On the Generation of X.509v3 Certificates with Biometric Information

Guillermo Martínez-Silva[1], Francisco Rodríguez-Henríquez[1], Nareli Cruz-Cortés [2] and Levent Ertaul[3]
[1] Computer Science Department
Centro de Investigación y de Estudios Avanzados del IPN
Av. Instituto Politécnico Nacional No. 2508, México D.F.
g_m_silva@yahoo.com, francisco@cs.cinvestav.mx
[2]Center for Computing Research (CIC), National Polytechnic Institute (IPN),
Av. Juan de Dios Bátiz s/n, Esq. Mendizábal, Col. Zacatenco, 07738 Mexico City, Mexico.
nareli@cic.ipn.mx.
[3] Department of Math & Computer Science
California State University, East Bay.
levent.ertaul@csueastbay.edu

Abstract

We present the kernel implementation of a Mobile Certification Authority (MCA). Our MCA kernel is able to issue digital certificates fully-complying with the X.509v3 standard; it supports either RSA or ECDSA as a public key cryptosystem engine and; it can incorporate biometric-based user identification information (in the form of fingerprint recognition) to the digital certificate. The MCA application was entirely written in C++ and it was tested in an iPAQ Pocket PC h5550. Our experiments show that an ECDSA-based digital certificate using the NIST recommended 163K elliptic curve constitutes an ideal selection for wireless environments. Such certificate can be generated with a size of 1635 bytes, and 592 bytes when including (or not) biometric information, respectively.

keyword: X.509v3 digital certificates, mobile PKI, biometric authentication

1 Introduction

Public key cryptosystems and security protocols such as the Digital Signature Standard (DSS), Elliptic curve Digital Signature Algorithm (ECDSA), RSA and the Diffie-Hellman protocol [7, 8] have been utilized for providing information security services such as: *data authentication*, *data integrity* and *non-repudiation*, among others. Unfortunately, such schemes do not provide by themselves reasonable protection against potentially devastating attacks such as man-in-the-middle attack, identity-misbinding attack, identity usurpation and so on [12].

Consequently, it has been necessary to create an infrastructure able to cover aforementioned security gaps. That infrastructure is known as Public Key Infrastructure (PKI) [9]. The *de facto* X.509 PKI standard [3, 5] comprises a collection of software, cryptographic technologies and services that allow the protection of the information transactions security in a distributed system. This way, PKI X.509 integrates *digital certificates, public key cryptography and Certification Authorities (CA)* in a single security architecture. The main responsibility of a CA is to issue digital certificates to its users and to publish and maintain a Certificate Revocation List (CRL).

PKI X.509 defines a digital certificate as a document that binds user's information (such as name, address, organization, etc.) to his/her corresponding public key. It is signed by a CA in order to guarantee its validity and integrity. It can be used as a *token-based* identification method, which corresponds to one of the three identification methods commonly used in practice as it is explained below.

Accurate user-identification is essential to offer a reliable *access control* security service. Typically, information systems employ three types of identification methods. *Token-based* authentication which relies on something that the user has (such as a smart card); *knowledge-based* authentication which identify users who can prove knowledge of something (such as a password) and finally; *biometric-based* authentication which identify users by measuring human characteris-

tics, either physiologic ones, such as: fingerprint, iris and retina recognition, face geometry, etc., or behavioral ones, such as user's typing or signature dynamic, etc. Among them, probably the most popular and one of the most reliable methods is fingerprint recognition, which has legal validity worldwide.

In this contribution we present the kernel implementation of a Mobile Certification Authority (MCA). Our MCA kernel is able to issue digital certificates complying with the X.509v3 standard; it supports either RSA or ECDSA as a public key cryptosystem engine and; it can incorporate biometric-based user identification information (in the form of fingerprint recognition) to the digital certificate. The MCA application was entirely written in C++ using the *iPAQ Pocket PC h5550* Personal Digital Assistant (PDA) as the target device platform. The iPAQ h5550 PDA is powered by a 400 MHz Intel XScale processor, with 128 MB RAM memory and 48 MB ROM memory. Furthermore, this PDA device is equipped with a biometric fingerprint scanner able to recognize users' fingerprints with high precision.

