Generalized Architecture for Reconfigurable Hardware Security Incorporating PKI

P.P. Rahoof, Latha R Nair, V.P. Thafasal Ijyas

Abstract--- Software-based techniques represent the dominant paradigm for the implementation of network security. Due to the rapidly growing and dynamic nature of contemporary networks security threats are becoming challenging and the overheads of a purely software approach are mounting to enormous proportions. Higher scales of hardware integration have also produced novel types of hardware threats and attacks which cannot be addressed by a purely software approach. Hence the interest in hardware security approaches is increasing. In this paper, we propose a general architecture for hardware-based security that can address the custom security needs of the users. The architecture incorporates the security model adopted in the Public Key Infrastructure (PKI) model for security.

Keywords--- Authentication, MAC, Encryption, Security, AES, FPGA, PKI.

I. INTRODUCTION

Software based techniques have dominated the field of network security for decades [1]. There are many vendors producing security solutions using the software paradigm. With innumerable new devices connecting every day to the internet, the security risks and issues are mounting to gargantuan proportions day by day. The conflicting demands for a free and open environment for the users while providing the highest levels of security demands a fundamental rethinking on security models [2]. For example, if we take a look at the public key infrastructure (PKI) which is the dominant method for establishing secure connections, we can see that it is riddled with many problems requiring very expensive solutions [3]. A thorough authentication system employing PKI will have to deal with many issues. Update of certificate revocation lists (CRLs), latency, certificate authority (CA) compromise, a wide array of threats and malware techniques etc. are making the software architecture and implementation of security a tremendous grind on the system resources. This in turn leads to loss of productivity and other related problems [4].

Development of hardware-based security implementations have a relevance in this context [5]. Software security solutions run as background applications which use the computing resources intended for the productive purposes of an enterprise. A hardware approach can be implemented as a separate dedicated processor which eliminates the need for using the valuable main processing resources of the system. Other advantages that can be envisaged include cheaper and cost-effective implementation, customization for specific needs, simpler design etc.

Manuscript received September 16, 2019.

P.P. Rahoof, Divison of Computer Science and Engineering, School of Engineering, Cochin University of Science and Technology, Kerala, India. (e-mail: rahoof@cusat.ac.in)

Latha R Nair, Divison of Computer Science and Engineering, School of Engineering, Cochin University of Science and Technology, Kerala, India. (e-mail: latharnair@cusat.ac.in)

V.P. Thafasal Ijyas, Department of Electrical Engineering, College of Engineering, King Khalid University, Abha, KSA.
(e-mail: ithafasal@kku.edu.sa)

Security implementation in the form of hardware modules are indispensable for achieving higher efficiency and throughput [6]. Due to this, various public key cryptographic security protocols like ECDH (key agreement protocol) and ECDSA (sign-verify digital signatures for authentication) have been realized in hardware form by manufacturers like Microchip and Atmel [7, 8]. Emerging paradigms like Internet of Things (IoT), Ubiquitous Computing, Pervasive Computing, Wearable Computing etc. demand the integration of the hardware realization of security in the Public Key Infrastructure (PKI) [9]. Specialized hardware can significantly reduce the time required for executing the computationally-heavy security algorithms [10].

Hardware implementation of security is not a totally novel concept. The TrustZone approach developed by ARM implements embedded security for Cortex processor-based systems [11]. In this, the whole computing environment is divided into two partitions, namely, the 'Trusted' and the 'Non-Trusted' environments. They are called as Trusted Execution Environment (TEE) and Rich Execution Environment (REE). One characteristic of such systems is that security is implemented from the lowest physical layer of the network. An implication of this is that, security starts from the booting of the system itself. In contrast to this, software-based security applications typically start running after the system boot-up. This is one striking advantage of hardware-based solutions. The problem with many hardware solutions is that they are all proprietary.

In this paper, we try to develop a general architecture for hardware-based security approaches. The motivation behind this work is to incorporate all the relevant aspects and components into an architectural model that can address the demands of custom security implementations. We also try to incorporate the proposed architecture in PKI framework.

II. KEY ELEMENTS IN A GENERALIZED ARCHITECTURE FOR SECURITY

In this section, we discuss the key necessary elements and aspects in a generalized architecture for security.

