Security needs in Embedded Systems: Challenges

Shaliesh D. Nandgaonkar[1], Dr. Viddhulata Mohite [2]
[1]Research Scholar, Singhania University, Rajasthan, India
[2] Professors, Department of E&TC, M.G.M College of Engineering, Navi Mumbai, India

Abstract — *The paper discusses the hardware and software security requirements in an embedded device that are involved in the transfer of secure digital data. The paper gives an overview on the security processes like encryption/decryption, key agreement, digital signatures and digital certificates that are used to achieve data protection during data transfer. The paper also discusses the security requirements in the device to prevent possible physical attacks to expose the secure data such as secret keys from the device. The paper also briefs on the security enforced in a device by the use of proprietary security technology and also discusses the security measures taken during the production of the device.*

Keywords— *Embedded systems, security, architecture, hardware design,processing requirements, battery life, security protocols, cryptographic algorithms, encryption, decryption, authentication, security attacks, tamper resistance.*

I. INTRODUCTION

The embedded or handheld devices are getting increasingly connected and are more and more involved in network communications. The users of these devices are now able to execute almost all the network/internet applications that run in a PC on these devices. These devices are also increasingly involved in transfer of secure data through public networks that needs protection from unauthorized access and thus the security requirements in embedded devices have become critical.

The secure data falls in different categories requiring different levels of security. According to whose interest the protection of the data is, the secure data can be classified as two: the users private data and the user restricted data. The users private data are those data which when its security is compromised impacts directly on the user. A simple example of compromising such security is having access to a user's internet banking password. But in case of user restricted data, it's not the user but the content (data) provider who suffers direct loss on compromising the security of that data. The examples of such data are digital multimedia content such as copyrighted digital photos, audio and video contents.

The secure data not only requires protection during data transfer but also while handling the data at the end user devices. Vulnerability at the end user device, like easy access to the secret keys that are used to encrypt or decrypt the data, can easily turn down the entire security measures. The protocol involved for the secure transmission of either of the above mentioned contents through a public network uses more or less the same techniques but the handling of the user restricted data at the user's end involves much more care as the content is protected from the user itself.

Thus an embedded device must implement methods or protocol for secure data transfer and also should implement security methods to defeat attempts of unauthorized access of secure data from the device. The security needs for an embedded device thus can be classified into two: Security needs for data transfer and Security needs within the device.

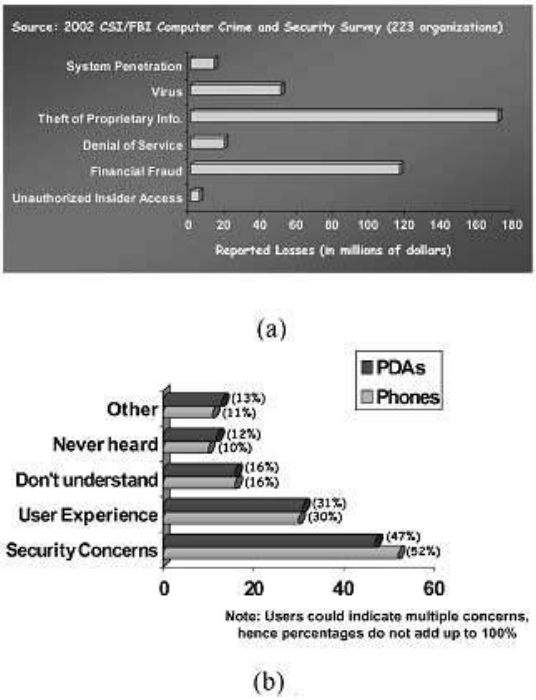


Fig.1. (a) The cost of insecurity (source: [Computer Security Institute]) and (b) factors preventingthe adoption of mobile commerce (source: [ePaynews]).

