AN ABSTRACT OF THE THESIS OF

Murat Aydos for the degree of Doctor of Philosophy in
Electrical & Computer Engineering presented on June 13, 2000.

Title: Efficient Wireless Security Protocols
based on Elliptic Curve Cryptography

Redacted for Privacy

Abstract approved: _____

Çetin K. Koç

In recent years, the elliptic curve cryptosystems (ECC) have received attention due to their increased security with smaller key size which brings the advantage of less storage area and less bandwidth. Elliptic curve cryptography provides a methodology for obtaining high-speed, efficient, and scalable implementations of network security protocols. In addition low power consumption and code size reductions are the other benefits of the ECC-based security architectures.

In this thesis, we mainly concentrate on public key authentication and key agreement protocols. After discussing several well-known protocols, we propose an authentication and key agreement protocol for wireless communication based on the elliptic curve cryptographic techniques. The proposed protocol requires significantly less bandwidth than the Aziz-Diffie and Beller-Chang-Yacobi protocols, and furthermore, it has lower computational burden and storage requirements on the user side.

Additionally, we present an end-to-end mobile user security protocol. The protocol is an improved version of the previous one in terms of security and interoperability. The achievement on the protocol goals and the complete security analysis are also given in this thesis. The proposed protocols overcome the known security flaws in existing private and public key protocols, improve the resiliency and interoperability using carefully selected design methods.

The rest of the thesis deals with the fast and efficient implementation of the protocols on different platforms. We first present the performance timings of the field and elliptic curve operations used in the proposed protocols on a Pentium processor. Then, we report a practical software implementation of cryptographic library which supports variable length elliptic curve digital signature algorithm for both signature generation and signature verification. We also present the real-time authenticated call-setup timings for the proposed ECC based protocols. The ECC library with variable key length was successfully implemented in an acceptable code size on a 32-bit ARM microprocessor.

Efficient Wireless Security Protocols based on Elliptic Curve Cryptography

by

Murat Aydos

A THESIS submitted

to

Oregon State University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

Completed June 13, 2000

Commencement June 2001

Doctor of Philosophy thesis of Murat Aydos presented on June 13, 2000

APPROVED:

Redacted for Privacy

Major Professor, representing Electrical & Computer Engineering

Redacted for Privacy

Head of Electrical & Computer Engineering Department

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Murat Aydos, Author

ACKNOWLEDGMENTS

I thank my advisor Çetin K. Koç for his reviews, help, guidance and encouragement in producing the papers we published, which formed a basis for this thesis.

I also thank my committee members for their efforts and patience in going through my work.

At various stages in the writing of this thesis, a number of people have given me invaluable technical help, advise and comments on the practical implementation of the security protocols. In this regard I owe a debt of gratitude to the Information Security Lab researchers. In particular, I acknowledge Erkay Savaş, Tugrul Yanık, Serdar Erdem for providing the software cryptographic libraries on which the timing performances of the protocols proposed in this thesis have been obtained. I also would like to thank Mehmet Musa and F. Rodríguez-Henríquez for giving me valuable feedback during the preparation of the thesis.

Finally, I acknowledge financial support for my Ph.D. education from Pamukkale University, Denizli/Turkey.

Murat Aydos

Corvallis, Oregon

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

LIST OF TABLES

To my wife...

for her endless love and support

Efficient Wireless Security Protocols based on Elliptic Curve Cryptography

Chapter 1
INTRODUCTION

"Coding Theorist's Pledge: I swear by Galois that I will be true to the noble traditions of coding theory; that I will speak of it in the secret language known only to my fellow initiates; and that I will vigilantly guard the sacred theory from those who would profane it by practical applications"

J. L. Massey

1.1 Summary of the Research

In the past several years there has been an overwhelming demand for new communication technologies that has prompted new security questions. There is a huge amount of intensive study in literature in this area and many public key cryptographic techniques have been developed. In recent years elliptic curve cryptosystems (ECC) have gained a lot of respect due to their increased security with smaller key size, which brings the advantage of less storage area and less bandwidth. Elliptic curve cryptography provides a methodology for obtaining high-speed, efficient, and scalable implementations of network security protocols. In addition low power consumption and code size reductions are the other benefits of ECC based security architectures. All these features of ECC make it very useful in many application areas such as wireless transactions, ATMs, cellular and PCS phones, storage of medical records, electronic cash, handheld computing, broadcast and smart card applications.

In this report we first present a brief introduction to cryptography. Private key cryptography and public key cryptography are described shortly. A general overview of Public Key Infrastructure (PKI) is given. Then we summarize the existing public key cryptosystems: Integer factorization systems (RSA), discrete logarithm systems

(El-Gamal, DSA) and finally elliptic curve cryptosystems that are based on elliptic curve discrete logarithm problem.

After this brief introduction to cryptography, we overview and analyze some of the existing private key and public key based authentication and key agreement protocols. Among them are Beller-Chang-Yacobi, Aziz-Diffie and Molva-Samfat-Tsudik protocols. A brief description of each protocol along with its advantages and disadvantages is given.

In next step, we propose an authentication and key agreement protocol for wireless communication based on elliptic curve cryptographic techniques. The proposed protocol requires significantly less bandwidth than the Aziz-Diffie and Beller-Chang-Yacobi protocols, and furthermore, it has lower computational burden and storage requirements on the user side. The use of elliptic curve cryptographic techniques provide greater security using fewer bits, resulting in a protocol which requires low computational overhead, and thus, making it suitable for wireless and mobile communication systems, including smartcards and handheld devices as mentioned before.

Next we propose an updated version of the previous protocol. End-to-end user authentication and key agreement are covered in this protocol. Analysis of the protocol's achievement on the proposed goals and security concerns are also addressed in this section.

We then concentrate on obtaining fast performance timing results of the arithmetic operations used in elliptic curve digital signature algorithms (ECDSA). These include field operations such as multiplication, squaring and inversion on the underlying field (in our case $GF(2^k)$) and elliptic curve operations such as addition, doubling and multiplication.

In our second implementation we give implementation timing results for ECC over a field of prime characteristic on a 32-bit ARM microprocessor (80 MHz). We report a practical software implementation of cryptographic library which supports variable length elliptic curve digital signature algorithm for both signature generation and signature verification. We also present the real-time authenticated call set-up timing for recently proposed wireless authentication protocol based on ECC. The

timings were also obtained for Secure Hash Algorithm (SHA) and Data Encryption Standard (DES) algorithm. The ECC library with variable key length, SHA and DES were successfully implemented in an acceptable code size.

1.2 Motivation

In personal communication systems, open access to the radio exposes the context of communication over the wireless link between a mobile unit and the wired network. Such openness also gives an intruder the opportunity to masquerade as a legitimate subscriber to make free calls. To provide proper protection on this wireless link, security features, such as confidentiality and fraud control, need to be provided. In general, these features can be achieved through authentication protocols that verify the identities of entities on both ends of the wireless link and establish a secret session key between them for the following voice or data communications.

The nature of wireless devices prevent designers to apply similar available security features that are already in use for wired networks. For example, considerations on hardware complexity, battery power, and available memory space have prevented a mobile unit from performing computations that require expensive hardware. While some of the security features are already provided in wireless communication systems, they need to be further developed and enhanced. The requirement that the user must also authenticate the network in order to prevent an intruder from masquerading as a network operator or service provider is motivated by the observation that a user may want to make sure that he is connected to the network of an operator whom he trusts. This becomes increasingly important as the number and variety of competing public and private network operators and service providers grow larger.

