Embedded security for Internet of Things

CONFERENCE PAPER · APRIL 2011

DOI: 10.1109/NCETACS.2011.5751382 · Source: IEEE Xplore

3 AUTHORS, INCLUDING:

Jaydip Sen

National Institute of Science & Technology

91 PUBLICATIONS **341** CITATIONS

SEE PROFILE

Embedded Security for Internet of Things

Arijit Ukil
Innovation Labs
Tata Consultancy Services
Kolkata, India
arijit.ukil@tcs.com

Jaydip Sen
Innovation Labs
Tata Consultancy Services
Kolkata, India
jaydip.sen@tcs.com

Sripad Koilakonda
Innovation Labs
Tata Consultancy Services
Kolkata, India
sripad.k@tcs.com

Abstract—**Internet of Things (IoT) consists of several tiny devices connected together to form a collaborative computing environment. IoT imposes peculiar constraints in terms of connectivity, computational power and energy budget, which makes it significantly different from those contemplated by the canonical doctrine of security in distributed systems. In order to circumvent the problem of security in IoT domain, networks and devices need to be secured. In this paper, we consider the embedded device security only, assuming that network security is properly in place. It can be noticed that the existence of tiny computing devices that form ubiquity in IoT domain are very much vulnerable to different security attacks. In this work, we provide the requirements of embedded security, the solutions to resist different attacks and the technology for defying temper proofing of the embedded devices by the concept of trusted computing. Our paper attempts to address the issue of security for data at rest. Addressing this issue is equivalent to addressing the security issue of the hardware platform. Our work also partially helps in addressing securing data in transit.**

Keywords-ubiquitous computing;Internet of things (IoT); security, embedded device; Trustzone; ARM; confidentiality;

I. INTRODUCTION

Wireless and mobile communication technologies are already widely deployed and their capabilities are ever increasing. New technologies, such as WiMAX, ZigBee, Wireless Mesh Networks, and 4G Networks emerge giving rise to the notion of ubiquitous computing. The vision of Mark Weiser in his famous 1991 article "The Computer of the 21st Century", according to which "the most profound technologies are those that disappear; they weave themselves into the fabric of everyday life until they are indistinguishable from it," is today a reality [1]. Dix et al. define ubiquitous computing as: "Any computing activity that permits human interaction away from a single workstation" [2]. Since then, there have been tremendous advances in mobile and wireless technologies toward supporting the envisioned ubiquitous and continuous computation and, consequently, ubiquitous applications that are intended to exploit the foregoing technologies have emerged and are constantly pervading our life [3]. We can observe that from cars to smart phones, refrigerators to multimedia players - embedded computing increasingly pervade our lives. But most of them are unsecured in nature. Security for these systems is an open question and could prove a more difficult long-term problem than security does today for desktop and enterprise computing. Security issues are nothing new for embedded systems. However, as more embedded systems are connected to the Internet, the potential damages from such vulnerabilities scale up dramatically. Internet connections expose applications to intrusions and malicious attacks. Unfortunately, security techniques developed for enterprise and desktop computing might not satisfy embedded application requirements. Internet connections expose applications to intrusions and malicious attacks. Unfortunately, security techniques developed for enterprise and desktop computing might not satisfy embedded application requirements [4]. System designs for embedded devices are complicated, including multiple independent processor cores, secondary bus masters such as DMA engines, and large numbers of memory and peripheral bus slaves. In addition to these functional components there is typically a parallel system infrastructure that provides invasive and non-invasive debug capabilities, as well as component boundary scan and Built-In-Self-Test (BIST) facilities. Due to this kind of importance as well as the pervasive deployment of embedded devices from home to big enterprises, embedded device security becomes a big issue. Many research initiatives have been undertaken to counter the issues of security in embedded systems. In fact, security has been the subject of intensive research in the context of general-purpose computing and communications systems. However, security is often misconstrued by embedded system designers as the addition of features, such as specific cryptographic algorithms and security protocols, to the system. In reality, it is a new dimension that designers should consider throughout the design process, along with other metrics such as cost, performance, and power. The challenges unique to embedded systems require new approaches to security covering all aspects of embedded system design from architecture to implementation. The diverse security requirements are especially apparent in embedded systems where increased connectivity, portability, and pervasive design objectives are need to be considered. In fact, pervasive networks have led to widespread use of embedded systems, like cell phones, PDAs, RFIDs etc., in increasingly diverse applications. Many of these embedded system applications handle sensitive data (e.g., credit card information on a mobile phone/PDA) or perform critical functions (e.g., medical devices or automotive electronics), and the use of security protocols is imperative to maintain confidentiality, integrity and authentication of these applications. Evolution of embedded systems towards devices connected via Internet, wireless communication or other interfaces as well as the trend towards always growing numbers of devices (IoT) requires a re-consideration of embedded systems engineering processes. It is no longer possible to achieve the required level of security by adding security measures late in the development process. Security

