Enhancing security in Wireless Networks

Stelios Koutroubinas*, Theodore Karoubalis*, Panagiotis Rozos, Panayiotis Nastou* and Panayiota Bei

*Member, IEEE

Abstract — *Vulnerability of the safety of the LAN is one of the biggest disadvantages of a wireless network. The integration of a Smart Card or a Crypto Memory to a WLAN device as a repository for all critical information (certificates, keys, credentials) that today are stored and manipulated in the "unsafe" operating system may allow the biggest deployment of WLAN systems as they have a lot of advantages.*[1].

Index Terms — Communication, Home electronics, Security, Smartcard, WLAN.

I. INTRODUCTION

As cryptography made a great progress in the past years, smartcards (SC) has proven to be an ideal medium for safely storing cryptographic keys and algorithms. In the last two years, the members of the 802.11i Task Group have given a great effort in order to provide WLAN users a more powerful security protocol. Currently, most protocol implementations store sensitive data of the protocol into OS repositories or public directories of the host. This strategy makes an 802.11 station less portable since most of the critical data (certificates and private keys) is stored into a specific host and thus it is difficult to use the station in another host since sensitive information must be transferred from one host to another. Moreover, storing sensitive data into public places and repositories is less secure since malicious applications (worms, Trojans, etc) can be used to retrieve sensitive data during OS operations.

In this paper, a new system for enhancing the Security of 802.11 Stations with the use of Smart Cards is presented, based on the 802.11i requirements. This approach is smoothly integrated without the need to alter the 802.11i security standard, and complements its capabilities in a reliable way without introducing protocol changes, since it aims to integrate seamlessly with it.

[1] This work was done in ATMEL Hellass SA, Stadiou St. – Platani, 26500 Rio, Greece). ATMEL Hellas is a Design Center of ATMEL, Part of ATMEL Multimedia & Communication Group. ATMEL Hellas SA today employs 140 employees for the design of communication products for WLAN, VoIP and Bluetooth. ATMEL Hellas SA designs the chip which includes one or more risc CPUs or DSPs and additional logic needed for the implementation of the communication protocols (e.g. IEEE 802.11), the firmware that runs on the embedded CPUs and the drivers needed to operate in several operating systems. Finally ATMEL Hellas provides the reference design to the end customer for easier adoption and fast "time to market" product introduction.

II. 802.11 Infrastructure Functional Model Overview

In general, the 802.1X protocol [5] performs authentication in a layer above the IEEE 802.11 MAC layer. The authentication processing has been removed from the IEEE 802.11 MAC, delegating this function to 802.1X. The 802.11 MAC passes all data packets it receives from higher layers, which means that all 802.1X messages are sent as 802.11 data messages to/from Authenticator, delegating the filtering of any unauthorized traffic to 802.1X. This filtering and the steps of the 802.1X authentication are opaque to the 802.11 MAC itself. The 802.1X encapsulates the Extensible Authentication Protocol (EAP) [10,11], which supports multiple authentication methods such as certificates and public key authentication, into 802 frames (EAP over LAN or EAPoL) with a few extensions to handle unique characteristics of 802 LANs.

If an IEEE 802.1X Authentication Server exists (enterprises) then authentication credentials must be distributed to the Supplicant and Authentication Server (AS) prior to any association (Distribution of Certificates). The Authenticator and Authentication Server authenticate each other and establish a secure channel using an authentication protocol (e.g. RADIUS). A Supplicant discovers the AP's security policy through passively monitoring Beacons or through active probing. The AP and a Supplicant exchange 802.11 Open System Authentication Request/Response and Association Request/Response. Upon the completion of the association, both the Supplicant and the Authenticator have determined the security policy (which authentication mechanism and which ciphers will utilize) that will use in the authentication phase and the during data transmission. The Supplicant and Authentication Server authenticate each other (e.g., EAP-TLS) [12,13] and generate a Pairwise Master Key (PMK) independently. The PMK is sent from the AS to the Authenticator over the secure channel.

