large scale of and the limited host processor computation power in an IoT system, effective security provisioning is shifting from software-based security implementation to hardware-based security implementation in terms of efficiency and effectiveness. Moreover, FPGA can take over the work of infrastructure components to preserve and protect critical components and minimize the negative impacts on these components. In this paper, we employ Xilinx Zynq-7000 Series System-on-Chip (SoC) ZC706 prototype board to design an IoT device. To defend against threats to FPGA design, we have studied Zynq-ZC706 to (1) encrypt FPGA bitstream to protect the IoT device from bitstream decoding; (2) encrypt system boot image to enhance system security; and (3) ensure the FPGA operates correctly as intended via authentication to avoid spoofing and Trojan Horse attacks.**

I. Introduction

Internet of Things (IoT) is network connected "Things" such as vehicles, buildings, embedded systems, sensors, as well as people. IoT enables these objects to collect and exchange data of interest to complete various tasks including patient health monitoring, environmental monitoring, system condition prognostics and prediction, smart grid, smart buildings, smart cities, and do on. Specifically, IoT devices allow physical objects to share data and be controlled remotely across existing network infrastructures, resulting in improving efficiency, accuracy, and economic benefit. Intel has estimated that the number of connected devices worldwide will rise from 15 billion in 2016 to 200 billion by 2020. IoT has a variety of applications in health, consumer, and military applications. Therefore, it is significant to effectively and efficiently secure and protect IoT devices so that they can be adopted dramatically toward a connected future.

Due to the large scale of and the limited host processor computation power in an IoT system, effective security provisioning is shifting from software-based security implementation to hardware-based security implementation in terms of efficiency and effectiveness, illustrated in Fig. 1. Specifically, pure software-based applications are not practical in IoT with higher data rates. For example, Bro in [1] and WebSTAT in [2] cannot handle data rates higher than 100M bits per second (Mbps). Moreover, increased network bandwidth can make all off-the-shelf processors overburdened and hard to maintain high throughput. Using dedicated FPGAs (Field Programmable Gate Array) can relieve the burden on processors. Meanwhile, the FPGA can take over the work of infrastructure components to preserve and protect critical components and minimize the negative impacts on these components. Thus, we employ Xilinx Zynq-7000 System-on-Chip (SoC) ZC706 FPGA prototype board to design an IoT device.

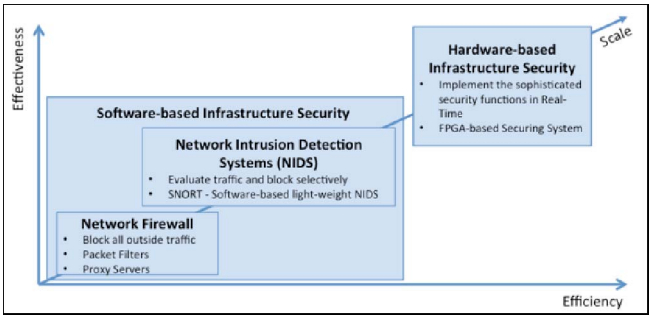

Fig. 1. Trend of security implementation

However, the main challenges of designing an FPGA-based IoT device are to defend against threats and attacks to an FPGA design. These threats and attacks include: (1) bitstream decoding where the attackers can recover a bitstream or original netlist by decoding the bitstreams; (2) spoofing where all or part of an FPGA bitstream is replaced or modified; and (3) Trojan Horse where the system is accessed by attackers who insert their own logic functions into the design to access data stored in FPGA, obtain insight of the overall design, or hijack the system.

In this paper, to defend against such threats and protect intellectual property (IP), we have studied Zynq-ZC706 to (1) encrypt FPGA bitstream to protect the IoT device from bitstream decoding and spoofing; (2) encrypt system boot image to enhance system security; and (3) ensure the FPGA operates correctly as intended via authentication to avoid spoofing and Trojan Horse attacks.

The rest of the paper is organized as follows: Section II reviews Xilinx Zynq-7000 SoC FPGA platform; Section III presents the implementation of secure boot of IoT device; Section IV concludes the paper and discusses future work.