engineering as stated above needs to be part of the system development in all stages of the process. Typically embedded systems have low computing power and finite energy supply based on a battery, and these factors are at odds with the computationally intensive nature of the cryptographic algorithms underlying many security protocols. In addition, secure embedded systems are vulnerable to attacks, like physical tampering, malware and side-channel attacks. Thus, design of secure embedded systems is guided by the following factors: small form factor, good performance, low energy consumption (and, thus, longer battery life), and robustness to attacks.

The paper is organized as follows. In section II, we have discussed the requirements of embedded security in IoT. Then, in section III, embedded security solutions to counter some of the security challenges are mentioned. In section IV, we described trusted computing and its importance in IoT security. Lastly, we conclude in section V.

II. EMBEDDED SECURITY REQUIREMENT IN IOT

With the advent of powerful computing and communication gadgets and tools, the possibility of invasion on our daily life is increased many folds. Now, with the advent of IoT (Fig. 1), we are encountering a third wave of hacking—one that encompasses not only wired computers and networks, but intelligent devices: wireless phones, routers and switches, printers, SCADA (Supervisory Control And Data Acquisition) systems, and even medical devices. This new hacking wave is poised to bypass the amateur "street-creed" phase and move directly to well-honed, massively coordinated, sophisticated attacks. It is now becoming clear that hacking's third wave will almost certainly include terrorist cyber-strikes against the utility and industrial infrastructure (the "smart grid")—a danger we can no longer dismiss as a spy movie scenario. One of the most common attacks on IoT is "war driving," in which hackers drive around a neighborhood, hunting for unsecured wireless nodes. In the latest twist on war driving, a security expert cruised around Fisherman's Wharf, armed with a cheap RFID scanner and a low-profile antenna, and managed to clone half a dozen electronic, wallet-sized passports in an hour [19]. Ross Anderson [27] has several chapters devoted to the basic vulnerabilities of devices and systems used for banking, energy metering, and wireless mobile communication, signaling the increasing importance of this area.