II. SECURITY REQUIREMENTS OF EMBEDDED SYSTEMS

Embedded systems often provide critical functions that could be sabotaged by malicious entities. Before discussing the common security requirements of embedded systems, it is important to note that there are many entities involved in a typical embedded system design, manufacturing, and usage chain. Security requirements vary depending on whose perspective we consider.

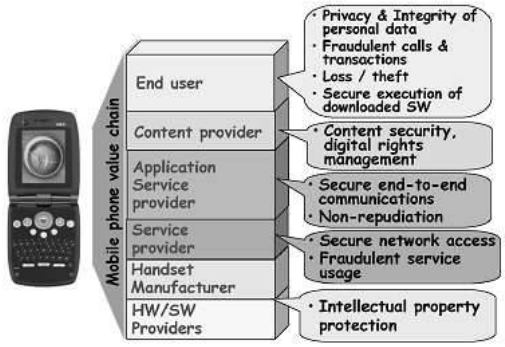


Fig. 2. Security requirements for a cell phone.

Fig. 3. Common security requirements of embedded systems.

security requirements from the viewpoint of the provider of HW/SW components inside the cell phone (e.g., baseband processor, operating system), the cell phone manufacturer, the cellular service provider, the application service provider (e.g., mobile banking service), the content provider (e.g., music or video), and the end user of the cell phone. The end user's primary concerns may include the security of personal data stored and communicated by the cell phone, while the content provider's primary concern may be copy protection of the multimedia content delivered to the cell phone, and the cell phone manufacturer might additionally be concerned with the secrecy of proprietary firmware that resides within the cell phone. For each of these cases, the set of untrusted (potentially malicious) entities can also vary. For example, from the perspective of the content provider, the end user of the cell phone may be an untrusted entity. While this section outlines broad security requirements typical of embedded systems, the security model for each embedded system will dictate the combination of requirements that apply

III. SECURITY NEEDS FOR DATA TRANSFER

The data in a public network passes through a number of untrusted intermediate points. Therefore the secure data must be scrambled in such a way that the data will be useless or unintelligible for anyone who is having unauthorized access to the secure data. This can be achieved with the help of cryptographic methods such as Encryption/Decryption, Key Agreement, Digital Signatures and Digital Certificates. The use of these cryptographic methods in an embedded system to achieve data security is explained in the following sections.

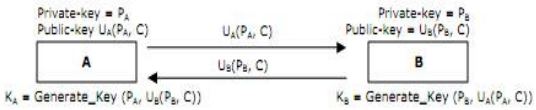
Encryption is the process of scrambling/encrypting any amount of data using a (secret) key so that only the recipient, who is having access to the key, will be able to descramble/decrypt the data. The algorithm used for the encryption can be any publicly available algorithm like DES, 3DES or AES or any algorithm proprietary to the device manufacturer. The key is known only between the communicating devices and will typically of length 100s of bits. If publicly available algorithms are used, the security of the transferred data totally depends on the secrecy of the keys used for the encryption. Sharing and maintaining the secret key between the communicating devices without any unauthorized entity getting access to the keys is important for foolproof secure data communication. These keys can be embedded in the device prior to the communication, exchanged offline in a secure manner or established online using any key agreement algorithm.

When there are 100's of devices in a network, sharing and maintaining secret keys between all the devices for data encryption seems difficult, even unrealistic. This is where the Key Agreement Algorithm is used. Using Key Agreement algorithm, a shared secret can be established between communicating parties without the need for exchanging any secret keys or secret parameters online or offline. This works as follows.

Key agreement algorithm is a public-key cryptography algorithm. For devices that use Key agreement algorithm will have a private-key and an associated public-key. The private-key is generally a

random number of hundreds or few thousands of bits and the public-keys are derived from the private-key using the one-way function specified by the key agreement algorithm. One-way functions are mathematical function in which the forward operation can be done easily but the reverse operation is too difficult that it is practically impossible. The public-key is derived using private-key on the forward operation of the one-way function. The reverse operation of obtaining the private-key from the public-key is too difficult that it is practically impossible.