In the absence of an Authentication Server (home or small business WLANs), the administrator sets a secret key into the Authenticator which is called Pre Shared Key (PSK) and informs every user to use it when his/her Supplicant is associated in the PSK mode. In this case, the PSK plays role of the PMK. Currently, a Supplicant application saves the PSK in an OS repository either protected by a password or not. This obviously limits the portability of the Supplicant, i.e if the supplicant is to be installed into another workstation we have to set again the PSK and to remove it from the previous workstation, and the security of the system since it is feasible for someone with access to the workstation to retrieve the PSK (dictionary attacks) and to use it for his/her purpose. Thus, the

PSK is considered as sensitive data and must be stored in a secure store with which a client device is equiped. Storing the PSK into a secure store, there is no need for setting again the PSK if the user tries to connect the client device to another workstation.

The Authenticator initiates firstly the 4-way Handshake protocol [6] which is a Key Management protocol through which the existence of the PMK and that it is current is confirmed, a unique Pairwise Transient Key (PTK) from the PMK is derived and the unicast encryption and integrity keys into IEEE 802.11 are installed. On the successful completion of the 4-way handshake the Authenticator initiates the Group Key Handshake in order to transport safely the Group Transient Key (GTK) to Supplicant and install it.

Upon the successful completion of the above Key Management Protocols, the AP changes the state of the IEEE 802.1X access port, opening the controlled port to permit general data traffic to pass onto the DS.

III. SMART CARD ARCHITECTURE

As cryptography made a geat progress in the 1960's and the security mechanisms could be proved mathematically, smartcards (SC) proved to be an ideal medium for safely storing cryptographic keys and algorithms. The International Organisation for Standardization (ISO) standard 7810 "Identification Cards- Physical Characteristics" defines physical properties such as flexibility, temperature resistance, and dimensions for three different cards formats (ID-1, ID-2, ID-3). The Smart Card standard, ISO 7816, is based on the ID-1 format. In this section, a secure memory, a microprocessor smart card and a cryptographic coprocessor cards, which are ID-1 type smart cards are presented briefly.

Secure memories are typically much less expensive and much less functional than microprocessor cards. They contain EEPROM and ROM memory, as well as some security logic [8]. In the simplest designs, logic exists to prevent writing and erasing of the stored data while more complex designs allow for memory read access to be restricted and they offer Random Number Generator. Their disadvantage is the fact that cannot perform cryptographic operations which means that some sensitive data needed for these operations (like private keys) must be transfered from the secure memory to the microprocessor that will perform these operations. It is ideal for storing secrets (like PSK and premaster secret mentioned in the previous section) only if the data are transfered encrypted from the secure memory to the host [8].

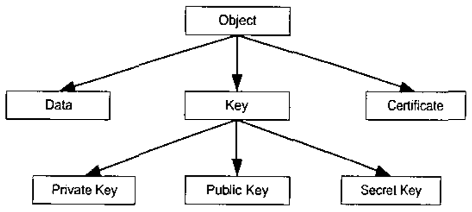

Figure 1. Cryptoki's Object Classes

Basically, microprocessor cards include a CPU and its working RAM, a ROM for the operating system, and an EEPROM for storing sensitive data and IO peripherals. Asymmetric cryptographic algorithms of the day require math calculations where its execution by an 8-bit microprocessor with very little RAM can take several minutes. If a cryptographic processor or a 32-bit microprocessor [9] is added to the architecture, the time required for these calculations is reduced to a few microseconds. Although the cost of the final system is increased, the security of the system is also increased since the private keys never leave the smart card.

IV. CRYPTOGRAPHIC TOKEN INTERFACE-CRYPTOKI

In this section, the fundamental concepts of the PKCS #11 standard [7] that specify an Application Programming Interface (API), called "Cryptoki" pronounced Crypto-Key and short for "cryptographic token interface", to device that hold cryptographic information and perform crypptographic functions is presented. Since the mechanical characteristics and electrical connections as are the methods for supplying commands and receiving results are well defined by existent standards (ISO-7816), it is remained to define particular commands for performing cryptography. The primary goal of Cryptoki is a low-level programming interface that abstracts the details of portable cryptographic devices, such as those based on smart cards, PCMCIA cards, and smart diskettes, and presents to the application a common model of the cryptographic device, called a "cryptographic token" or simply token. Non-cryptographic functions of the device are left to other interfaces.