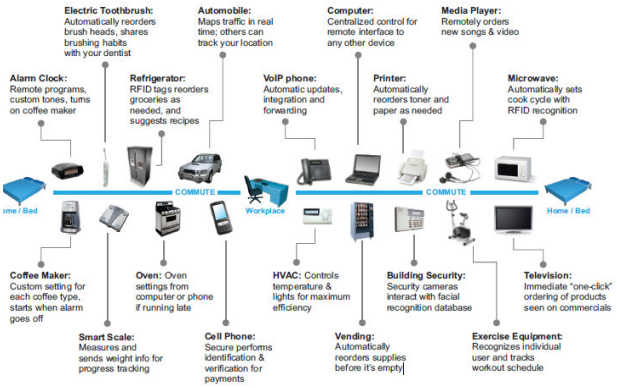

Figure 1. IoT architecture

Another challenging area which embedded security needs good amount of attention is: in-vehicular security. Ever since electronic devices were installed into cars, they have been a feasible target for malicious attacks or manipulations. Mileage counter manipulation, unauthorized chip tuning or tachometer spoofing [5] are already common. Many analyses [7] can verify the safety and reliability of vehicle networks against random failures. Analyses that consider also intended malicious manipulations, i.e. discuss vehicular communication security, are still very rare [8]. Thus, most existing automotive communication systems are virtually unsecured against malicious encroachments [9]. We can observe that from cars to smart phones, refrigerators to multimedia players - embedded computing increasingly pervade our lives. But most of them are unsecured in nature. Security for these systems is an open question and could prove a more difficult long-term problem than security does today for desktop and enterprise computing. Security issues are nothing new for embedded systems. However, as more embedded systems are connected to the Internet, the potential damages from such vulnerabilities scale up dramatically. Internet connections expose applications to intrusions and malicious attacks. Unfortunately, security techniques developed for enterprise and desktop computing might not satisfy embedded application requirements. Internet connections expose applications to intrusions and malicious attacks. Unfortunately, security techniques developed for enterprise and desktop computing might not satisfy embedded application requirements [4]. With the advent of IoT and pervasive nature of embedded computing, attacks on network, data, hardware and software are in rise [19 -20]. Many embedded systems are especially susceptible to a type of non-invasive attacks called side-channel attacks. Non-invasive techniques consist of software attacks (using viruses, worms, etc) and attacks based on the statistical analysis of operational characteristics of the device to extract secret information. When a system is under attack, different goals are targeted; the first kind of attack is the extraction of secret information, the second one is trying to put the system out of order [28]. Coron et al. [29] formulated a set of statistical tests which can be used to detect the presence of side-channel leakage from any given cryptographic computation.

III. EMBEDDED SECURITY SOLUTION

There are many existing solutions to counter different attacks. Encryption of information is used for confidentiality. The most popular cipher algorithms are: RSA, ECC, AES, 3DES.The hash of information is used to check the integrity of a message by providing a signature which is unique for each message. The most known algorithms are MD5 and SHA. In addition, non-repudiation, availability and authenticity are guaranteed by communication protocols like IPSec for example. Most of these algorithms and processes are very much computationally intensive. So, we require dedicated hardware or Digital Signal Processors (DSP). A dedicated processor implements specific instruction dedicated to security primitives. An analogy can be done with DSP through its multiplication-accumulation instruction for digital signal processing. In most cases, security processors are dedicated to one class of ciphering algorithm (symmetric or asymmetric). Specific execution units are added into the datapath. Authors in

[21] propose processors with instructions for symmetric ciphering algorithms. Specific instructions have been defined like logical operation (xor-add) or data permutation. In CryptoManiac processor, a fast and flexible co-processor for cryptographic workloads is developed. Authors have presented an analysis of a 0.25um physical design that runs the standard Rijndael cipher algorithm (3DES) 2.25 times faster than a 600MHz Alpha 21264 processor. For processors dedicated to asymmetric ciphering algorithms [22], specific instructions are defined. For instance to efficiently compute the modular exponentiation is used in ECC and RSA. However, there still exists significant difference between requirements of security processing and the capability of an embedded processor. This difference is termed security processing gap [23].

System designs for embedded devices are complicated, including multiple independent processor cores, secondary bus masters such as DMA engines, and large numbers of memory and peripheral bus slaves. In addition to these functional components there is typically a parallel system infrastructure that provides invasive and non-invasive debug capabilities, as well as component boundary scan and Built-In-Self-Test (BIST) facilities [8]. Due to this kind of importance, complexity as well as the pervasive deployment of embedded devices from home to big enterprises, embedded device security becomes a big issue. Many research initiatives have been undertaken to counter the issues of security in embedded systems. We find great treatment on the issues of embedded system security in [10], where authors have described security requirements, design challenges, basic concepts, different security protocols like Secure Socket Layer (SSL) [11], open SSL [12], architectures. The SSL protocol is typically layered on top of the transport layer of the network protocol stack, and is either embedded in the protocol suite or is integrated with applications such as web browsers. This is shown in Fig. 2.