Public key cryptography has not previously been used in mobile communication environments due to performance constraints. It was not deemed suitable for second generation systems because of the resulting length of messages and the necessary computational loads. To overcome these problems we have developed new protocols for authentication between user and network. The protocols are based on Elliptic

Curve Cryptographic algorithms. Our research is basically concentrated around these protocols and the implementation issues in constrained environments.

Chapters 2 through 8 provide the background necessary to understand the motivation and the contents of the contributions made, which are presented in Chapter 9. A brief description of each chapter is as follows:

Chapter 2 provides a brief introduction to cryptography and to the terminology commonly used. It also explores different types of cryptographic systems commonly used, and summarizes the basic ideas on which these systems are built. Strong and weak points of the systems are also presented to compare the level of security they provide.

Chapter 3 summarizes some of the well known authentication and key agreement protocols. Features and protocol flows of the protocols are given. These protocols have critical bandwidth and storage problems that are not desirable in constrained environments such as in mobile networks. The ones that are based on private key techniques are more efficient in terms of bandwidth and storage than the public key ones, however they bring the key management and key distribution problems to the servers and the users.

Chapter 4 presents the first ECC based mutual authentication and key agreement protocol for wireless communication. The storage and bandwidth requirements of the protocol are also given. This protocol combines the advantages of both secret key and public key techniques. The special use of ECC presents significant advantages in bandwidth and storage use.

In chapter 5, the previous idea of providing mutual authentication and key agreement between users and servers is extended to end-to-end user mutual authentication and negotiated key agreement. In this section, multiple use of certificates, crossing domain boundries and mobile cloning issues are also addressed. A complete security analysis is given.

In chapter 6, we concentrate on obtaining fast performance timing results of the arithmetic operations used in elliptic curve digital signature algorithms (ECDSA). These include field operations such as multiplication, squaring and inversion on the

underlying field (in our case $GF(2^k)$) and elliptic curve operations such as addition, doubling and multiplication.

In Chapters 7 and 8, we present the execution timings of the proposed protocols obtained on 32-bit ARM microprocessor which is used to simulate the protocol in order to obtain the performance characteristics of the proposed protocol. ARM7TDMI core was selected from the ARM family due to its suitable features for wireless and handheld devices. A general Elliptic Curve software library was constructed and then compiled on the ARM software development toolkit. The timing values were obtained for necessary cryptographic operations performed during the on-line protocol execution such as point multiplication on the curve, signature generation/verification and secret key encryption/decryption. The timing results are given in the tables. The significance of this part of the research is that the software library for ECC is scalable. This means that required key size increase for greater security can be provided with the current architecture if needed.

Chapter 9 concludes the report with the summary of the results, contributions and discussions.

Chapter 2
CRYPTOGRAPHIC SYSTEMS

"Only two things are infinite, the universe and the human stupidity, and I am not sure about the former."

A. Einstein

2.1 Introduction

The need for information security in today's digital systems is growing. For this reason cryptography has become one of these systems' critical component. Cryptographic services are now required across a variety of electronic platforms such as secure access to private networks, data banks, electronic commerce, cellular and PCS phones, all kinds of data communications and smart card technology. A well-defined and implemented cryptographic system should provide the following services:

- Confidentiality: Keeps the data involved in an electronic transaction private. Meaning that the transmitted information is accessible only for reading by authorized parties. Encryption provides confidentiality.

- Authentication: Ensures that the origin of a message or electronic document is correctly identified. In mutual authentication both the server and user or both parties in general case authenticate each other. It is typically provided by verifying other parties digitally signed certificates.

- Data Integrity: It basically means that the information exchanged in an electronic data transfer is not alterable without detection. Modification types include writing, changing, deleting, etc.

- Nonrepudiation: This simply tells that the actions performed by the service user in an electronic transaction are nonrevocable so that they are legally binding. Therefore, neither the sender nor the receiver of a message should be able to deny the transaction.

The fundamental goal of cryptography has historically been to achieve privacy, i.e., to enable two people to send each other messages over an insecure channel in such a way that only the intended recipient can read (or understand) the message. This objective has traditionally been met by using private key cryptosystems. As will be explained in the following sections this system has some disadvantages that make it unsuitable for use in certain applications. Public key cryptosystems overcome the key distribution and management problems inherent with private key systems. And elliptic curves offer more security using smaller bandwidth, which make them important.

2.1.1 The Definition of Cryptography

Cryptography is , in general, the science of concealing data. The term has come to mean something more precise. There are at least three approaches to concealing data:

- A *Code* is the substitution of a phrase for another phrase, so that "The wolf comes in night" could mean "attack on the next signal". Codes are inflexible since only those upon which both the sender and the receiver have agreed can be transmitted.
- *Steganography* is the concealment of one message within another, for example as the least-significant bits of a 32-bit bitmap file.
- *Ciphers* are the substitution of one block of text by another according to some generally-applicable rule.

When we use the term *cryptography*, we usually mean the study and use of ciphers.

2.1.2 Ciphers

Ciphers work in various ways, but well-defined ciphers should have the following things in common:

- Their input and output are treated as byte streams.
- To *encrypt* the data, a random key is used with the cipher.
- To *decrypt* the data, the cipher is used with a key.

Ciphers are divided into *asymmetric* and *symmetric* ciphers on the basis of whether they use the same key to encrypt and decrypt or two different keys for the processes. The first method corresponds to symmetric ciphers and the latter one corresponds to asymmetric ciphers. In asymmetric ciphers, the encryption key is also called as the *public key*, and the decryption key as the *private key*.

Symmetric ciphers have the advantage of speed; they are frequently 100-1000 times faster than typical asymmetric ciphers in software. While encrypting files for local storage the obvious choice is the symmetric ciphers. However, a solution should be found for secure transmission of the key if a message were to be sent to the receiver in encrypted form. Well-known symmetric ciphers include DES, RC2, RC4 and IDEA. The strength of a symmetric cipher is roughly measured by its key length. A 40-bit key is considered to be weak, whereas a 128-bit key is strong.

2.1.3 Authenticity

Encrypted messages give *confidentiality* ensuring that no unauthorized person can see the details of an transaction across the channel. Secure Socket Layer (SSL) is used to encrypt all the data communicated over the web.

But confidentiality is only half of the story. In order to make electronic commerce or any important electronic transactions safe, the origin of the message received should be determined correctly. *Tamperproofing* and *authenticity* are provided by *digital signatures*. A simple signature generation and verification can be shown as follows:

2.1.3.1 *Signing a Message*

1. Hash the message M to get its hash value, H.
2. Encrypt H with the sender's private key to get the signature S.
3. Send M, S.

2.1.3.2 *Verifying a Signed Message*

1. Receive the message M, S and separate them.
2. Hash the message M to get its hash value, H'.
3. Decrypt S with the sender's public key to get the decrypted hash, H''.
4. Compare H' and H''. If they are the same, the message has verified correctly and can be trusted. Otherwise it has been tampered with in transit or the sender is not who he/she claims to be. In either case, the signature should be rejected.

The replay attack is simple to implement, and simple to avoid. A precise time, a serial number or some other unique identifier should be included in the signed message. This type of information should be in the signed data, not appended to it after signing.