The rest of this paper is organized as follows. In Section 2 a brief description of the mandatory data structure of an X.509v3 certificate is given. Then, in Section 3 application's main architecture and its most prominent modules are explained. Section 4 gives an outline of the process followed by the MCA application in order to generate/verify X.509v3-compatible digital certificates. Finally, in Section 5 some conclusion remarks are drawn.

2 X.509v3 Certificates

The RFC2380 (*Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List Profile*) standard [5] defines the precise structure and format that a digital certificate complying with the X.903v3 standard must have. Essentially, an X.509 certificate is composed of three main structures: TBS certificate (*TBSCertificate*), algorithm identifier (*signatureAlgorithm*) and digital signature (*signatureValue*). The TBS certificate and algorithm identifier consists of ten common fields, six of them mandatory and four optional. The six mandatory fields are: serial number, signature's algorithm identifier, Certification Authority's name, period of validity, entity's public key, and entity's name. The four optional fields are: version number, Certification Authority's and user's unique identifier plus the extension fields [1].

Additionally, an X.509v3 certificate must be formatted according to the (*Abstract Syntax Notation One*) ASN.1 language [6]. ASN.1 is a standard norm for data representation used by a high number of applications and devices in the IT sector. This standard is intended for allowing the normalization and compression of data among different platforms in a smooth and transparent way.

We developed an ASN.1 language subset library as is specified in [5]. Our library implements all coding and decoding processes required for the generation/parsing of digital certificates. It consists of two main modules:

2.0.1 ASN.1 Encoder

This module encodes the digital certificate fields (such as final entity's name, validity period, public key, algorithm identifier, etc.) according to the ASN.1 format, supporting both, RSA and ECDSA public key cryptographic engines. It was developed based on an *open source* compiler described in [1].

2.0.2 ASN.1 Decoder

This module allows the parsing of previously generated certificates. Its implementation is based on the compiler reported in [4]. It supports both, RSA and ECDSA public key cryptographic engines.

In the next Section we explain MCA system architecture designed for generating/verifying X.509v3 digital certificates.

3 System Architecture Description

Figure 1 shows the block diagram of the MCA kernel. The main modules of our architecture are explained below.

As it has been mentioned, our MCA application can support both RSA and ECDSA as public key cryptographic schemes needed for the creation and verification of digital signatures. The RSA cryptosystem included in our library complies with the PCKS #1 [8] standard, while the ECDSA module complies with the IEEE P1363 standard [7].

For security reasons, we used a 128-bit AES cryptosystem for protecting the security of the mobile certification authority's private keys, using a 128-bit *master key*. That master key is generated by means of the MD5 hash value corresponding to the pass-phrase given by the mobile certification authority.

3.0.3 BioAPI Biometric Library

The iPAQ h5550 digitally reads user's fingerprints by using a thermal sensor and maps the measured analogical temperature with a digital image. That thermal