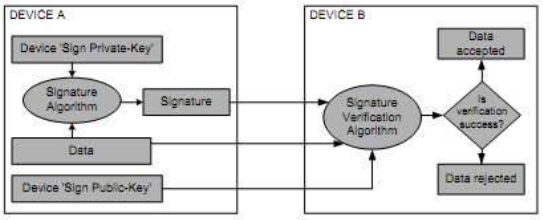
The devices that need to establish shared secret between them exchanges their publickeys and other public constants, if any. Both the device on receiving the other devices public-key performs key generation operation using its private-key to obtain the shared secret. The key agreement works such a way that the shared secret calculated by both the devices will be the same.



In the key agreement algorithm the private-key used is a secret key and should not be shared with any other devices in the network, even to the device to which it is communicating. It is only the agreed shared secret key that is known between the communicating devices. It must also be ensured that the private-key is not disclosed from the device. The safe storage of the private-key of key agreement algorithm on the device is thus important.

IV. Digital Signature

The device in a network may be communicating with the unknown or less familiar device located 100s of kilometers apart. The communication may also require routing through many intermediate points. During Key Agreement process, for establishing a secret key, any middlemen can substitute a devices public-key to its public-key and thus results in establishing a shared secret with the device. Therefore, for establishing shared secret using the key agreement algorithm, it is important for device to receive an authenticated public-key from the peer. For authenticated exchange of public-key, Digital Signature and Digital Certificates are used.



Digital signature is a public-key method to verify the authenticity of a received data from the peer. In digital signature, like the key agreement algorithm, a device uses a pair of keys, 'sign private-key' and 'sign public-key'. Only the device knows its sign private-key whereas the sign public-key is distributed to all the communicating devices. A device signs the message using a signatures algorithm with its sign private-key to generate a signature and any device that has got the access to the sign public-key of the signed device can verify the data with the signature using the signature verification algorithm. If any third party modifies the data or signature, the verification fails. Since only the signed device knows its sign private-key, it will be impossible for any other device to forge the signature. Examples of Digital Signature algorithms are RSA, DSA or ECDSA.

V. Security Processing Requirements and Architectures

Security processing refers to the computations that must be performed in a system for the purpose of security. In this section, we will analyze the challenges imposed by security processing on embedded system design in greater detail, using the popular Secure Sockets Layer (SSL) protocol as an example.

The cryptographic primitives described in Section 3 are used to provide the basic services offered by most security protocols: encryption, peer authentication, and integrity protection for data exchanged over the underlying unprotected networks. In this section, we will examine the functioning of a popular security protocol SSL [SSL],which is widely used for secure connection-oriented transactions.

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. The SSL protocol itself consists of two main layers as shown in Figure 4. The SSL record protocol, which forms the first layer, provides the basic services of confidentiality and integrity. The second layer includes the SSL handshake, SSL change cipher, and SSL alert protocols. Let us now examine how the SSL record protocol is used to process application data. The first step involves breaking the application data into smaller fragments. Each fragment is then optionally compressed. The next step involves computing a message authentication code (MAC), which facilitates message integrity. The compressed message plus MAC is then encrypted using a symmetric cipher. If the symmetric cipher is a block cipher, then a few padding bytes may be added. Finally, an SSL header is attached to complete the assembly of the SSL record. The header contains various fields including

the higher-layer protocol used to process the attached fragment.

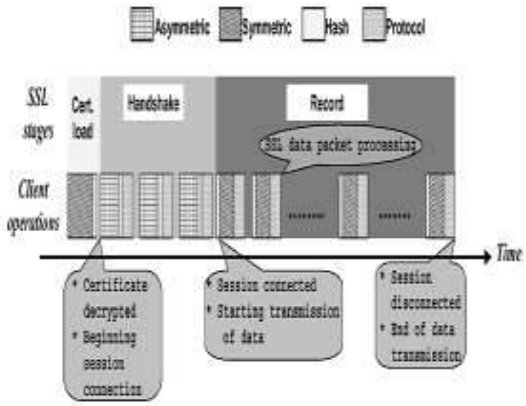


Fig. 5. Typical sequence of client-side operations performed during an SSL session.