2.1.4 The Need For Certificates

In some cases, an attacker-we call her Eve- can put a public key in a database next to Bob's name, and then Alice might send her encrypted messages that were intended for Bob. Furthermore, if she used her private key to create a digital signature on (for example) a payment authorization, someone else could use the public key from the database to check the signature, verify that it decrypted correctly and authorize the payment on Bob's behalf.

So the picture is more complicated. Now, to trust a signature, two things must be true:

- The signature must verify correctly.
- One must be sure that the public key belongs to the right person.

Digital certificates provide a solution for this problem. A *digital certificate* is like an electronic ID. It contains the following information:

- Personal information such as name, company name, e-mail address
- An expiration date
- Public key
- A unique serial number, which is the fastest way to reference it in a database
- A digital signature to guarantee authenticity

Some advance level certificates such as SPKI [18] certificates also require some additional information such as *rights* and *redelegation* sections in order to provide *authorization.*

These certificates are signed by *Certification Authorities* (CAs). A CA is a *Trusted Third Party* which receives the public key and associated personal information, then verifies the identity by some out-of-band method (such as phoning or meeting in person), and finally signs the details. CAs have their own certificates that can be obtained to verify their signatures on the issued certificates. In some applications, there is a *chain of certificates*, which means that each certificate is signed by some other CAs. Each CA's certificate is attached to another. While verifying a certificate, each of the attached certificates are also verified leading back to a *root CA*. Verisign is an example of a company providing such certificates.

2.1.5 Public Key Infrastructure

The widespread adoption of these cryptographic techniques will require a global PKI, based on a strictly defined standards that control every aspect of a certificate's life cycle. *PKIX*, Public Key Infrastructure for X.509 [4] Certificates, Working Group defines a PKI as "The set of hardware, software, people and procedures needed to create, manage, store, distribute and revoke certificates based on public-key cryptography".

A PKI comprises five types of components:

- CAs: These are responsible for issuing and revoking certificates.
- Registration Authorities (RAs): These verify the binding between public keys and the identities of their holders.
- Certificate holders (or subjects): People, machines or software agents that have been issued with certificates and can use them to sign digital documents.
- Clients: These validate digital signatures and their certification paths from a known public key of a trusted CA.
- Repositories: These store and make available certificates and certificate revocation lists (CRLs).

A PKI must carry out the following functions:

- **Registration:** This is the process in which a prospective certificate holder presents itself to the CA in order to request a certificate. The subject will be required to supply attributes such as a common name, fully qualified domain name, IP address and others as specified in the CA's certification policy statement. The CA will then follow the guidelines in this statement to verify that the details supplied by the subject are correct, before issuing a certificate.
- **Certification:** Certification is the process in which the CA actually issues a certificate that contains the subject's public key, delivers this certificate to the subject and publishes it in a suitable public repository.

- **Key pair recovery:** Some PKI implementations will require that all key-exchange and/or encryption keys are backed up in a secure store. This means that they are recoverable if the subject loses the key.
- **Key generation:** In some cases, the subject will generate a key pair in his local environment, before passing the public key to the CA for certification. If the CA is responsible for key generation, the keys are normally supplied to the subject as an encrypted file or on a physical token such as a smartcard.
- **Key update:** All key pairs and their associated certificates, should be updated at requlars intervals. There are two scenarios that will require the replacement of a key pair.
  - The date on the certificate expires.
  - The private key of one of the PKI entities is compromised.
- **Cross-certification:** This allows users in one administrative domain to trust certificates issued by a CA operating in a different domain.
- **Revocation:** In most cases, a certificate remains valid until its validity period expires. There are some cases that will require the early revocation of a certificate's validity. These include:
  - The subject changes his/her/its name.
  - An employee leaves a company that issued a certificate.
  - A compromise or a suspected compromise of the private key corresponding to the public key in the certificate occurs.

2.2 Private Key Cryptography

Let $\mathcal{M}$ denote the set of all possible plaintext messages, $\mathcal{C}$ the set of all possible ciphertext messages and $\mathcal{K}$ the set of all possible keys.

A *private key cryptosystem* consists of a family of pairs of functions

$$E_k : \mathcal{M} \longrightarrow \mathcal{C} \quad \text{and} \quad D_k : \mathcal{C} \longrightarrow \mathcal{M} \qquad k \in \mathcal{K} \ ,$$

such that

$$D_k(E_k(m)) = m \qquad \text{for all } m \in \mathcal{M} \text{ and } k \in \mathcal{K} \ .$$

To use such a system, the two users must initially agree upon a secret key $k \in \mathcal{K}$. They can do this by physically meeting, but this can be impractical or sometimes even impossible, or they might use the services of a trusted courier. Besides the problem of availability, the secureness of the courier is still questionable.
The main disadvantages of the private key cryptosystems are:

1. Key distribution problem (a secure channel may not be available)
2. Key management problem (if the number of pairs (of users) is large then the number of keys becomes unmanageable)
3. No signatures possible

A *digital signature* is an electronic analogue of a hand-written signature that allows a receiver to convince a third party that the message is in fact originated from the sender.

2.3 Public Key Cryptography

This section presents a brief introduction to some of the concepts and techniques involved in public-key cryptography. Today's cryptographic techniques are mostly based on public key cryptosystems due to their implementation efficiency and easier key management protocols.

Since the invention of public-key cryptography in 1976 by Whitfield Diffie and Martin Hellman, numerous public-key cryptographic systems have been proposed. All of these systems rely on the difficulty of a mathematical problem for their security. Over the years, many of the proposed public-key cryptographic systems have been broken, and many others have been demonstrated to be impractical. Today, only

three types of systems should be considered both secure and efficient. Examples of such systems, classified according to the mathematical problem on which they are based, are:

1. Integer factorization problem (IFP): RSA and Rabin-Williams.
2. Discrete logarithm problem (DLP): The US government's Digital Signature Algorithm (DSA), Diffie-Hellman key exchange, ElGamal encryption and signature schemes.
3. Elliptic curve discrete logarithm problem (ECDLP): The elliptic curve analogues of the DLP systems.

Although years of intensive study by leading mathematicians and computer scientists have not proven these problems intractibility, they are believed to be intractable because all the effort to solve them has not produced efficient algorithms to be implemented. As this kind of effort with today's enormous computer power is expanded over time in studying these problems, the confidence in the security of the related cryptographic systems will continue to grow.

Various evaluation criteria may be relevant to the application at hand when comparing public-key cryptosystems, including:

- Security (hardness of the underlying math problem)
- Key lengths
- Signature sizes
- Speed
- Suitability for implementation in hardware, software, and firmware
- Complexity of implementation (code size, gate count, power consumption, etc.)
- Storage requirements
- Industry/government standards

- Patent coverage
- Licensing terms

The best algorithms known for the IFP and DLP take subexponential time. In contrast, the best algorithms known for the ECDLP take fully exponential time. This means that, as the problem size increases, the algorithms for solving the ECDLP become infeasible much more rapidly than the algorithms for the IFP and DLP. For this reason, elliptic curve cryptosystems (ECC) offer equivalent security to systems such as RSA and DSA, while using far smaller key sizes. This relative attractiveness of ECC will continue to increase as computing power improvements force a general increase in key sizes.

The benefits of this higher strength-per-bit include higher speeds, lower power consumption, bandwidth savings, and smaller certificates. These advantages are particularly beneficial in applications where bandwidth, processing capacity, power availability, or storage are constrained (including chip cards, electronic commerce, web servers, cellular telephone, and pagers).