A. Secured and Normal Modules

This is a key feature that is indispensable for any hardware implementation of security in a system. The basic motivation for a security infrastructure is that any system is vulnerable to security threats. Threats can be broadly categorized as hardware threats and software threats. Traditionally, hardware threats used to be less important compared to software threats. But, recently, exploitation of hardware vulnerabilities has become a serious issue.

Added to this is the fact that hardware bugs and vulnerabilities are more difficult to deal with. Many of the recent advances in hardware design like miniaturization ironically contribute to serious vulnerabilities. A recent example is the Rowhammer problem affecting DRAM chips. Side channels are another issue. A general partition of a system into its secured or protected module and the unprotected or normal module is as shown in Figure 1.

This basic model does not incorporate the important issue of the communication between the normal and secure areas, which may be needed. For example, the normal environment may require access to the data, programs or hardware resources under the control of the secure environment. In such a case, a communication channel has to be established between the two environments. The nature of this communication channel depends on the architecture of the isolation between the two modes, viz., secure and non-secure. Typically, this can be implemented as a software or hardware interface. In the TrustZone implementation by ARM, it is implemented as an API named as TrustZone API (TZAPI).

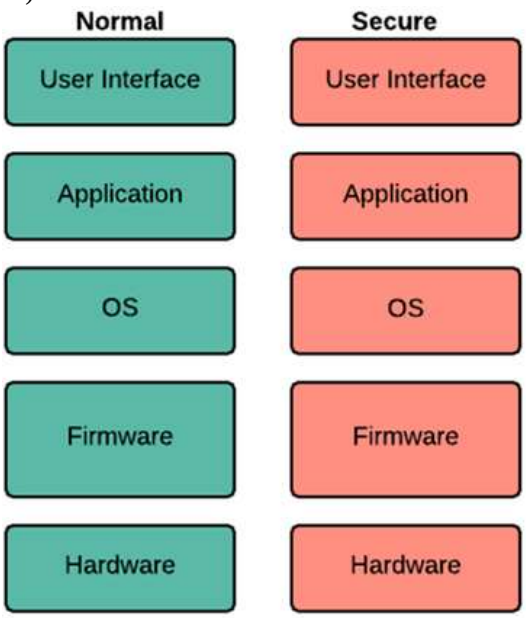

Fig. 1: Modularization for Hardware Implementation of Security

B. Establishing the Root of Trust for the Secure Mode

As mentioned earlier, a striking feature of the hardware model is the establishment of security from system boot. The chain of trust has to be built for the boot sequence itself. The boot image has to be authenticated first. The authentication can be done using a public-private key pair which may be vendor specific [7]. The boot sequence may be as follows. After the system is powered on, a ROM-based boot loader can be used to initialize the critical peripherals. This is a crucial step and demands the implementation of highly secure protocols. ROMs incorporating One-Time-Programmable (OTP) hardware can be used to store different values for the public keys used in the authentication of the boot images. Unique values guarantee protection against class-break attacks characteristic of electronic security systems.

C. Reconfigurability and Programmability of Hardware Security Configuration

A lot of advantages could be accrued to a hardware-based security architecture if it is reconfigurable and programmable. Upgradability, virtualization, higher speed and bandwidth, better security monitoring, customizability etc. are some of these advantages. A convergence of field programmable gate arrays (FPGA), virtualization and software defined networking (SDN) can yield unprecedented benefits to the security implementation. This will do away with some of the limitations and vulnerabilities of the conventional hardware configurations.

Conventional reprogrammable hardware cannot be used out-of-the-shelf for implementing a robust hardware security system. It has many vulnerabilities. Malicious functionality could be added to a system using design tools. The hardware itself could be modified. Invasive techniques could be used to reverse engineer the hardware. Stealing information through hardware side channels is also a formidable risk. In a conventional hardware system like a microprocessor, an attacker has general idea about the design structure and the way of program execution. Specific parts of the overall need to be addressed to mount an attack using a minimal amount of logic gates [12]. A custom-built FPGA implementation has the advantage of having the design configuration not easily traceable [13]. Even if an attacker makes some modification, the modified portion could be relocated to another part of the programmable logic once this is detected.