A cryptographic token is a device that stores objects and can perform cryptographic operations. Figure 1 presents the three object classes that the Cyptoki defines. A data object is defined by an application, a certificate object stores a certificate and a key object stores a cryptographic key which may be a private, a public, or a secret key. A token can create and destroy objects, manipulate them and search for them. Besides the cryptographic functions a token can perfrom, it may have an internal Random Number Generator.

Whenever an application is to gain access the token 's objects and functions, it opens one or more sessions. A session provides a logical connection between the application and the token. It can be read/write (R/W) session (i.e. it can create, read, write and destroy both public and private objects) or read-only (R/O) session (i.e it can only read private objects while it can create, read, write and destroy public objects).

Cryptoki recognizes two token user types: the Security Officer (SO) and the normal user. The role of the SO is to initialize the token and to set the normal user PIN's, and possibly to manipulate some public objects. Private objects can be accessed only by a normal user and that access is granted only if the normal user has been authenticated. The normal user cannot log in until the SO has set the normal user's PIN.

The Cryptoki API consists of a number of functions:

- General Purpose Functions
- Token Management Functions

- Session Management Functions
- Object Management Functions
- Encryption/Decryption Functions
- Message Digesting Functions
- Signing and MACing Functions
- Functions for verifying signatures and MACs
- Dual-purpose cryptographic functions
- Key Management Functions
- Random Number Generation Functions

A token may be able to perform all the above functions or a small subset of them. Since the cost of a Smart Card with high cryptographic capabilities is high, it is designer's responsibility to choose the Smart Card that provides certain cryptographic capabilities and then to use the appropriate subset of the Cryptoki subset.

V. Smart Card based Station Architecture

Figure 2 shows two different interfaces between a Wireless Client Application and a Station. The left column shows how a Wireless Client Application and station currently communicate. The Application passes non-cryptographic operations to a station through station's driver interface. The cryptographic operations of the 802.1X authentication are executed in Host engine. The certificates and the keys needed during authentication are stored into Operating System repositories and are retrieved by using Operating System calls. Not only is not such a structure portable since the user must transfer sensitive data from one system to another whenever the user wishes to get network access using the same station device from a different system but its security is highly dependent on the OS resistance to malicious application code.

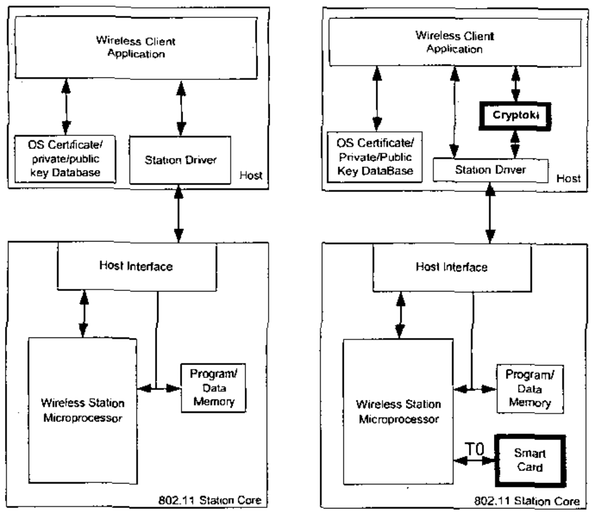
Wireless Client Application
OS Certificate/ private/public key Database
Station Driver
Host
Host Interface
Wireless Station Microprocessor
Program/ Data Memory
802.11 Station Core
Wireless Client Application
Cryptoki
OS Certificate/ Private/Public Key DataBase
Station Driver
Host
Host Interface
Wireless Station Microprocessor
Program/ Data Memory
T0
Smart Card
802.11 Station Core

Figure 2. Station's Structure with a Smart Card

A smart card based Station allows to Client Applications (right structure in Figure 2) to offer users the chance to store the sensitive data defined in previous sections either in the Operating System repositories or into station's smart card. Moreover, the smart card may offer cryptographic operations like random number generation, signing messages and verifying signatures and MACs so as sensitive data must never leave the smart card leading to highly secure systems. Non cryptographic functions are passed from the Application to station through the station driver interface straightforward while cryptographic operations are passed from the application to station using the Cryptoki API described in the previous section.