2.3.1 Diffie-Hellman Key Exchange

In 1976, W. Diffie and M. Hellman [24] invented public key cryptography to address the deficiencies in private key cryptography, stated in the previous section. Their protocol is known as *Diffie-Hellman key exchange.* And in terms of an arbitrary group it can be described as:

1. (Setup) A and B publicly select a (multiplicatively written) finite group G and an element $\alpha \in G$.
2. A generates a random integer a, computes α^a in G, and transmits α^a to B over a public communications channel.
3. B generates a random integer b, computes α^b in G, and transmits α^b to A over a public communications channel.

4. A receives α^b and computes $(\alpha^b)^a$.

5. B receives α^a and computes $(\alpha^a)^b$.

A and B now share the common group element α^{ab}. Note that an eavesdropper knows G,α,α^a and α^b, and his task is to use this information to reconstruct α^{ab}. This problem is commonly referred to as *Diffie-Hellman problem.*

The problem of computing a, given G, α and α^a is called the *discrete logarithm problem.* It has not been proven but widely believed that the discrete logarithm problem and the Diffie-Hellman problem are computationally equivalent [55].

2.3.2 Trapdoor One-way Functions

Easy-hard : We will use the term *hard* to mean computationally infeasible, i.e., infeasible using the best known algorithms and best available technology. In terms of software engineering only, *easy* and *hard* will mean requiring polynomial and exponential time, respectively.

One-way function : An invertible function $f : \mathcal{M} \longrightarrow \mathcal{C}$, such that $\forall m \in \mathcal{M}$ it is *easy* to compute $f(c)$, but for most $c \in \mathcal{C}$ it is *hard* to compute $f^{-1}(m)$. There is no function that is proven to be a one-way function, but there are some candidates.

Trapdoor one-way function : A one-way function that can be efficiently inverted by using some extra information. The extra information is called the *trapdoor.*

A public key cryptosystem is constructed using a family $f_k : \mathcal{M} \longrightarrow \mathcal{C}$, $k \in \mathcal{K}$, of trapdoor one-way functions. The trapdoors $t(k)$ should be easy to obtain for all $k \in \mathcal{K}$. Also an efficient algorithm is needed to find f_k, but knowing it, should not lead one to any information that will make it feasible to recover k (and thus $t(k)$). If these conditions are satisfied then each user U selects a random key $u \in \mathcal{K}$ and publishes the algorithm A_u, his *public key*, for computing f_u. Then the trapdoor $t(u)$, which is used to invert f_u, will be the user's *private key.* Any message m is

encrypted to $f_u(m)$ using A_u, and only the user U can find out m from $f_u(m)$, as only *he* knows $t(u)$ to invert f_u. To use digital signatures we assume $\mathcal{M} = \mathcal{C}$ [55].

The group order and exponentiation are two commonly used trapdoor one-way functions. RSA and elliptic curves over the ring $\mathcal{Z}_n$ uses group order as a trapdoor one-way function, and ElGamal Cryptosystem uses exponentiation. Now we will further explain these systems.

Suppose that for the group G, multiplication is easy and finding its order is hard. Then we can construct a public key cryptosystem whose trapdoor one-way function is based on the difficulty of finding group order. Each user chooses a group G (with order n) and a random integer e such that $\gcd(e, n) = 1$ and computes (using the extended Euclidean Algorithm) an integer d, $1 \leq d \leq n-1$, such that

$$ed \cong 1 \pmod{n} \ .$$

A's public key consists of the group G and the integer e. To send the message m to A, one computes and sends m^e. A can recover m using d since $(m^e)^d = m$.

Exponentiation in a multiplicatively written finite group can be performed efficiently by repeated square-and-multiply method, if an efficient way to compute the product is known. Similarly, multiplication in an additively written finite group can be performed efficiently by repeated double-and-add method, if an efficient way to compute the addition is known.

2.3.3 RSA Cryptosystem

The RSA cryptosystem was invented in 1977 by Rivest, Shamir and Adleman [64] and was the first realization of Diffie-Hellman's abstract model for public key cryptography.

To set up this system each user picks two large primes p and q and computes their product $n = pq$. The group used is $G = \mathcal{Z}_n^*$. It is well known that the order of G is

$$\phi(n) = (p-1)(q-1) \ .$$

The public key is the pair of integers (n, e) and the private key is d. The problem of computing $\phi(n)$ using only n is computationally equivalent to the problem of factoring n, which is believed to be hard (but not proven). Thus the security of the RSA is based on the factoring problem.

2.3.4 ElGamal Cryptosystem

In 1985, T. ElGamal [27] proposed a public key scheme based on discrete exponentiation, which exhibits the properties of a trapdoor one-way function.

1. (Setup) A finite group G and an element $\alpha \in G$ are chosen. Each user picks a random integer l (the private key) and makes public α^l (the public key). We suppose that messages are elements of G and that user A wishes to send a message m to user B.

2. A generates a random integer k and computes α^k.

3. A looks up B's public key α^b, and computes $(\alpha^b)^k$ and $m\alpha^{bk}$.

4. A sends to B the pair of group elements $(\alpha^k, m\alpha^{bk})$.

5. B computes $(\alpha^k)^b$ and uses this to recover m.

Clearly, the security of the ElGamal cryptosystem and the Diffie-Hellman key exchange are equivalent, and hence the security of the ElGamal cryptosystem is based on the difficulty of the discrete logarithm problem. ElGamal [27] also designed a signature scheme, which makes use of the group G [55].

2.3.5 Elliptic Curve Cryptosystems

When an elliptic curve is defined over a finite field, the points on the curve form an Abelian group. The addition operation in this group is "easy" to implement, both in hardware and software. The discrete logarithm problem in this group is believed (but not proven) to be very "hard", in particular harder than the one defined in finite fields

of the same size. It was for this reason that the elliptic curves were first suggested in 1985 by N. Koblitz [43] and V. Miller [56] for implementing public key cryptosystems. Elliptic curves over finite fields can be used to implement the Diffie-Hellman key exchange, and the ElGamal cryptosystem. Also they can replace trapdoor one-way functions in any system. These systems potentially provide the same security as the existing public key cryptosystems, but uses shorter key lengths. Having shorter key lengths mean smaller bandwidth and memory requirements, which is a crucial factor in some applications such as design of smart cards, where both memory and processing power are limited. Another advantage of using the elliptic curves is that each user may select a different curve, even though the underlying field is the same for all. That means each user can change his curve periodically (for extra security) without changing the hardware [55].

2.3.5.1 *Elliptic Curve Diffie-Hellman*

This protocol establishes a shared key between two parties. The original Diffie-Hellman algorithm is based on the multiplicative group modulo p, while the elliptic curve Diffie-Hellman (ECDH) protocol is based on the additive elliptic curve group. We assume that the underlying field $GF(p)$ or $GF(2^k)$ is selected and the curve E with parameters a, b, and the base point P is set up. The order of the base point P is equal to n. The standards often suggest that we select an elliptic curve with prime order, and therefore, any element of the group would be selected and their order will be the prime number n. At the end of the protocol the communicating parties end up with the same value K which is a point on the curve. A part of this value can be used as a secret key to a secret-key encryption algorithm. Figure 2.1 shows the steps of the algorithm.