VI. Security Needs within the Device

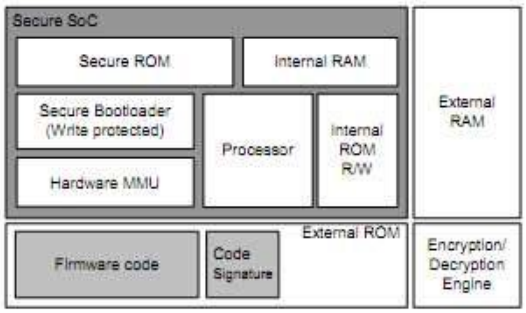
Whether it is the private-key of any public-key algorithm as discussed in section 2 or it is any previously negotiated shared secret between the devices, the security of data transferred depends in the secrecy of these keys. To enforce additional security, some cryptographic algorithms may also specify a set of constant values that should not be disclosed from the device. These secret keys and secret values stored in the device that requires protection from unauthorized exposure are referred as 'secret keys' in this document.

The secret keys are stored inside the device, some even for the lifetime of the device. Hardware and software security measures implemented in the device must defeat any attempts of unauthorized access to retrieve these secret keys. Also, there are data such as the Root CA Certificate in the device that can be disclosed but should be prevented form unauthorized modification. If Root CA certificate can be modified, then the attacker can make the device to accept any certificate by substituting a fake root CA certificate and thus defeating the purpose certificate and secured communication. It is therefore also important that the security in the device is such that the data such as Root CA Certificates in the device is not subjected to unauthorized modification.

The level of security within the device varies depending on the nature of the protected content. The need for device security is more in the case of device handling user restricted data like copy-protected* video than in the case of user's private data like personal files or bank transactions. This is mainly because, in the case of user's private data since the user will suffer the direct loss on compromising such data, he/she will be responsible for restricting the physical access to the secret keys and other secured contents stored in the device. Also, the general implementation of secure data transfer protocols recommends a unique secret key for each device. Therefore if the hardware security of any of the device is compromised, it doesn't affect the security of other device in the network. But in the case of user restricted data, compromising the secret key of a single device results in the compromise of the security of all the copy-protected content handled by that device. One vulnerable device can thus results in helping an unauthorized device to access the copy protected content, decrypt it and distribute countless copy of the copy protected content.

VII. Secure SoC

The Secure SoC provides physical protection to secret keys by keeping the components like Secure ROM, which is handling the secret keys, inside the Secure SoC. During execution time, the protected secure keys from the Secure ROM has to be loaded to the RAM in clear text and during that time the bus from the Secure ROM to the RAM can be monitored to access the secret keys. This can be prevented by allocating buffers for secret keys or intermediate values of cryptographic operations involving secret keys in the Internal RAM of the Secure SoC. This prevents the protected keys being available to any bus outside the Secure SoC. The Secure Bootloader in the Secure SoC ensures that the device boots up with the genuine OS or firmware with right process privileges. The Memory Management Unit (MMU) configured by the OS permits the access to the buffers in the Internal RAM that involves secret key operations only to the secure processes with special OS privileges. In the case where the Secure ROM is limited or pre-programmed by the hardware manufacturer, the Secure ROM can be programmed with a master key. This master key can be used to encrypt and store the device secret keys in the internal ROM.