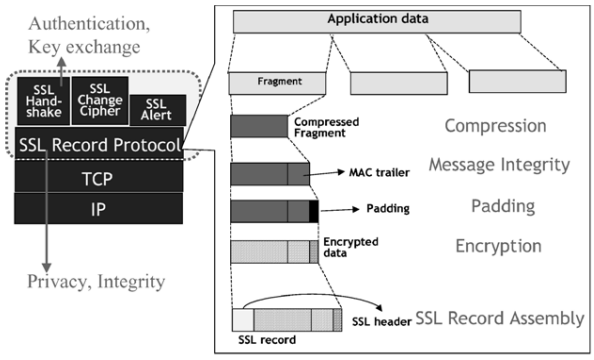

Figure 2. SSL protocol, with an expanded view of the SSL record protocol

IV. TRUSTED COMPUTING

In order to provide security at the physical or execution level, we need to build our solution based on secure execution environment (SEE). An SEE is a processing unit which is capable of executing applications in a protected manner, meaning the attacks originating from outside the SEE cannot tamper with code and data belonging to the SEE. The first building block of an SEE is of course a secure processor – either a dedicated processor or one capable of supporting a secure mode, which is hardware compartmentalized from the non-secure mode. Utilizing a dedicated processor has the advantage of ease of separation as well as offloading the main processor from handling security tasks. The disadvantage of a dedicated processor is the increase in silicon footprint. The advantages of using one processor with two compartments is exploiting remaining Millions Instructions Per Second (MIPS) if available, while the disadvantages include the need for better system design, and harder proof of security robustness. The second building block is secure code and data memory – most likely dedicated on-chip RAMs. It is important to remember that whenever code is present outside the SEE memory it should be integrity protected against modifications (and possibly protected for confidentiality by means of encryption if required). Whenever data is present outside the SEE memory it should be protected both for confidentiality and for integrity [25]. In this respect, we find that recently good amount of development has taken place in embedded platform security. Among the commercial releases, Trusted Platform Module by Atmel [13] and Trustzone by ARM [14] are worth mentioning. Trusted platform module (TPM) is to provide the minimal hardware needs to build a trusted platform in software. While usually implemented as a secure coprocessor, the functionality of a TPM is limited enough to allow for a relatively cheap implementation – at the price that the TPM itself does not solve any security problem, but rather offers a foundation to build upon. Thus, such a module can be added to an existing architecture rather cheaply, providing the lowest layer for larger security architecture. The main driver behind this approach is the Trusted Computing Group (TCG), a large consortium of the main players in the IT industry, and the successor to the Trusted Computing Platform Alliance (TCPA) [15]. TrustZone consists of a hardware-enforced security environment providing code isolation, together with secure software that provides both the fundamental security services and interfaces to other elements in the trusted chain, including smartcards, operating systems and general applications. TrustZone separates two parallel execution worlds: the non-secure 'normal' execution environment, and a trusted, certifiable secure world. TrustZone offers a number of key technical and commercial benefits to developers and end-users. TrustZone software components are a result of a successful collaboration with software security experts, Trusted Logic, and provide a secure execution environment and basic security services such as cryptography, safe storage and integrity checking to help ensure device and platform security. By enabling security at the device level, TrustZone provides a platform for addressing security issues at the application and user levels. Below (Fig. 3 & 4) we show the hardware and software architecture of ARM Trustzone. It is to be noted that one of the main features of trusted computing is secure boot. Secure Boot (also known as High Assurance Boot) is a technique for verifying and asserting the integrity of an executable image prior to passing the control to it. Assuming the verification mechanism is based on the digital signature of the image being verified, the reliability of this verification is at best as good as the reliability of the protection mechanism provided in the device for the public key of the image signer. The most important assumption here is that the code that performs the integrity verification process is itself trustworthy.