II. Overview of Zynq-7000 SoC FPGA

The Zynq-7000 All Programmable SoC ZC706 Evaluation Kit is equipped with dual-core ARM cortex A9 processors, DDR3, Ethernet PHY, general purpose I/O, PCI Express interface, UART interface and 28nm programmable logic (PL). The

978-1-5090-6389-5/17/$31.00 ©2017 IEEE

Zynq-7000 architecture can implement custom logic in the PL and custom software in the processing system (PS). It allows the realization of unique system functions. Fig. 2 illustrates the block diagram of ZC706 board.

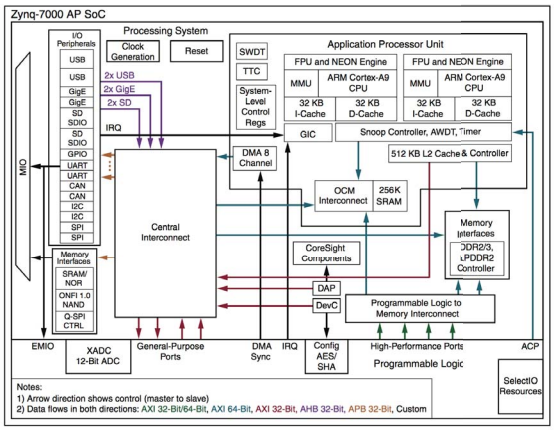

Fig. 2. XC7Z045 Block Diagram [3]

The ZC706 evaluation board employs XC7Z045 AP SoC which consists of a PS (integrated processing system) block and PL (programmable logic) block. There are four major blocks that comprise the PS which are Application processor unit (APU), Memory interface, I/O peripherals (IOP), and interconnection. The XC7Z045 AP SoC enables both secure boot and non-secure boot processes. The PL-side I/O banks implement the DDR3 interface. And the DDR3 component memory is connected to XC7Z045 AP SoC PS. Bitstream can be downloaded to PL using JTAG connection from host computer. Meanwhile, JTAG connector allows the software debugger and Vivado serial I/O analyzer to access the SoC. Marvell Alaska PHY device is used on the ZC706 evaluation board which can offer different Ethernet communications such as 100Mb/s and 1000Mb/s. The PHY interface connects user-provided Ethernet by Halo HFJ11-1G01E RJ-45 connector. Once the board is reset or power-on, the PHY is configured to operate in RGII mode. The USB-to-UART bridge is also included in ZC706 evaluation board which allows a connection to a host computer with a USB port. The memory interface unit, APU and the IOP are all connected to the PL through a multilayered ARM AMBA AXI interface. The interconnection supports multiple simultaneous master-slave transactions which is non-blocking. It is designed with ARM CPU which is latency sensitive masters. The interconnect also has shortest paths to memory and bandwidth critical masters like potential PL masters and also has high throughput connections to the slaves with which they need to communicate. The Quality of Service (QoS) block in the interconnect can regulate the traffic which is generated by the CPU, DMA controller and a combined entity representing the master in IOP. Another important part is PS-PL interface. The AXI provides access between PS and PL. High performance AXI ports and ACP interface are used to transfer data and can be managed under software control if required [4].

In this paper, ZC706 is employed to design an IoT device since FPGA can interface with the outside world easily with low power, low latency, high determinism with re-programmable and re-configurable capabilities.

III. Implementation and Demonstration of Secure Boot IoT Device

The hardware components used in secure boot process are NVM (non-volatile memory), OCM (on chip memory), BootROM and secure storage as shown in Fig. 3.

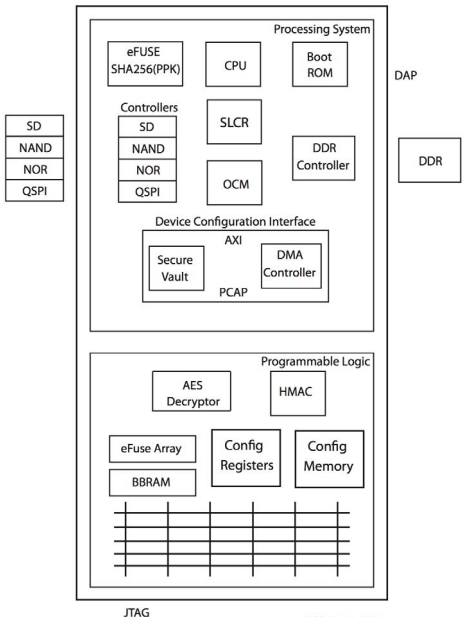

Fig. 3. Hardware Components Used in Secure Boot [8]