[1] The optional fields are only included in X.509 v2 and X.509 v3 certificates

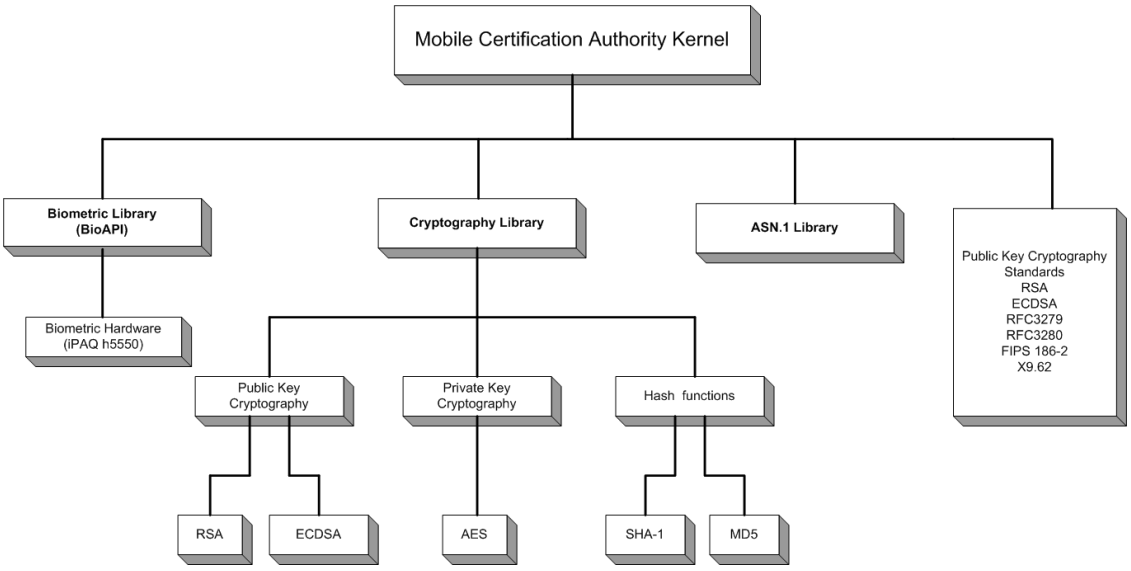

Figure 1: Main Architecture of the mobile certification authority kernel.

sensor detects the temperature differences between the digital fingerprints edges and valleys. [2] We used the Biometric Authentication Technology library, which is based on the BioAPI standard [2], as a software interface for handling the fingerprint scanner. The main BioAPI primitives utilized by our application are:

- **`Capture`**, commands a new digital fingerprint biometric reading.
- **`Verification`**, it is applied when a user claims that his/her fingerprints have been already registered in the application database. The library compares the user's biometric data against the stored record in its database (one-to-one comparison).
- **`Identification`**, also called searching or recognition, it is executed if the user's identity is unknown. The library compares user's biometric data against all the records in its database (one-to-all comparison).
- **`Storage`**, it allows an efficient way to keep digital fingerprints information. Stored data is not kept as an image. Instead, the image is compressed producing a 1024 bytes data, which is sufficient for representing the approximately 200-KByte original image.

The *error window* fingerprint can be set during the configuration of the authentication system. It is possible to select one out of three levels: regular, high and extra-high. For the three different security levels, the iPAQ h5540 authentication system performance is shown in Table 1.

[2]Thermal sensors show an important advantage with respect to other sensor types because they are almost impossible to deceive [2].

4 Generation and verification of an X.509v3 Digital Certificate.

In this Section we give the main steps followed by our application in order to generate and verify digital certificates complying with the X.509v3 standard.

4.1 Digital Certificate Generation Process

The generation process of a X.509v3 certificate consists of the following steps. First, a user must fill in a form provided by the application Graphic User Interface (GUI). That form asks for user's personal information such as name, e-mail, company, etc. Then, the application assigns an arbitrary validity period for the digital certificate (currently set to one year). In the event that the user had required an X.509v3 certificate with biometric information, then user's fingerprints are captured and stored using PDA's integrated fingerprint scanner and the BioAPI library. Thereafter, the user must select the digital signature algorithm to be utilized. The two possible choices are: RSA or ECDSA. If ECDSA is chosen, then, additionally, an elliptic curve must be selected (currently, our application supports NIST recommended [11] 163K, 192P, 224P and 233K elliptic curves). At that moment, the MCA kernel generates user's public/private key pair followed by his/her TBS certificate as is shown in Fig. 2. Finally, by using our

Table 1: iPAQ Pocket PC - False positives and False negatives rates

Security level	False Negatives (FRR)	False Positives (FAR)	Average of attempts
Regular	0.2331 % (1 of 429)	0.0010 % (41 of 4054737)	1.081967
High	0.4662 % (2 of 429)	0.0001 % (4 of 4054737)	1.107573
Extra-High	0.6993 % (3 of 429)	0.0000 % (0 of 4054737)	1.153226

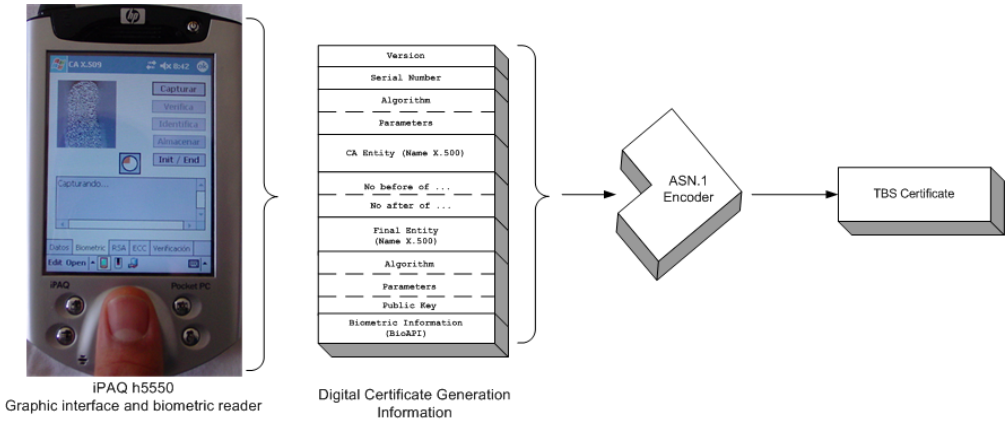

Figure 2: TBS certificate generation.