To assert this assumption, the implementations typically put the public key material (as well as the verification code) into non-writable areas of memory, which in turn are protected using some sort of hardware protection mechanism. Generic Secure boot architecture is shown in Fig. 5 [17]. In this approach, the first step after boot-up is to verify the integrity of the Secure Boot code itself using digital signature verification. Next, the Secure Boot code performs integrity checking of basic security parameters (such as the signers' public key), and then after that validation of system images (such as the entire kernel or individual system libraries) occurs, and finally the user-space application validation takes place. The integrity of each layer relies on the integrity of the layers underneath. At any point, if the verification fails, the system can be put in a halt-state. In ARM Trustzone, the secure boot scheme adds cryptographic checks to each stage of the Secure world boot process. This process aims to assert the integrity of all of the Secure world software images that are executed, preventing any unauthorized or maliciously modified software from running. The secure boot process implements a chain of trust. Starting with an implicitly trusted component, every other component can be authenticated before being executed. The ownership of the chain can change at each stage - a PuK (Personal Unblocking Key) belonging to the device OEM might be used to authenticate the first bootloader, but the Secure world OS binary might include a secondary PuK that is used to authenticate the applications that it loads. Unless a design can discount hardware shack attacks the foundations of the secure boot process, known as the root of trust, must be located in the on-SoC ROM. The SoC ROM is the only component in the system that cannot be trivially modified or replaced by simple reprogramming attacks. Storage of the PuK for the root of trust can be problematic; embedding it in the on-SoC ROM implies that all devices use the same PuK. This makes them vulnerable to class-break attacks if the PuK is stolen or successfully reverse-engineered. On-SoC One-Time-Programmable (OTP) hardware, such as poly-silicon fuses, can be used to store unique values in each SoC during device manufacture. This enables a number of different PuK values to be stored in a single class of devices, reducing risk of class break attacks. Another secure boot implementation is found for Linux platform, which is part of SELinux [18]. To provide the appropriate levels of protection, these environments are enhanced with mandatory access control (MAC) mechanisms. One method to achieve a MAC is by implementing Role-Based Access Control (RBAC). NSA's SELinux, among other features such as MLS (Multi Level Security), provides Linux with MAC through RBAC [18]. With the explosive growth of mobile devices and application, it is true that the next generation of open operating systems won't be on desktops or mainframes but on the small mobile devices, which enables greater integration with existing online services. Developed by the Open Handset Alliance (led by Google), Android is a widely anticipated open source operating system for mobile devices that provides a base operating system, an application middleware layer, a Java Software Development Kit (SDK), and a collection of system applications. Android restricts application interaction to its special APIs by running each application as its own user identity. This controlled interaction has several beneficial security features. Android protects applications and data through a combination of two enforcement mechanisms, one at the system level and the other at the inter-component communication (ICC) level. ICC mediation defines the core security framework. It is built on the guarantees provided by the underlying Linux system. As the central point of security enforcement, the Android middleware mediates all ICC processes by reasoning about labels assigned to applications and components. A reference monitor provides MAC enforcement of how applications access components. Security enforcement in Android occurs in two places: each application executes as its own user identity, allowing the underlying Linux system to provide system-level isolation; and the Android middleware contains a reference monitor that mediates the establishment of ICC. Both mechanisms are vital to the phone's security, but the first is straightforward to implement, whereas the second requires careful consideration of both mechanism and policy [24]. In [26], authors have presented SCANDROID, (Security Certifier for anDroid) a tool for automated security certification of Android applications. SCANDROID statically analyzes data flows through Android applications, and can make security-relevant decisions automatically, based on such flows. In particular, it can decide whether it is safe for an application to run with certain permissions, based on the permissions enforced by other applications. Alternatively, it can provide enough context to the user to make informed security-relevant decisions.