The second version provides a little more flexibility in the sense that the established value can be preselected by the user and sent to the server. The protocol steps shown in Figure 2.2 can be modified slightly for sending of a secret value from the server to the user.

Figure 2.1. Elliptic Curve Diffie-Hellmann (Version 1.)

USER		SERVER
• Choose $d_u \in [2, n-2]$		• Choose $d_s \in [2, n-2]$
• $Q_u = d_u \times P$		• $Q_s = d_s \times P$
• Send	$\xrightarrow{Q_u}$	• Receive
• Receive	$\xleftarrow{Q_s}$	• Send
• $K = d_u \times Q_s = d_u d_s \times P$		• $K = d_s \times Q_u = d_s d_u \times P$

Figure 2.2. Elliptic Curve Diffie-Hellmann (Version 2.)

USER		SERVER
• Choose $d_u \in [2, n-2]$		• Choose $d_s \in [2, n-2]$
• $e_u = d_u^{-1} \bmod n$		• $e_s = d_s^{-1} \bmod n$
• $Q = d_u \times K$		
• Send	$\xrightarrow{Q}$	• Receive
		• $R = d_s \times Q = d_s d_u \times K$
• Receive	$\xleftarrow{R}$	• Send
• $S = e_u \times R = e_u d_s d_u \times K$		
$= d_s \times K$		
• Send	$\xrightarrow{S}$	• Receive
		• $T = e_s \times S = e_s d_s \times K = K$

Chapter 3
AUTHENTICATION AND KEY AGREEMENT PROTOCOLS

"Because we can engineer genetics, because we can telecast real lives–of course we must, right? But are these good things to do? The irony is, the people who will finally answer that question will be the very ones produced by the process. "

Roger Ebert

3.1 Literature Review

In this chapter we briefly describe some of the well known cryptographic protocols designed for mobile networks. These protocols are either based on public key or private key techniques.

- **The Protocol by Molva, Samfat and Tsudik.**

  The protocol by Molva, Samfat and Tsudik [62] is a private-key based authentication protocol. The design is based on top of existing two- and three- party authentication and key distribution protocols. These protocols are borrowed from *KryptoKnight*, which is an authentication and key distribution service developed at IBM Research. Figure 3.1 illustrates the two-party authentication system to authenticate mobile users in their home location area. The original paper also shows the temporary foreign domain residence authentication.

  K_{ab} is a secret key between the mobile users A and B. $Auth_{K_{ab}}$ is computed as a result of applying strong encryption function such as DES, IDEA or RC4 with the key K_{ab}. N_{ab} and N_{ba} are two nonces and B is the identity of the mobile B. $ACK_{K_{ab}}$ is computed by the initiating party to complete two-way authentication. Comparison of the received and calculated expressions completes the authentication phase.

Figure 3.1. Molva, Samfat and Tsudik Protocol Protocol Flow.

MOBILE A		MOBILE B
• Send	$A, N_{ab} \longrightarrow$	• Receive
• Receive	$Auth_{Kab}(N_{ab}, N_{ba}, B), N_{ba} \longleftarrow$	• Send
• Send	$ACK_{K_{ab}}(N_{ab}, N_{ba}, A) \longrightarrow$	• Receive

Similar to other private-key based protocols, this protocol also assumes a shared secret key between two parties. This assumption alone brings a big key management and key storage problem on both servers and users. It is not possible to implement digital signatures in this protocol, therefore non-repudiation is not provided. Furthermore user identity confidentiality is not provided.

- **An Authentication Protocol by Beller and Yacobi.**

  Beller and Yacobi [16] have also proposed a key agreement and authentication protocol. This protocol provides security comparable to the well known RSA public-key protocol, but with the orders of magnitude less on-line computation required on one side of the protocol. This advance can make public-key security economical for applications where one side of the interaction is a low-cost customer device, e.g. portable phones, home banking terminals, or smart cards.

  The protocol makes use of the asymmetric public-key building blocks of Rabin and ElGamal. Figure 3.2 illustrates the initializing process for the network server and the mobile user. The server is assigned the unique identity j by the CA. The server picks a Rabin secret key (p_j, q_j) and gives the corresponding public key N_j to the CA, which certifies it (using the modular square root operation to sign a hashing of the concatenation of j and N_j). The same process is performed for the user. The user chooses a random secret key S_i, and generates the associated public key P_i. The user acquires its certificate

Figure 3.2. Beller and Yacobi Protocol Initiations.

SERVER		CERTIFICATION AUTHORITY
• Choose p_j, q_j prime		
• Send $N_j = p_j \cdot q_j$	$\xrightarrow{N_j}$	• Receive
		• Pick unique identity j
• Receive	$\xleftarrow{j, c_j, \alpha, N_s, N_u}$	• Send $c_j \equiv \sqrt{h(j, N_j)} \bmod p_u \cdot q_u$
• Store $j, c_j, N_j, \alpha, N_s, N_u$		

TERMINAL		CERTIFICATION AUTHORITY
		• Pick unique identity j
• Receive	$\xleftarrow{i, \alpha, N_s, N_u}$	• Send
• Choose random S_i		
• $P_i \equiv \alpha^{S_i} mod N_s$		
• Send P_i	$\xrightarrow{P_i}$	• Receive
		• $c_i \equiv \sqrt{h(i, P_i)} \bmod p_u \cdot q_u$
• Receive	$\xleftarrow{c_i}$	• Send c_i
• Store $i, c_i, S_i, P_i, N_s, N_u$		

from the CA. Prior to each access request, the terminal chooses a random secret r, and performs the precomputations shown in Figure 3.3. The real-time portion of the protocol is basically presenting certificates and challenge numbers to the other party. The protocol is based on the fact that performing modular square root operation without knowing the prime factors of ElGamal public keys N_u and N_j is not feasible. Modular square operation is just a large modular multiplication operation.

This protocol is bandwidth and storage inefficient. In addition, the user has to perform many precomputations before making each call. The comparison values of this protocol to our protocols are given in the next chapter.

Figure 3.3. Beller and Yacobi Real-Time Protocol Execution.

TERMINAL I		SERVER J
Precomputation		
Terminal i :		
choose random r, compute and		
store $v \equiv \alpha^r \bmod N_s$,		
$r^{-1} \bmod (N_s - 1)$,		
$S_i \cdot v \bmod (N_s - 1)$		
Real-time execution		**Real-time execution**
TERMINAL I		SERVER J
• Receive	$[j, N_j, c_j] \longleftarrow$	• Send
• $h(j, N_j) \equiv {c_j}^2 mod N_u$?		
• Abort if not		
• Choose random x, s.t.		
• $x = (0^k, x_L, x_R)$		
• Send $y \equiv x^2 mod N_j$	$\longrightarrow [y]$	• Receive
		$x \equiv \sqrt{y} \bmod p_j \cdot q_j$
Encrypted with $key = x_R$		
• Receive	$[m, 0^k] \longleftarrow$	• Choose random m
• Continue if 0^k is present.		
• $w \equiv (m - S_i \cdot v) \cdot r^{-1}$		
$\bmod (N_s - 1)$	$[i, P_i, c_i, v, w] \longrightarrow$	• Receive
		$h(i, P_i) \equiv {c_i}^2 mod N_u$?
		• Abort if not
		$P_i^v \cdot v^w \equiv \alpha^m mod N_s$?
		• Abort if not

Figure 3.4. Aziz and Diffie Protocol Flow.