ASN.1 encoder module, the three main structures of the X.509v3 certificate are put together as shown in Fig. 3.

4.2 Digital Certificate Verification Process

In order to verify the correctness of a X.509v3 certificate our application proceeds as follows. First the certificate is parsed using our ASN.1 decoder module as is shown in Fig. 4. If everything is correct the three certificate's main components, namely, *TBSCertificate, signatureAlgorithm, signatureValue*, should be obtained. Once that the *TBSCertificate* component has been identified, a SHA-1 hash function is applied to it in order to execute the customary digital verification process shown in Fig. 5.

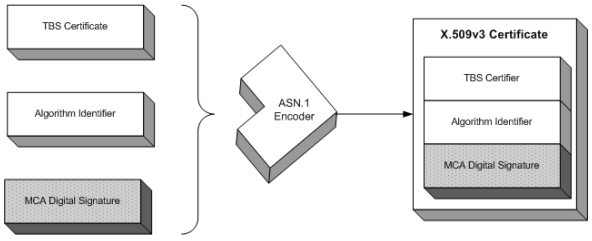

Figure 3: X.509v3 Certificate Generation.

It is noticed that we validate the correctness of the digital signature only. A more realistic application should also verify the certificate's validity period along with the Certificate Revocation List (CRL) issued by the CA [5].

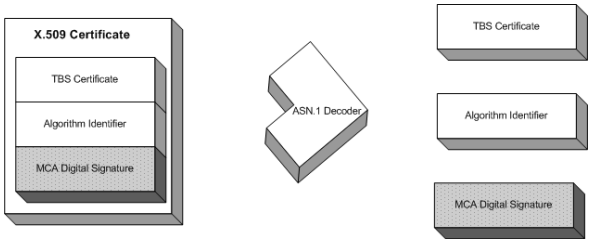

Figure 4: Parsing of an X.509v3 certificate.

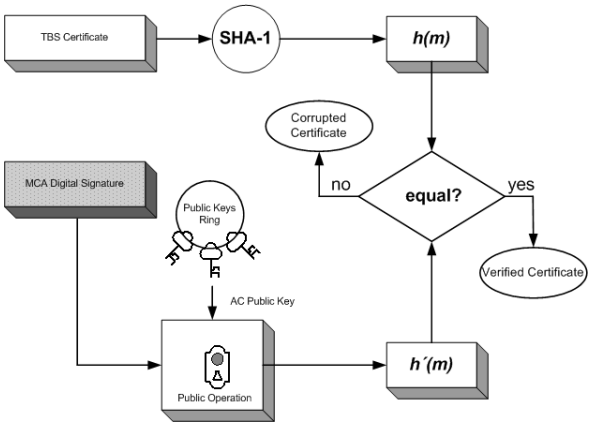

Figure 5: Verification of an X.509v3 Certificate.

5 Conclusion

Figure 6 shows the RSA and ECDSA certificates sizes (in bytes), with and without fingerprint biometric information. Notice that fingerprint biometric information