VI. System Implementation

If we observe the block diagram of a WLAN system, it consists of the Medium Access Controller (MAC), the Baseband, the Radio, the PA, the volatile and non-volatile (EEPROM or flash) memory. High integrated products include all the above in two or three chips. In the system used for our development the heart of the system was the AT76C505A chip, a MAC processor from ATMEL with a USB interface to the host, integrated memory and Baseband. As the reference design was targeting a small form factor board (USB memory stick) we selected the crypto memory form factor but the same applies on a Smart Card as it is a matter of packaging.

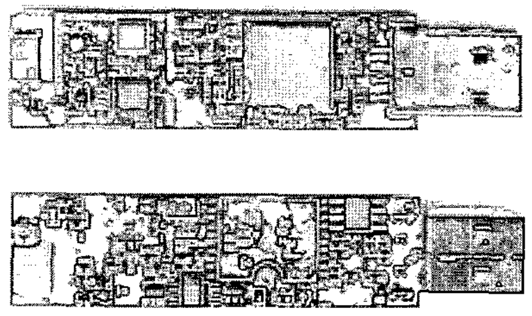

Fig 3. USB stick with WLAN and Smart Card functionality

The USB stick printed circuit board (PCB) is 6 layers. For the WLAN and the crypto memory only functionality a 4 layers card is realized. The material is FR4 and the thickness of the board is approximately 1mm. Since this is a WLAN implementation, special care has been taken for the undisturbed operation of the Radio section. All layers are separated into two sections, one analog and one digital. Different voltage regulators supply the two parts. The secure memory is located on the digital section and it is fed from the digital voltage regulator.

Crypto memory was easily integrated to the system using two IO pins. On the processor side, the allocated resources for the manipulation of the crypto memory are minimum. The overall overhead in the chip function is also minimized. *Communication is made using a synchronous Two-Wire* interface.

Moreover the communication with the crypto memory is handled in the firmware level with the minimum code overhead. The Smart Card function is transparent to firmware and is abstracted to the use of one general thus powerful command. No function specific information is stored in the firmware level reserving confidentiality. Accessing of the crypto memory is impossible without the use of its specification since no information concerning the commands and parameters is hardcoded anywhere in the firmware and driver level.

The driver functionality is enhanced with two extra commands thus the crypto memory is accessible from the driver level with two simple read/write access commands. Finally a C++ Class was build to handle the communication details with the MAC processor in the driver so that it can be easily integrated in any application.

All the above are designed and implemented in order to allow applications and drivers to have transparent access on the crypto memory, allowing them to develop any kind of application that will store safely their critical data.

The crypto memory selected was AT88SC1616C device. It supports two communication protocols: The ISO 7816-3 T=0 asynchronous protocol and the two-wire synchronous communication protocol. Since in our case the AT88SC1616C is used in a closed circuit board the Two-Wire Synchronous protocol is selected as more suitable. It uses the minimum processor recourses and is more efficient in terms of communication speed.

All communication aspects are handled by the above protocol reserving the minimum possible firmware code size overhead, and the minimum physical allocated resources from the side of the processor.

In the driver level Communication with the crypto memory is performed through the USB interface. The Driver accesses crypto memory through the use of two new commands.

READ_SMART_CARD is used for commands that read data from crypto memory.

WRITE_SMART_CARD is used for commands that write data to the smart cards.

The use of one general access command is avoided because the USB specification that handles the communication between Driver and Firmware does not support bi-directional accesses, and in order to overcome this crypto memory commands would have to be hardcoded in the Driver code. However hardcoding commands and parameters in the code were considered both unsafe and costly in terms code size thus the use of unidirectional accesses was preferred.

The user that wishes to access crypto memory either from the driver or the application level should be familiar with the crypto memory commands and know which ones are for read and which for write accesses.

The crypto memory functionality is exposed to software applications in much the same way with the rest of the Driver functionality. Two new Custom Defined Object Identifiers are used one for READ and one for WRITE accesses respectively.

OID_CUSTOM_READ_SC is used to retrieve data from crypto memory

OID_CUSTOM_WRITE_SC is used to insert data to crypto memory

Crypto memory accesses are performed through the easy access library, which is a general code library for accessing the ATMEL drivers' functionality.