D. Separability of the Modules

Lack of separation between the different modules of a system implies that security threats emerging in one or more modules may put the entire system on risk. The use of intellectual property (IP) cores sophisticate this problem. The cores used for developing the modules may come from different sources with different levels of trust. Establishment of trust thus requires trust management throughout the entire life cycle of the system. Bit stream specification of the IP cores may hinder the problem of reverse engineering of the design. The whole life cycle has to be secured against vulnerabilities and attacks. Currently, in a life cycle, designing, manufacturing and testing of the system is done at different locations all over the world. There is an associated supply chain with the life cycle. The trust should be established in the manufacturing process itself. Only through this can be trust managed effectively throughout the entire life cycle.

The security of bitstream specification can be augmented by securing the bitstream itself. The authenticated bitstreams can be used for the design. The design need only be decrypted once the key is authenticated. This shown in Figure 2. Message Authentication Code (MAC) - based authentication can be used for authenticating the bitstream [14].

Fig. 2: Authentication and Decryption of Bitstream File

A typical authentication has the following format:

$$X \rightarrow Y: \; N_X \in \{0,1\}^n, M_k(N_X \,||S_i) \qquad (1)$$

X sends Y a cryptographic nonce and a MAC authentication M_k. S_i is the current state which is concatenated with the nonce for the MAC and n is the size of the nonce. The reply from the FPGA could be as follows:

$$Y \rightarrow X: \; M_k(N_X \,||S_{i+1}) \qquad (2)$$

The nonce is used to avoid replay attacks. In a replay attack, an attacker can replay an authentication request many times. A reasonable size of the nonce makes replay computationally expensive with a very low probability for repeating a previously generated value. For this the value of n should be reasonably large.

In a generic composition for encryption and authentication, the MAC operation could precede encryption or vice versa or even done parallel [15]. The choice depends on the specific application. In our scheme, encryption is followed by authentication. Symmetric ciphers can be used for encryption. Arbitrary length messages or bit streams can be encrypted using Cipher Block Chaining (CBC). For this cipher text blocks will be of variable size. The advantage of CBC is that, bit errors in the blocks can be recovered in two decryption operations. Another advantage of CBC is that it can also produce a MAC [16].

A trust management module (TMM) can be built in the FPGA device itself to undertake all the trust related operations. A modified architecture which includes all these features can be depicted as follows:

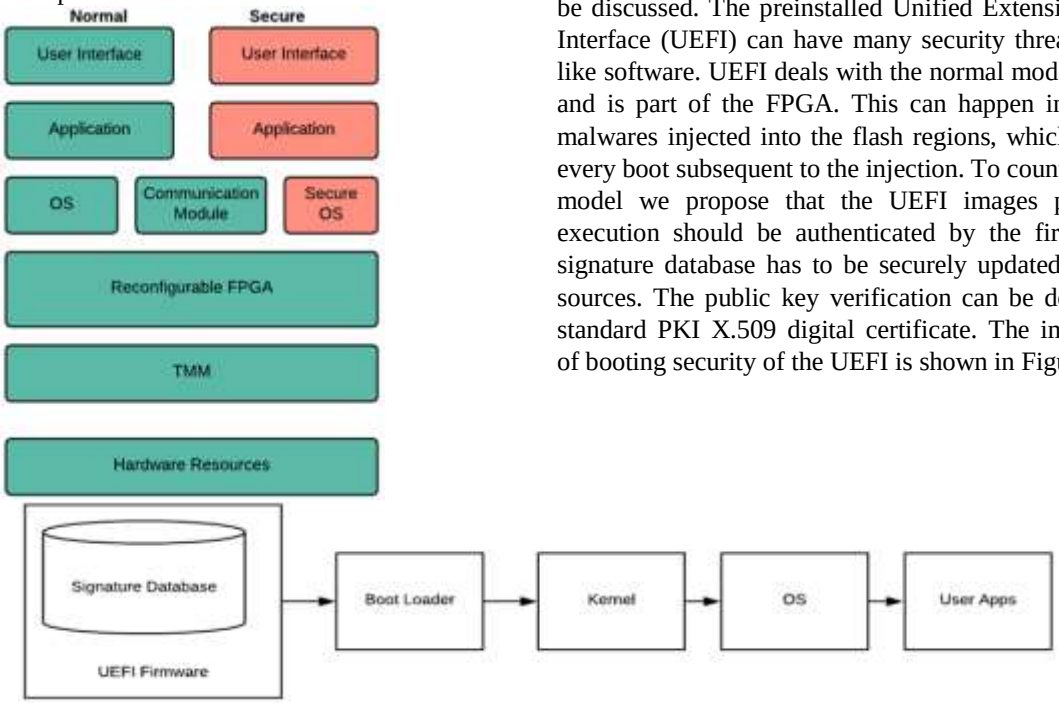

Fig. 4: Block Diagram of Secure Boot

This model defines the trust relationship between the various entities so that the whole model complies with the idea of complete chain of trust. The relationship between the end user or platform owner and the firmware is established through a private-public key pair. Let $\boldsymbol{P1_{priv}}$ be the private key owned by the user while $\boldsymbol{P1_{pub}}$ is the public key in the firmware.

Fig. 3: A Generalized Architecture for Trust-based Hardware Security

The hardware resources include generic resources as well as secure resources like hardware keys, crypto accelerators etc. The TMM undertakes the management of all authentication and decryption related operations and processes. The reconfigurable FPGA can cater to the specific security needs and demands of the users. It also contains the firmware interface of the system. The communication module can deal with the secure communication between the normal and secure modes in the system.

III. INCORPORATING THE ARCHITECTURE IN PKI FRAMEWORK & RESULTS

The trust-based hardware security architecture outlined in the previous sections can be visualized in Public Key Infrastructure (PKI) framework. In this section, we discuss the details of such an implementation. Hardware functionality is generally implemented in the form of hardware service modules (HSM).

As mentioned earlier, the prime motivations of this is better speed and security. Typically, in electronic transactions, SSL negotiations can create bottlenecks even bringing the servers down. A more serious issue is the security of the private keys. Through the use of HSMs, a protected area can be provided for generation and placing the private keys. The keys will always be in an encrypted state in the active memory of the system. As was mentioned earlier, our objective is to establish a complete chain of trust. For this we have to consider the whole stack depicted in Figures 1 and 3.

The security threats to the interfacing firmware has also to be discussed. The preinstalled Unified Extensible Firmware Interface (UEFI) can have many security threats and issues like software. UEFI deals with the normal mode of operation and is part of the FPGA. This can happen in the form of malwares injected into the flash regions, which may run on every boot subsequent to the injection. To counter this, in our model we propose that the UEFI images prior to their execution should be authenticated by the firmware. The signature database has to be securely updated from trusted sources. The public key verification can be done using the standard PKI X.509 digital certificate. The implementation of booting security of the UEFI is shown in Figure 4.

Similarly, the trust relationship between the n^{th} software vendor and the firmware is also established using the private-public key pair. $\boldsymbol{P2n_{priv}}$ is the private key owned by the vendor while $\boldsymbol{P2n_{pub}}$ is the public key in the firmware. A secure storage is used to hold the list of the public keys with the firmware as well as the signature database. The signature database can be updated using the private keys. The updating can be done using runtime services like SetVariable().

A. Secure Boot of UEFI

The platform owner has to verify the credentials of the UEFI image. Typically, the integrity and security have to be verified. A secure boot policy uses white and black lists of the images and certificates comprising the signature database [17]. The verification process consists of the following steps:

1. Check the format and structure of the image.
2. Check whether the signature of the signed image is in the whitelist or black list. If it is the black list, deny its run.
3. If not, then check whether its certificate is authorized. If it is authorized, run.
4. If it is not authorized, check whether its signature is in the white or black list. If in the white list, then run. Otherwise, deny.

B. Security of the TMM

Prior to the handover of control to the firmware interface, there is a cold boot path starting from the power-on of the system till this handover is achieved. This also has to be secure for the achievement of complete chain of trust. The trusted firmware in Arm platforms is an example of this [18]. The cold boot may typically start from a boot loader contained in a secure ROM. The chain of trust commences from the hardware root of trust extending up to the firmware interface. This model is generic in the sense that any third-party solutions for the elements in the chain is feasible as long as their trust has a beginning in the hardware Root of Trust (RoT). This chain of trust with its management aspects is shown in Figure 5.