A. Create a Secure Boot Image

Typically, there are four steps to create a secure boot image: (1) generate AES key using Xilinx Software Develop Kit (SDK) Bootgen tool or an external tool; (2) program AES key by Secure Key Driver in Xilinx Vivado Design Suite; (3) use SDK Bootgen to create an image where both AES encryption and RSA authentication can be enabled; and (4) load the boot image, a BIN file, onto a SD card. The process is shown in Fig. 4. Meanwhile, enabling RSA authentication can also avoid spoofing or Trojan Horse attacks to increase system security.

The OpenSSL is used to generate RSA key. There are four different keys that can be used in RSA authentication: PPK (Primary Public Key), PSK (Primary Secret Key), SPK (Secondary Secret Key) and SSK (Secondary Public Key). The primary and secondary keys are used in RSA authentication. The primary keys authenticate the secondary keys and the partitions can be authenticated by secondary keys. In RSA authentication, the public key can be obtained from the private key which means we can extract the public key from the private key. The hash of the PPK is stored in the PS eFUSE array. Using eFUSEs for the hash of the PPK is an efficient use of the silicon.

In addition, Petal-linux is used to customize the Linux system on ZC706 processing systems. We create the PetalLinux project on Linux OS based on ZC706 BSP (Board Support Package) which collects all of the basic libraries and drivers. In

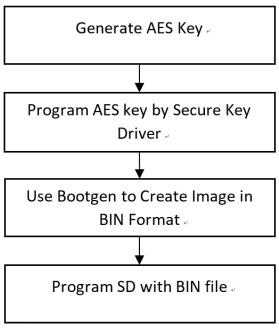

Fig. 4. Secure System Developing Step

the configure window, we can decide which Linux component is selected and the output of the project. The PetalLinux will generate a Boot file and an image file. The U-Boot Boot loader is also important when we boot the Linux system. It includes functions such as erasing, reading, writing NVM and reading DDR memory. We choose to boot from SD mode where we save the boot image to a SD card.

B. Demonstration of Boot Process

PS on Zynq-ZC706 is the master of the boot and configuration process. To boot the device securely, PL must be powered on to enable decryption and authentication algorithms that reside within the PL. The ARM Cortex-A9 core contains two processing unit which are CPU0 (used for boot) and CPU1.

Specifically, secure boot process starts from loading the FSBL (first stage boot loader) into OCM (On Chip memory) by BootRom code. The BootROM code, which is a masked programmable ROM code, is running at power-up. Then, the FSBL, which is now in OCM, will copy different partitions from NVM to DDR (Dual Data Rate Random Access Memory) and PL configuration memory (the destination of the bitstream).

If RSA authentication is used, the partition will load to RSA verification to test the spoof. If the RSA authentication is not used, it will skip this step and directly go to the next step. Because of using encryption and authentication, the partition is loaded to the AES engine for decryption and authentication before loaded to the destination address. The information inside FSBL includes the status of the encryption, location of the key and some characteristics of the FSBL. ZC706 board provides the encryption key storage via backup battery circuit. The Seiko TS518FE rechargeable 1.5V lithium button-type battery B2 is soldered to the board. The positive output is connected to the XCZ7045 AP SoC U1 VCCBATT Pin p9. The battery is charged from the VCAUX 1.8V rail through a series diode. Both the BBRAM and eFUSE array are secure storage in Zynq device and inaccessible by an adversary. OCM is a secure storage because it has no address or data lines. It can be used to store sensitive software after boot [8]. Last, the decrypted bitstream will be directed to PL for configuration. The flow chart of FSBL is illustrated in Fig. 5.

A UART connection between the PS and the host PC can be set up through connecting to serial port in SDK, shown in Fig. 6. Therefore, the boot information can be retrieved through such a connection.