A test application (CryptoTester) that exhibits crypto memory accesses is build in order to verify the solution and to be used as a reference to developers. Through this simple application the user can send commands to crypto memory. Underlying functionality is transparent to the user who only needs to be familiar with the crypto memory specification.

Furthermore, a C++ library for the crypto memory is created that the user can add to his application and access crypto memory, quick easy and without having to posses any knowledge on the underlying functionality. Once more the simple and powerful *sendCommand* interface that can be used to send any command to crypto memory.

The proposed solution eliminates the need of an external smart card reader that some PCs or laptops have in order to increase security. At the same time it enables user mobility as with the use of this USB stick someone can transfer his credentials to any PC without sacrificing security as he is not setting up the keys to any PC he is using to connect in a WLAN.

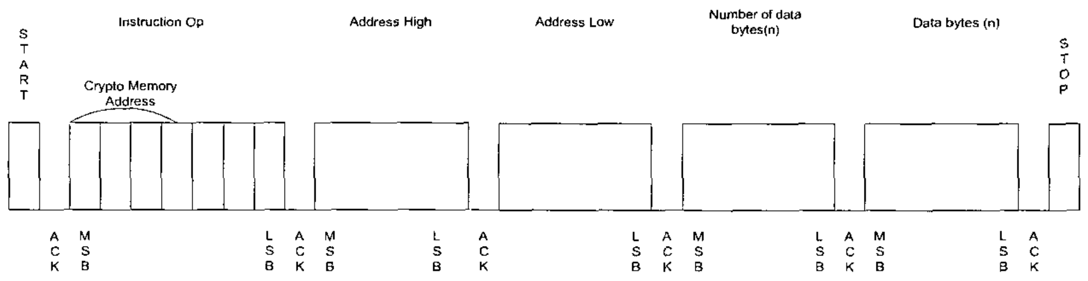

Fig. 4. Smartcard command communication protocol

VII. Conclusion

Security is one of the most critical issues and the proposed architecture is already tested in order to be used in a product. An extension to the described functionality available in the existing board will be to add a NAND flash in order to add bulk storage in the same board. This reference design can support 64, 128 and 256 Mbyte of NAND flash. The NAND flash is controlled from the ARM who implements the WLAN

MAC functions. No extra controller is needed for the USB interface. No additional driver is needed for the versions o Windows that support Mass Storage functionality. For the operating system it is a USB composite device with 2 interfaces. Each interface has a separate function. One is the WLAN and the other is the Mass Storage. Secure memory on the board can be used to encrypt the stored data. This may require some extra special drivers but it will enable the encryption of the data stored in the memory with very secure methods.

REFERENCES

[1] IEEE Document P802.11/D6.1.97/5, *"Wireless LAN, MAC and Physical Specifications"*, June 1997

[2] Dr.Bernard Aboba, *"Wireless LANs:The 802.1x revolution"*, Internet World Wirelless West,December 2001

[3] ATMEL Corporation, *"802.11b Media Access Controller (MAC) and baseband USB Interface.AT76C505"*, Datasheet . [Rev. =374CX-WLAN-01/03]

[4] ATMEL Corporation,*"CryptoMemory®16 Kbit,AT88SC1616C Summary"* [Rev. 2030ES-SMIC–04/03]

[5] IEEE Document Std 802.1X-2001,*"IEEE Standard for Local and Metropolitan area networks"* ,June 2001

[6] IEEE Medium Access Control (MAC) Security Enhancements, IEEE Task Group i P802.11i/D6.0.

[7] PKCS #11 v2.11: *Cryptographic Token Interface Standard, RSA Security Inc. Public-Key Cryptographic Standards* (PKCS).

[8] 8 x 128 x 16 CryptoMemory, AT88SC1616C, ATMEL datasheet.

[9] Secure Microcontroller for Smart Cards, AT90SC9616RC ATMEL datasheet.

[10] Blunk, L. and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998.

[11] Aboba, B. and D. Simon, "PPP EAP TLS Authentication Protocol", RFC 2716, October 1999.

[12] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, November 1998.

[13] Blake-Wilson, S., Nystrom M., Hopwood D., Mikkelsen, J. and Wright T., "The Transport Layer Security (TLS) Extensions", RFC 2246, June 2003.