MOBILE		BASE
• Send	$C_A, N_A, list_1 \longrightarrow$	• Receive
• Receive	$C_B, K_A(R_B), list_2, K_B{}^{-1}[hash(K_A(R_B), list_2, N_A, list_1)] \longleftarrow$	• Send
• Send	$K_B(R_A), K_A{}^{-1}[hash(K_B(R_A), K_A(R_B))] \longrightarrow$	• Receive

- **The protocol by Aziz and Diffie.**

  Aziz and Diffie [12] proposed a public key protocol providing mutual authentication and key agreement. The protocol exchanges three messages over the channel and performs six encryptions and four decryptions. The mobile is not required to do precomputations before each call. The protocol flow is given in Figure 3.4 and can be summarized as follows (The initialization process is not shown.):

  Mobile A sends *Base B* its certificate, C_A, a nonce, N_A, and a list of shared key algorithms, $list_1$, to choose from. *Base B* validates the certificate by concealing the contents of the certificate using the CA's public-key. Then, it hashes the contents of the certificate and compares it with the digest value in the same certificate. If the certificate is valid, *Base B* extracts *Mobile A*'s public key from the certificate, generates a random number R_B, encrypts it with mobile's public key. *Base B*, then, signs the values $K_A(R_B)$, the nonce, the list algorithms, and the chosen algorithms with its private key and sends the signature value to the mobile. *Mobile A* validates *Base B*'s certificate. If it is valid, *Mobile A* follows the same steps that the base performed and sends the signature value to the base. *Base B* validates mobile's signature and completes the authentication phase.

  The comparison values of this protocol to our protocols are given in the next chapter.

- **Certificate-Based Protocol by Park.**

  Chang-Seop Park [61] proposed an end-to-end authenticated session key exchange protocol based on the certificate over several distinct networks. Although Park does not build his protocol on a certain public key system, he assumes that the protocol can be built upon Modular Square Root (MSR) or RSA using small public exponent to reduce the computational load of the mobiles. The disadvantages of using this protocol for mobile systems are:

  1. Both the MSR and RSA system security are based on the difficulty of factoring large integers. These public key systems require large storage and bandwidth requirements during the execution of the protocol compared to those of ECC.
  2. According to the protocol, the user identifications are stored in certificates and these certificates are exchanged between the users and the network in plaintext form. Therefore, user identity confidentiality is not supported.

- **Beller-Chang-Yacobi Protocol.**

  This is also a public key protocol [15] based on MSR and it has the advantage of hiding the identity of the user. On the other hand, it uses the communication channel five times, and the total computational load will be six public key operations (PK Encryptions and Decryptions) [14]. The associated disadvantages are:

  1. The mobile has to do two PKE and one PKD in addition to many precomputations each time it sets up a call.
  2. It requires large storage and bandwidth requirements during the execution of the protocol compared to those of ECC.

- **The Protocol by Mu and Varadharajan.**

  In their paper [59] the authors address the design of security protocols for mobile environment. They highlight some of the weaknesses of Beller-Yacobi's and

Carlsen's [19] authentication protocols and then suggest improvements to these protocols to counteract interception attacks and to preserve the anonymity of users against outside (adversaries on the wireless communication channel) attackers. They propose both private and public key solutions.

1. Although the protocol claims it provides a certain degree of anonymity of the communicating users to other system users through the use of subliminal identities, the certificates in the protocol are exchanged in plaintext. This gives eavesdroppers a valuable information about the identities and the locations of the communicating parties.
2. The protocol proposed is based on MSR and MSR+DH based techniques that are storage and bandwidth inefficient.

3.2 Conclusion

In this chapter, we summarized some of the mutual authentication and key agreement protocols. The protocol by Molva, Samfat and Tsudik is based on symmetric key technique while others are based on asymmetric (public) key techniques. It is our conclusion that private key based protocols are faster and easier to implement. However, they do not provide some of the rich features of the public key based systems such as digital signatures. Although these proposed protocols seem promising, they are not efficient in terms of bandwidth and storage. In addition, each protocol has its own incapabilities and security flaws. It is clear that wireless communication devices and systems need very efficient, fast and secure protocols. In the next chapter, we propose a new protocol based on elliptic curve cryptographic techniques.

Chapter 4
A NEW AUTHENTICATION PROTOCOL BASED ON ECC

" The atom of modern physics can be symbolized only through a partial differential equation in an abstract space of many dimensions. All its properties are inferential; no material properties can be directly attributed to it. That is to say, any picture of the atom that our imagination is able to invent is for that very reason defective. An understanding of the atomic world in that primary sensuous fashion...is impossible"

Heisenberg

4.1 Introduction

The rapid progress in wireless mobile communication technology and personal communication systems has prompted new security questions. Since open air is used as the communication channel, the content of the communication may be exposed to an eavesdropper, or system services can be used fraudulently. In order to have reliable proper security over the wireless communication channel, certain security measures, e.g., confidentiality, authenticity, and untraceability need to be provided [30].

In a wireless mobile communication system, users and network servers need to authenticate one another and agree on a session key to be used for encryption purposes in their conversation [25, 46, 54]. Several authentication protocols have been proposed for wireless mobile communication systems. Among them, the protocol of Beller, Chang, and Yacobi [15] provides mutual authentication and key agreement between users and servers with low computational burden on the user side. This is important since the users usually communicate using a small, portable handset with limited power and processing capability. Furthermore, Aziz and Diffie [12] also proposed a similar authentication protocol, which does not require pre-computation.

These two protocols along with several others [16, 49, 57, 59, 51, 61] use public key cryptography techniques and off-line certification procedures.

Conventional (secret-key) cryptographic techniques require on-line certification of the users using secret key encryption. Both parties must share a common secret key in advance. On the other hand, public-key cryptosystems have separate keys for encryption and decryption. In addition, public-key cryptography provides digital signature methods for signature generation. Since the generation of signatures for long messages has high latency, long messages often go through a one-way hash function and the signature operation is performed on the hashed message.

A good example of security architecture on existing mobile digital cellular networks is the GSM (Global System for Mobile) which was the first digital cellular system to provide security services, e.g., user authentication, message confidentiality, and key distribution [63]. However, some of the security algorithms of the GSM architecture have recently been cryptanalyzed [67, 31].

To meet today's needs for wireless digital communication, the developed protocols need to be highly secure, requiring low computational overhead and thus low power. Here, we introduce a new authentication and key agreement protocol for wireless mobile communication systems. The protocol is based on elliptic curve cryptography. In the following, we first provide the necessary background for security requirements in a wireless communication system in section §4.2, and then give a brief introduction to elliptic curve cryptography in section §4.3. We give the details of the proposed protocol in section §4.4 and its comparison to the Beller-Chang-Yacobi and Aziz-Diffie protocols in section §4.5. The conclusions are found in section §4.6.

4.2 Security Requirements

We describe the security and efficiency features desirable in modern wireless communication systems and devices.

The security features are mutual authentication, non-repudiation of service, confidentiality and anonimity of users.