The whole trust chain process can be explained as follows. When the system is powered on, the primary boot loader in the hardware ROM loads. This boot loader in the hardware has many functions. These include determination of the nature of the boot (whether cold or warm boot), initialization of the architecture including the CPU registers etc., platform initialization and the loading of the image of the next stage of the boot loader. This second stage is part of the TMM. It performs the architectural initialization for the secure and normal modes of operation depicted in Figure 3. It also loads the image of the firmware interface which is part of the FPGA. This completes the path till the firmware interface whose secure boot has been discussed previously.

Now, we will discuss how security is implemented in this process. This is depicted in Figure 5.

Each boot loader image that has to be authenticated has a key certificate as well as a content certificate. There is root of trust public key (RoT Public Key) and ROM boot loader in the hardware which act as the basis of the whole chain of trust. When the system is powered on, the ROM boot loader loads and verifies the RoT Public Key in the content certificate for the second boot loader in TMM using a hash.

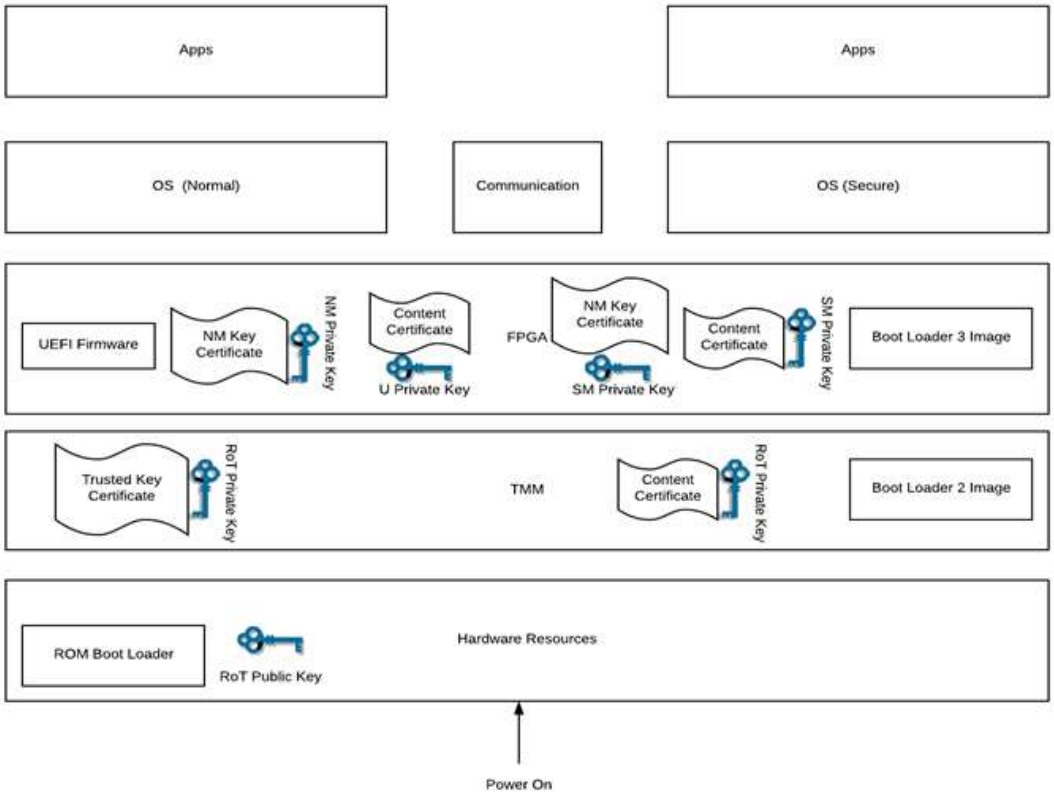

Fig. 5: Security Implementation

Using this verified public key, the content certificate is also verified. The content certificate contains the RoT Public Key as well as a hash for the image of the second boot loader. After this, the ROM boot loader can do the hash verification for the image of the second boot loader and load it.

During the execution of the second boot loader, initially it verifies the RoT public key in the Trusted Key Certificate using hash. Then the Trusted Key Certificate is verified. It contains the public keys for the secure and normal modes. Now, similar to what has happened in the previous stage, the second boot loader verifies the key certificate for the boot loader 3 for the secure mode using the secure mode public key in the Trusted Key Certificate. This key certificate contains the public key for the boot loader for the secure mode. This key is used to verify the content certificate for this boot loader. The hash from this content certificate is used to verify the image file for the third boot loader. Similarly, the key certificate and the content certificate for the normal mode is also verified by the second boot loader. Thus, it can be seen that the security for the system is implemented according to the PKI framework.