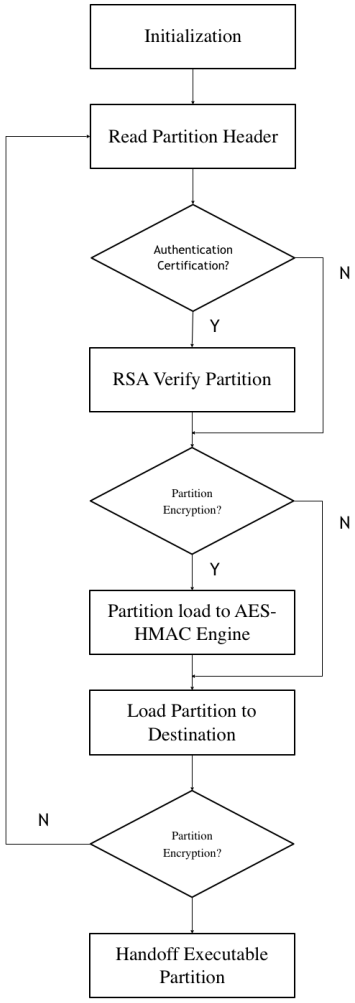

Fig. 5. FSBL Flow

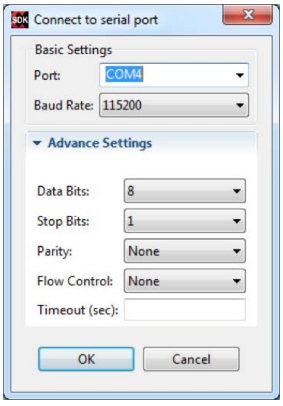

Fig. 6. Serial Port Setup

If we program the key, which has been used to encrypt the bitstream in SDK, onto BBRAM, the system can be booted from SD card successfully, as shown in Fig. 7. However, if a different key is programmed onto BBRAM, the system

boot will fail since encryption and decryption algorithms use different keys. The red LED shown in Fig. 8 indicates boot failure.

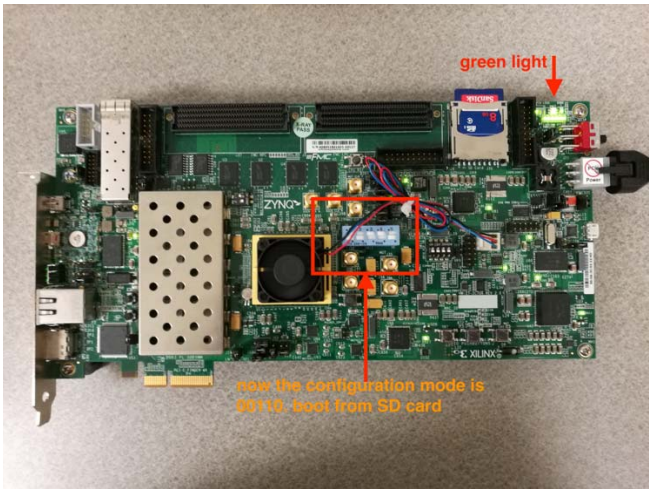

Fig. 7. Secure Boot - with a Correct Encryption Key

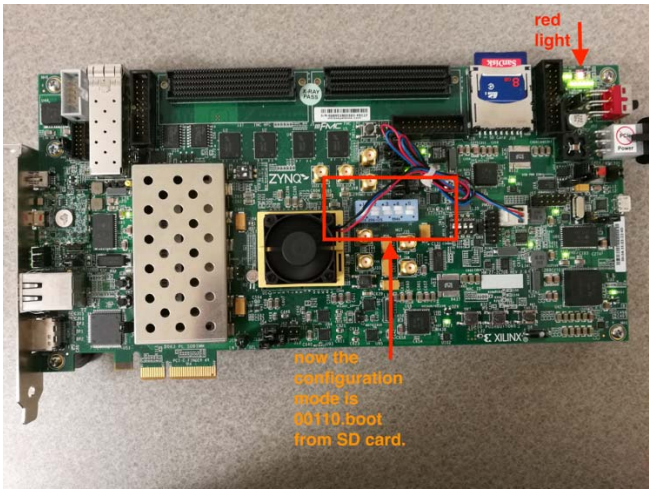

Fig. 8. Secure Boot - with a Wrong Encryption Key

Once the system is booted securely, the device IP address can be retrieved as shown in Fig. 9 through the SDK terminal. Then, we connect both the device and a computer to the same local network. After setting the IP address accordingly, the computer can ping the FPGA device, as shown in Fig. 10.