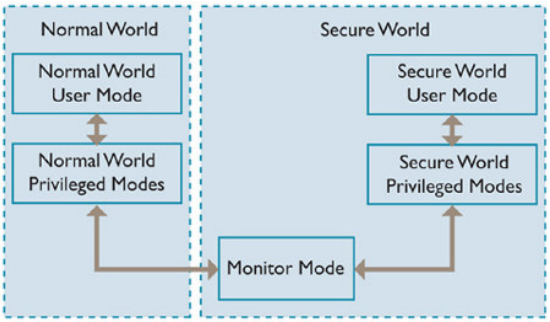

Figure 3. Trustzone hardware architecture

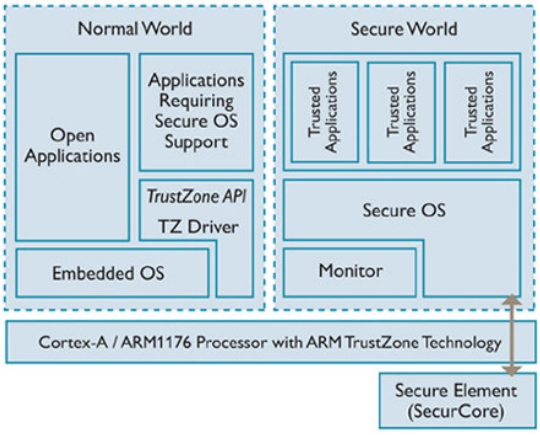

Figure 4. Trustzone software architecture

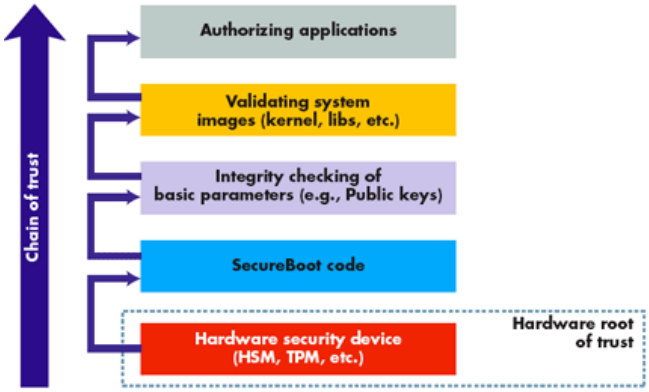

Figure 5. Generic secure boot architecture

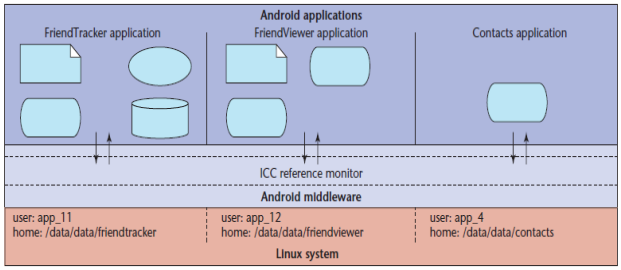

Figure 6. Android security architecture

V. CONCLUSION AND FUTURE WORK

With the advent of pervasive nature of today's computing, security is becoming very critical for wide range of applications. As most of today's and next generation computing applications involve embedded systems, in this work, we have presented the requirements, issues, designs and solutions of embedded design to counter the different attacks. While some aspects of security have been addressed in the context of traditional general-purpose computing systems, embedded systems usher in many new challenges. We have highlighted the security-related problems faced by designers of embedded systems, and outlined recent developments and innovations to address them. Several issues, however, still remain open to find a holistic solution to the problem of embedded system security.

Efficient security processing alone is of limited use if an embedded system does not successfully address attacks that could potentially compromise its security. A clear cost and risk analysis becomes essential to determine the levels of attack resistance that a device must support. Since attacks continue to increase in sophistication, the development of countermeasures remains a challenging and on-going exercise. It is also important to remember that countermeasures applicable to one system (e.g., smartcards) may not be able to applicable to other embedded systems (e.g., PDAs or smart phones). Thus, system-specific attack-resistance measures are crucial.

IoT mainly consists of tiny devices with limited processing power. As the attackers become sophisticated, it becomes necessary to dedicate entire co-processor with high scalability to offer entire security features that an embedded system may require. It is very crucial to reduce susceptibility to side-channel attacks through the use of hardware techniques that reduce correlation between data values and side-channel information, like power, time, etc.

Another important issue needs to be adopted is the adaptability of the embedded systems to dynamically accustom it to the required security requirements in order to minimize the unnecessary computational expenses due to using complicated cryptographic algorithms. For example, if it is sufficient to use 1024 bit RSA, the system should not apply 2048 bit RSA. This context-aware feature is vital for next generation embedded systems (particularly in IoT) where the (mobile) system can face different kinds of hostile situation with varieties and sophistications of attacks. Dynamic adaptation needs to become a de facto rule for embedded system security. This in a simplified form can be a formulation of scalable security protocols whose security level and energy consumption can be varied by altering the number of protocol rounds and the complexity of operations in each round.

REFERENCES

[1] Mark Weiser, "The Computer for the Twenty First Century," Scientific American, pp. 94-104, September, 1991.

[2] A. Dix, J. Finlay, G. Abowd, and R. Beale, "Human-Computer Interaction," Prentice Hall, 3e, 2004.