VIII. Proprietary Technologies for Secure Data Transfer

Proprietary security modules are implemented on some devices for the secure data transfer between the compliant devices. The modules can be some or all the modules mentioned in section 2 like Encryption/Decryption, Key Agreement, Digital Certificate or Digital Signature modules.
The proprietary technologies implemented in device are usually kept secret to enforce additional security for data transferred between the devices. It is therefore ensured by the device manufacturer to not to disclose the technology, from or outside the device, and thereby compromising the security enforced by these technologies. The device manufacturer should find a method to store the software module in the device so that the secrecy of the technology is not compromised from the device. Code (binary) of these proprietary technology software modules usually will be in clear text inside the device. Thus if someone can get access to the code in the device it may be easy to extract and understand the implementation of the software module with the help of code reengineering tools like 'objdump'. Thus the device must prevent access to retrieve the code by placing it inside Secure SoC or encrypting and storing the code using a secret key knowing only to the device manufacturer. If the code is stored encrypted, the boot loader of the device must support the decryption and loading of the code during execution and the secret key used for encryption and decryption of the code can be stored in the device by the methods specified in section 3.

A proprietary technology can be either a device specific or a standard specific. In the case of a device specific technology, the secure data transfer can happen only between the devices of same manufacturer. An e.g. can be Apple's iPod music player. Apple can use their proprietary technology for secured transfer of files between their compliant devices or applications like iPod, iTunes and iTunes store. Since the devices are from a single manufacturer, keeping it secret, other than any user retrieves and understands the code through any weak links from the device, seems to be practically possible.

IX. Conclusion

Security is critical to enabling a wide range of applications involving embedded systems. While some aspects of security have been addressed in the context of traditional general-purpose computing systems, embedded systems usher in many new challenges. This paper highlighted the security-related problems faced by designers of embedded systems, and outlined recent technological developments and innovations to address them. Several issues, however, remain open at the intersection of security and embedded system design. The interplay of flexibility, performance, power consumption, and security level makes choosing the "right" security solution a highly complicated process. In addition to these metrics, cost and design turn-around times play a crucial role in deciding the security architecture. In many design scenarios today, it becomes hard to evaluate the effectiveness of a given security solution, or to trade-off between the above metrics, due to the absence of complete system-level analysis and evaluation tools.

Thus the hardware security measures implemented in the device are a trade of between the cost of implementation and the cost of the data protected. Achieving a cost effective yet foolproof method to protect the secret keys and secure data within the device will be a boon to the owner of the contents that needs security, especially to the content provider of copy-protected digital contents.

Reference

[1] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996

[2] FIPS PUB 186-2, Digital Signature Standard (DSS), January 2000, Available at http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf

[3] RSA Laboratories, PKCS#1 v2.1: RSA Cryptography Standard, June 2002, http://www.rsa.com/rsalabs/node.asp?id=2125

[4] RFC 2631, Diffie-Hellman Key Agreement Method, June 1999, Available at http://tools.ietf.org/html/rfc2631

[5] Certicom, Standards for Efficient Cryptography, SEC 1: Elliptic Curve Cryptography, Version 1.0, September 2000, Available at http://www.secg.org/download/aid385/sec1_final.pdf

[6] ITU, Recommendation X.509, Available at http://www.itu.int/rec/T-REC-X.509-200508-I

[7] FIPS 197, Advanced Encryption Standard (AES), November 2001, Available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[8] NIST, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, May 2004, Available at http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf

[9] FIPS 140-2, Security requirements for cryptographic modules, May 2001, Available at http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[10] OpenCable, OpenCable System Security Specification, October 2006, Available at http://www.cablelabs.com/specifications/OC-SP-SEC-I07-061031.pdf

[11] DTLA, Digital Transmission Content Protection Specification Volume 1 (Informational Version), October 2007, Available at http://www.dtcp.com/data/info%2020071001%20DTCP%20V1%201p51.pdf

[12] Anoop MS, Public Key Cryptography – Applications algorithm and mathematical explanations, May 2007, Available at http://msitbox.blogspot.com/2008/03/public-key-cryptography.html

[13] Anoop MS, Elliptic Curve Cryptography - An implementation guide, May 2007, Available at http://msitbox.blogspot.com/2008/03/elliptic-curve-cryptography.html

[14] Openssl, http://www.openssl.org