IV. Conclusion and Future Work

We have studied Xilinx Zynq-7000 Series System-on-Chip (SoC) ZC-706 prototype board to design an IoT device. To defend against threats to FPGA design, we have studied Zynq-ZC706 and been able to encrypt FPGA bitstream to protect the IoT device from bitstream decoding; encrypt system boot image to enhance system security; and ensure the FPGA operates correctly as intended via authentication to avoid spoofing and Trojan Horse attacks. This fundamental study enables future work of securing ZC706 based IoT device through physical

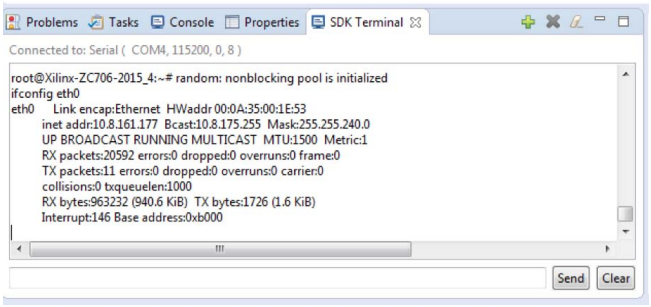

Fig. 9. IP Address of the Device

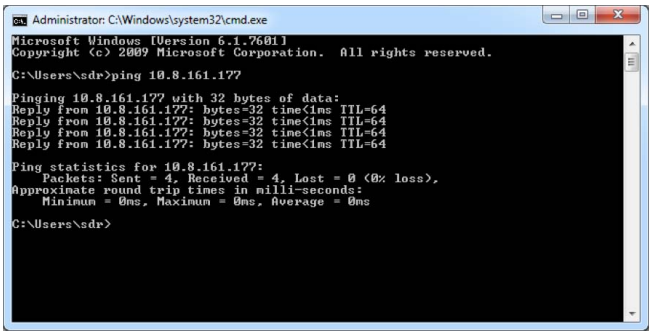

Fig. 10. Network Connection

unclonable function (PUF) key encryption, authentication, and reverse boot, and so on.

References

[1] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, no. 23-24, pp. 2435–2463, Dec. 1999. [Online]. Available: http://dx.doi.org/10.1016/S1389-1286(99)00112-7

[2] G. Vigna, W. Robertson, V. Kher, and R. A. Kemmerer, "A stateful intrusion detection system for world-wide web servers," in *Proceedings of the 19th Annual Computer Security Applications Conference*, ser. ACSAC '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 34–. [Online]. Available: http://dl.acm.org/citation.cfm?id=956415.956437

[3] UG954, "Zc706 evaluation board for the zynq-7000 xc7z045 all programmable soc user guide," March 2016, v1.6. [Online]. Available: https://www.xilinx.com/support/documentation/boards_and_kits/zc706/ug954-zc706-eval-board-xc7z045-ap-soc.pdf

[4] DS190, "Zynq-7000 all programmable soc overview," September 2016, v1.6. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[5] A. Lesea, "Ip security in fpgas," v1.6, 2007.

[6] A. S. Kavita Saroch, "Fpga based system login security lock design using finite state machine," in *IOSR Journal of Electronics and Communication Engineering*, Volume 5, Issue 3, 2013, pp. 70–75.

[7] M. B. Prasun Ghosal, Malabika Biswas, "Hardware implementation of tdes crypto system with on chip verification in fpga," in *Journal of Telecommunications*, Volume 1, Issue 1, 2010, pp. 113–117.

[8] L. Sanders, "Secure boot of zynq-7000 all programmable soc," April 2015, v2.0. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp1175_zynq_secure_boot.pdf