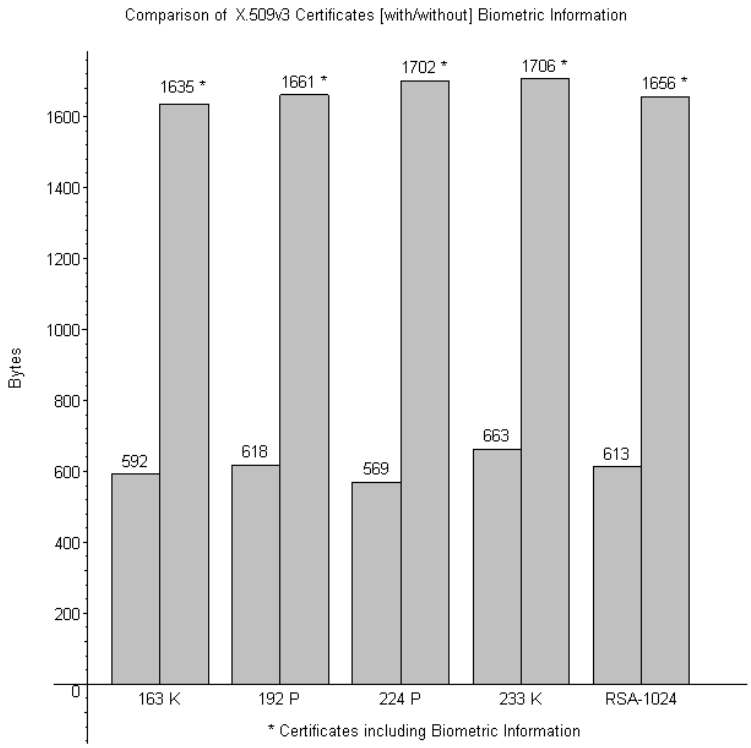

Figure 6: Certificates sizes comparison with and without biometric information.

increases the size of all certificates considered by about 1K byte. However, if required, this overhead value can be further reduced by compressing the fingerprint image information even more.

It results interesting to compare the certificate sizes generated by our application when using 1024-bit RSA against the ones that utilized ECDSA with NIST recommended curves 163K, 192P, 224P and 233K. It is observed that the size difference between the RSA-based and ECDSA-based digital certificates is surprisingly small. The shortest certificate was obtained when using NIST 163K curve, while the 1024-bit RSA certificate ranked second, with a slight difference in size of only 21 bytes. All the other ECDSA certificates showed a greater size than their 1024-bit RSA counterpart.

The reason why most ECDSA certificates have a greater size than the 1024-bit RSA certificate is due to the fact that the ECDSA standard mandates that all the elliptic curve domain parameters should be embedded into the digital certificate (i.e., elliptic curve a and b coefficients, order, cofactor, field type, polynomial generator, etc.).

According to already reported results, ECDSA is more efficient and quicker than RSA when using equivalent security levels [10]. This feature was once more confirmed in our experiments. Therefore, we conclude that ECDSA-based digital certificates should be preferred when operating in restricted environments.

Concretely, when working with constrained computational environments and/or wireless applications, the NIST-163K-ECDSA is the ideal selection. On the other hand, the NIST 163K curve offers less security level than the 192P, 224P, and 233K curves.

References

[1] The opensource ASN.1 compiler, 2004. Available at: http://lionet.info/asn1c.

[2] The BioAPI Consortium. BioAPI specification version 1.00, March 2000. Available at: http://www.bioapi.org.

[3] Internet Engineer Task Force. Public-key infrastructure X.509 PKIX, 2001. http://www.ietf.org/html.charters/pkix-charter.html.

[4] P. Gutmann. ASN.1/cryptlib, October 1997. Based on a program by David Kemp, available at: http://www.cs.auckland.ac.nz/ pgut001/dumpasn1.c.

[5] R. Housley, W. Polk, W. Ford, and D. Solo. *RFC3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List*

(CRL) Profile. IETF, April 2002. Available at: http://www.ietf.org/rfc/rfc3280.txt.

[6] ISO/IEC. Abstract syntax notation one ASN.1: Specification of basic notation, 2002. ITU-T Rec. X.680 (2002) ,ISO/IEC 8824-1:2002.

[7] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm ECDSA, 2001. Available at: http://www.certicom.com.

[8] B. Kaliski and J. Staddon. *RFC2437: PKCS #1: RSA Encryption*. RSA, October 1998. Available at: http://www.ietf.org/rfc/rfc2437.txt.

[9] Richard Kuhn, Vincent Hu, Timothy Polk, and Shu-Jen Chang. Introduction to public key technology and the federal PKI infrastructure. *NIST*, February 2001.

[10] A. Levi and E. Savas. Performance evaluation of public-key cryptosystem operations in WTLS protocol. In *(ISCC'03)*, pages 1245–1250. IEEE Computer Society, 2003.

[11] National Institute of Standrads and Technology. Recomended elliptic curves for federal government use, 1997.

[12] K. Schmeh. *Cryptography and Public Key Infrastructure on the Internet*. John Wiley & Sons, 2003.