The communication module as explained before comprise of the various hardware and software features of the system that enable the communication between the two modes.

IV. CONCLUSION

In this paper, we have proposed as generalized architecture for hardware-based security in computer networks. It encompasses all the essential features needed for an open-ended and customizable security infrastructure that addresses contemporary needs. Also, the framework works according to the PKI model that is widely used for security architectures.

REFERENCES

1. E. Champagne and R. B. Lee, "Scalable architectural support for trusted software," *Proc. International Symposium on High-Performance Computer Architecture*, pp. 1–12, 2010.
2. V. Costan, I. Lebedev, and S. Devadas, 2016. "Sanctum: Minimal hardware extensions for strong software isolation," Proc. USENIX Security Symposium. *USENIX Association*, pp. 857–874, 2016.
3. Peter Gutmann and Ian Grigg, "Security usability," *IEEE Security and Privacy Magazine*, July/August 2005.
4. Arwa Alrawais, Abdulrahman Alhothaily, Xiuzhen Cheng, Chunqiang Hu and Jiguo Yu, "SecureGuard: A Certificate Validation System in Public Key Infrastructure," *IEEE Trans. Vehic. Tech*, vol. 67, no, 6, pp. 5399 – 5408, June 2018.
5. P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a new dimension in embedded system design," *Proc. 41st Design Automation Conference* (DAC '04), San Diego, CA, June 2004.
6. Batina L, Örs SB, Preneel B, Vandewalle J. Hardware architectures for public key cryptography. Integration, *the VLSI Journal*. 2003; 34(1–2):1–64.
7. Atmel. Atmel CryptoAuthentication Device; 2015. Available from: http://www.atmel.com/Images/Atmel-8923S-CryptoAuth-ATECC508A-Datasheet-Summary. pdf [cited 04-07-2019].
8. Infineon. MIPAQ™ Pro A new dimension in smart protection; 2016. Available from: http://www.infineon.com/dgdl/Infineon-MIPAQ_Pro-PB-v02_00-EN.pdf?fileId=5546d4624cb7f111014d6a9e6d356d5e [cited 04-07-2019].
9. Rodríguez-Flores, Luis et al. "Compact FPGA hardware architecture for public key encryption in embedded devices." *PloS one* vol. 13, 1 e0190939. 23 Jan. 2018, doi: 10.1371/journal.pone.0190939.
10. M. Benhani, C. Marchand, A. Aubert, and L. Bossuet, "On the security evaluation of the ARM TrustZone extension in a heterogeneous SoC," Proc. *IEEE International System-on-Chip Conference*, pp. 108–113, 2017.
11. Guajardo J, Güneysu T, Kumar S, Paar C, Pelzl J. Efficient Hardware Implementation of Finite Fields with Applications to Cryptography. *Acta Applicandae Mathematica*. 2006; 93(1–3):75–118.
12. S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and implementing malicious hardware," *Proc. Usenix Workshop on Large-Scale Exploits and Emergent Threats*, 2008.
13. S. Trimberger, "Trusted design in FPGAs," *Proc.44th Design Automation Conference*, San Diego, CA, USA, June 2007.
14. Dworkin, M.: *Special Publication* 800-38B: Recommendation for block cipher modes of operation: The CMAC mode for authentication. NIST, U.S. Dept. of Commerce, 2005.
15. Black, J.: A. "Authenticated encryption." Encyclopedia of Cryptography and Security. Pp. 10-21, *Springer*, 2005.
16. Saar Drimer, "Authentication of FPGA Bitstreams: Why and How," *ARC*, 2007.
17. Beyond BIOS: Developing with the Unified Extensible Firmware Interface, 2nd Edition, Vincent Zimmer, ISBN 13 978-1-934053-29-4, Chapter 10 – Platform Security and Trust, www.intel.com/intelpress.
18. https://developer.arm.com/docs/den0006/latest/trusted-board-boot-requirements-client-tbbr-client-armv8-a