- **Mutual Authentication** One of the most important problems of the early cellular systems was the illegally made phone calls since a call request used to be usually granted by the service provider before the authentication has been done. By the time the fraud was perceived, several phone calls may have already been performed and therefore great losses have been incurred to the carriers [51]. This reason alone explains why authenticating the user is important. Authenticating the base is also necessary. In the case of more than one competitors located in the same market area, the base stations of one competitor should not be able to masquerade as those belonging to the other competitor [12]. In addition, impersonating servers helps adversaries obtain valuable confidential user information. With the advent of new digital technology and modern cryptographic techniques, such fraud can now be eliminated. In today's technology, it is very important for users and servers to authenticate one another before the call is being serviced.

- **Nonrepudiation of Service**

  For a perfectly designed mobile communication system, it should not be possible for users to deny the charges occurred by the use of service. This feature can easily be implemented in security protocols by the use of digital signatures.

- **Confidentiality**

  In today's technology, commercially available scanners can easily intercept radio signals traveling over the air in wireless mobile communication systems [65]. Although digital systems are advantageous over analog systems in terms of security, they are also prone to eavesdropping. To circumvent this problem, subscribers and servers must agree on a key that can be used to encrypt messages. Key agreement is usually the last part of the authentication process between subscribers and servers. Session keys must differ for each call to enhance security.

- **Anonymity of User**

  In a communication setting, some users may not want to expose their identities and/or their locations to third parties. In traditional phone systems, once the call is set up, the identities of the users are automatically exposed to the servers. In wireless systems, an authentication protocol can be designed in a way that every user is assigned a temporary identity when it gets its certificate from the Certificate Authority (CA). Then, the assigned identities can be used in that domain. The proposed protocol in this chapter hides the identity of the portable device from an eavesdropper on the communication channel. It is also possible to hide the identity of the portable device from foreign network server (the ones other than the device's home network). However, we do not address this issue in this chapter.

- **Physical Requirements**

  For any cryptographic protocol, security is the most important concern, however, when it comes to the practical implementation of the protocol, efficiency should also be considered. Modern wireless technology requires low power consumption and fast computational overhead, since light-weight, short-life batteries are used to operate handheld devices. This in turn requires that the complexity of the cryptographic operations should be minimal. Certain factors, e.g. hardware complexity, battery power, and computational delay, play important roles in the design of public key protocols for wireless communication [14]. Storage requirement is another important factor, since we often need to store the public keys, certificates, and temporary data used during the execution of the protocol. Although storage may not be a severe factor for the server side, it is highly important for the mobile side. The number of exchanged messages and the amount of memory required to store the necessary data should also be kept at a minimum in order to have an efficient protocol.

4.3 Elliptic Curve Cryptography

The protocol described in this paper depends on the security of the so-called *elliptic curve primitives*, e.g., key generation, signature generation, and signature verification. These operations utilize the arithmetic of *points* which are elements of the set of solutions of an elliptic curve equation defined over a finite field. The security of the protocol depends on the intractability of the elliptic curve analogue of the discrete logarithm problem which is a well known and extensively studied computationally hard problem.

We first define the nomenclature and then provide a general overview of these primitives. The mathematical background of the elliptic curve cryptography can be found in [55, 44].

4.3.1 Basic Definitions and Notation

- Scalar: An element belonging to either one of the fields $GF(p)$ or $GF(2^k)$ is called a scalar. Scalars are named with lowercase letters: r, s, t, etc.

- Scalar Addition: Two or more scalars can be added to obtain another scalar. In the $GF(p)$ case, this is the ordinary integer addition modulo p. When $GF(2^k)$ is used, this is equivalent to polynomial addition modulo an irreducible polynomial of degree k, generating the field $GF(2^k)$. We will denote the scalar addition of r and s giving the result e by $e = r + s$.

- Scalar Multiplication: Two or more scalars can be multiplied to obtain another scalar. In the $GF(p)$ case, this is the ordinary integer multiplication modulo p. When $GF(2^k)$ is used, this is equivalent to polynomial multiplication modulo an irreducible polynomial of degree k, generating the field $GF(2^k)$. We will denote the scalar multiplication of r and s giving the result e by $e = r \cdot s$.

- Scalar Inversion: The multiplicative inverse of an element a in $GF(p)$ or $GF(2^k)$ is denoted as a^{-1} which is the number with the property $a \cdot a^{-1} = 1$. It is often computed using the Fermat's method or the extended Euclidean algorithm.

- Point: An ordered pair of scalars satisfying the *elliptic curve equation* is called a point. Capital letters are used to denote these elements: P, Q, etc. We will also denote a point P using its coordinates $P = (x, y)$, where x and y belong to the field. Furthermore, the x and y coordinates of P will be denoted by $P.x$ or $P.y$, respectively.

- Point Addition: There is a method to obtain a third point R on the curve given two points P and Q, using a set of rules. This is called an elliptic curve point addition. We will use the symbol '+' to denote the elliptic curve addition $R = P + Q$. This should not be confused with scalar addition.

- Elliptic Curve Group: The set of the solutions of the elliptic curve equation together with a special point called *point-at-infinity* form an additive group if the point addition operation defined above is taken as the group operation.

- Point Multiplication: The multiplication of an elliptic curve point P by an integer e will be denoted by $e \times P$. It is equivalent to adding P to itself e times, which yields another point on the curve.

In addition to the above elliptic curve cryptographic primitives, we need a secret-key cryptographic algorithm and a one-way hash (message digest) function which are defined below:

- Secret Key Algorithm: A secret-key encryption algorithm is used to encrypt (hide) the data in the protocol. A conventional stream cipher (RC4 or SEAL) or a block cipher (DES, 3DES, IDEA) in the cipher-block-chaining mode can be used. The encryption and decryption operations using the key K acting on the plaintext M and the ciphertext C are denoted as $C := E(K, M)$ and $M := D(K, C)$, respectively.

- Message Digest Function: A message digest function compresses a long message into a short value which is usually 128 or 160 bits long. Two widely used and standardized hash functions are MD5 and SHA. We will denote the message

digest of a message M by $H(M)$. The signature functions take $H(M)$ as an input for efficiency reasons. The hash of the concatenation of two messages M_1 and M_2 is denoted as $H(M_1, M_2)$.

4.3.2 Elliptic Curve Digital Signature Algorithm

First, an elliptic curve E defined over $GF(p)$ or $GF(2^k)$ with large group of order n and a point P of large order is selected and made public to all users. Then, the following key generation primitive is used by each party to generate the individual public and private key pairs. Furthermore, for each transaction the signature and verification primitives are used. We briefly outline the Elliptic Curve Digital Signature Algorithm (ECDSA) below, details of which can be found in [37].

ECDSA Key Generation The user A follows these steps:

1. Select a random integer $d \in [2, n-2]$.
2. Compute $Q = d \times P$.
3. The public and private keys of the user A are (E, P, n, Q) and d, respectively.

ECDSA Signature Generation The user A signs the message m using the following steps.

1. Select a random integer $k \in [2, n-2]$.
2. Compute $k \times P = (x_1, y_1)$ and $r = x_1 \bmod n$.

   If $x_1 \in GF(2^k)$, then it is assumed that x_1 is represented as a binary number.

   If $r = 0$ then go to Step 1.
3. Compute $k^{-1} \bmod n$.
4. Compute $s = k^{-1}(H(m) + d \cdot r) \bmod n$.

   Here H is the secure hash algorithm SHA.

   If $s = 0$ go to Step 1.