[3] G.D. Abowd, G.R. Hayes, G. Iachello, J.A. Kientz, S.N. Patel, and M.M. Stevens, "Prototypes and paratypes: Designing mobile and ubiquitous computing applications," IEEE Pervasive Computing, vol. 4, no. 4, pp.67–73, 2005.

[4] P.Koopman, "Embedded system security," IEEE Computer, vol. 37, issue. 7, pp. 95-97, 2004.

[5] Ross J. Anderson, "On the security of digital tachographs," 5th European Symposium on Research in Computer Security (ESORICS '98), pp. 111–125, Springer-Verlag, London 1998.

[6] Richard Evans and Jonathan D. Moffett, "Derivation of safety targets for the random failure of programmable vehicle based systems," In SAFECOMP, pp.240–249, 2000.

[7] Maxim Raya and Jean-Pierre Hubaux, "The security of vehicular networks," Technical report, Laboratory for Computer Communications and Applications (LCA), School of Computer and Communication Sciences, EPFL, Switzerland, March 2005.

[8] M. Abramovici, C. Stroud, and J. Emmert, "On-Line BIST and BIST-Based Diagnosis of FPGA Logic Blocks," IEEE Trans. on VLSI Systems, Vol. 12, No. 12, pp. 1284-1294, 2004.

[9] K. Lemke, C. Paar, and M. Wolf (Eds.), "Embedded Security in CarsSecuring Current and Future Automotive IT Applications," Springer-Verilag, 2006.

[10] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in Embedded Systems: Design Challenges," ACM Transactions on Embedded Computing Systems, vol. 3, no. 3, pp. 461 – 491, 2004.

[11] URL: http://wp.netscape.com/eng/ssl13

[12] URL: http://www.openssl.org

[13] URL: http://www.atmel.com

[14] URL: http://www.arm.com

[15] URL: https://www.trustedcomputinggroup.org

[16] T. Alves and D. Felton, "TrustZone: Integrated Hardware and Software Security, Enabling Trusted Computing in Embedded Systems," ARM Whitepaper, July 2004.

[17] H. Nahari, J. Ready, "Employ a secure flavor of Linux," Embedded Systems Design, pp. 20 - 29, Oct, 2007.

[18] URL: http://www.nsa.gov/research/selinux/

[19] "Attacks on Mobile and Embedded Systems: Current Trends," Mocana whitepaper, www.mocana.com, 2009.

[20] D. Dagon, T. Martin, and T. Staner, Mobile Phones as Computing Devices: The Viruses are Coming!, IEEE Pervasive Computing, vol. 3, no. 4, pp. 11- 15, 2004.

[21] L. Wu, C. Weaver, and T. Austin, "CryptoManiac: a fast flexible architecture for secure communication", Proceedings of the 28th annual international symposium on Computer architecture (ICSA'01), vol. 29, issue. 2, May 2001.

[22] H. Eberle et al, "A Public-Key Cryptographic Processor for RSA and ECC," Proceedings of the Application-Specific Systems, Architectures and Processors (ASAP'04), 2004.

[23] S. Ravi, A. Raghunathan, N. Potlapally, and M. Shankaradass, "System design methodologies for wireless security processing platform," in Proc. Design Automation Conf., pp. 777-782, June 2002.

[24] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security," IEEE Security and Privacy, vol. 7, issue. 1, pp. 50 -57, 2009.

[25] E. Rippel, "Security challenges in embedded design," www. discretx.com, 2009.

[26] A. P. Fuchs, A. Chaudhuri, and J. S. Foster, "SCanDroid: Automated Security Certification of Android applications", http://www.cs.umd.edu/~avik/papers/scandroidascaa.pdf, 2009.

[27] R. Anderson, "Security Engineering: A Guide to Building Dependable Distributed Systems," Wiley, 2008.

[28] T. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," IEEE Trans. Computers, vol. 51, pp. 541- 552,May 2002.

[29] J. S. Coron, D. Naccache, and P. Kocher, "Statistics and information leakage," ACM Transaction on Embedded Computer Systems, vol. 3, pp. 492 - 508, Aug. 2004.