5. The signature for the message m is the pair of integers (r, s).

ECDSA Signature Verification The user B verifies A's signature (r, s) on the message m by applying the following steps:

1. Compute $c = s^{-1} \bmod n$ and $H(m)$.
2. Compute $u_1 = H(m) \cdot c \bmod n$ and $u_2 = r \cdot c \bmod n$.
3. Compute $u_1 \times P + u_2 \times Q = (x_0, y_0)$ and $v = x_0 \bmod n$.
4. Accept the signature if $v = r$.

4.4 Proposed Protocol

As is customary in most security protocols, we assume that there is a certificate authority (CA) which creates and distributes certificates to the users and servers on their request. These certificates contain a temporary identity assigned by the CA for the requesting party, the public key of the requesting party, and the expiration date of the certificate. The concatenated binary string is then signed by the CA's private key to obtain the certificate for the requesting party. By using a certificate the identity of a particular party is bound to its public key. The acquisition of the certificate is performed when the users and servers first subscribe to the service. The certificates are updated at regular intervals.

In a wireless environment, it is often necessary to request service outside of users home networks. In this case, the visited network checks the certificate's expiration date with the users home network in order to decide whether it needs to provide service to the requesting party. Thus, the authentication and communication protocols should be designed in such a way that the users can easily be authenticated on-line via their home networks.

Figure 4.1. Network Server Initialization.

SERVER		CERTIFICATION AUTHORITY
• Choose $d_s \in [2, n-2]$		• Choose $k_s \in [2, n-2]$
• $Q_s = d_s \times P$		• $R_s = k_s \times P$
• Send	$\xrightarrow{Q_s}$	• Receive
		• Choose unique I_s
		• $r_s = R_s.x$
		• $M = H(Q_s.x, I_s, t_s)$
		• $s_s = k_s^{-1}(M + d_{ca} \cdot r_s)$
• Receive	$\xleftarrow{Q_{ca}, I_s, (r_s, s_s), t_s}$	• Send
• $e_s = H(Q_s.x, I_s, t_s)$		
• Store $Q_s, Q_{ca}, I_s, (r_s, s_s)$ e_s, t_s		

4.4.1 Protocol Goals

The designed protocol should achieve the following goals by the end of a successful run:

1. Mutual authentication of server and user;
2. Agreement between user and server on a secret authentication key to protect the data used in mutual authentication process;
3. Non-repudiation of origin by user for relevant data sent from user to server;
4. Agreement between user and server on a secret session key which will be used to encrypt data sent by each party;
5. User confidentiality.

Figure 4.2. User Terminal Initialization.

USER		CERTIFICATION AUTHORITY
• Choose $d_u \in [2, n-2]$		• Choose $k_u \in [2, n-2]$
• $Q_u = d_u \times P$		• $R_u = k_u \times P$
• Send	$\xrightarrow{Q_u}$	• Receive
		• Choose unique I_u
		• $r_u = R_u.x$
		• $M = H(Q_s.x, I_s, t_s)$
		• $s_u = k_u^{-1}(M + d_{ca} \cdot r_u)$
• Receive	$\xleftarrow{Q_{ca}, I_u, (r_u, s_u), t_u}$	• Send
• $e_u = H(Q_u.x, I_u, t_u)$		
• Store $Q_u, Q_{ca}, I_u, (r_u, s_u)$		
e_u, t_u		

Non-repudiation of origin by user for relevant data is usually provided by performing on-line signature operations by users. The idea, here, is that these signatures contain the private keys of the users, which are only known by them. In our protocol, for efficiency reasons we do not apply on-line signature generation operations, instead we use the private keys of the users to obtain the mutual authentication keys. Therefore the protocol achieves the desired goal in a different way than almost all the other authentication protocols.

4.4.2 Terminal and Server Initializations

In order to receive a certificate, the terminal sends its public key Q_s together with its user identity through a secure and authenticated channel to the CA. The CA uses its private key to sign the hashed value of the concatenation of the public key, the temporary identity I_s, and the certification expiration date t_s. The CA then sends

the signed message through the secure and authenticated channel to the terminal as shown in Figure 4.1.

Establishing a secure channel from the certification authority to the terminal is a common and accepted assumption in almost all authentication protocols. In practice the CA may use the postage system as the secure channel to distribute the signed messages and temporary identities stored within a smartcard. The signed message is the certificate of the terminal which is used in future authentication and key generation processes. By repeating the very same process the user acquires its certificate as shown in Figure 4.2.

The certificate consists of a pair of integers which is denoted as (r_s, s_s) for the server and (r_u, s_u) for the user. Here r_u and r_s are the x coordinates of the (distinct) elliptic curve points R_u and R_s, respectively. As mentioned earlier, the proposed protocol is based on the ECDSA.

4.4.3 Mutual Authentication Between Terminal and Server

The protocols shown in Figures 4.1 and 4.2 are executed off-line. The mutual authentication and key agreement protocols between the terminal (user) and the server need to be executed in real-time. We give the combined protocol in Figure 4.3.

According to the protocol, whenever there is a service request either by the user or by the server, there is an immediate key exchange. The initiated party will also send a random challenge to the initiating party. Sending the public keys unprotected over the air does not introduce any threat to the security of the system. Once both sides have the other party's public key, they simultaneously generate a secret key to encrypt the data required to have a mutual authentication. This task is accomplished by multiplying the other party's public key Q_2 with this party's private key d_1 as shown in Figure 4.3.

To protect the certificates from an eavesdropper, it is necessary to send the certificates in encrypted form. For this reason, the protocol uses a secret key cipher to encrypt the certificates using the mutually agreed secret key $Q_k.x$. Here $Q_k.x$

represents the x-coordinate of the point Q_k on the curve. The server encrypts the concatenation of its certificate $e_s, (r_s, s_s)$, the certificate expiration date t_s, and a random number g_s which will be used to obtain the final mutual key of the communication. The final content should also include the challenge if the server is the initiating party. The certificates are usually sent unencrypted in almost all the other authentication protocols. In our protocol, we do not reveal the content of the certificate which may be useful for spoofing attacks. This increases the encryption time only slightly since the certificate is not very long (on the order kilobits) and the encryption algorithms are very fast (on the order of megabits per second).

The encrypted message C_0 is then sent to the user. The user then decrypts C_0 and obtains the certificate of the server, the random number g_s and the challenge g_u which in this case sent by itself. Obtaining the original challenge value back from the server confirms the freshness of the message and prevents the reply attacks. The user immediately encrypts the concatenation of its certificate $e_u, (r_u, s_u)$, the certificate expiration date t_u, and the random number g_s. This encrypted data which is denoted as C_1 is sent to the server.

Next, the user checks the validity of the certificate, and if it is invalid, the user aborts the communication. On the other side, the server decrypts C_1 and checks whether g_s and the time certificate are valid. If not, it aborts. This mechanism, specifically the use of g_s, defeats *spoofing* attacks by the user side and also prevents unnecessary computation. Next, the server checks the validity of the certificate and accordingly grants or aborts the service. Note that it may be a good idea to generate multiple g values in advance so that the protocol could save some time. However, storing these multiple random numbers will increase the storage requirement of the protocol, which is undesirable.

4.4.4 Key Agreement

After the verification procedure has been completed by both sides, the user and the server are now ready to use the channel that has been reserved for their communi-

Figure 4.3. Mutual Authentication and Key Agreement.

USER		SERVER
• Receive	$\xleftarrow{Q_s}$	• Send
• Generate a random number g_u		
• Send	$\xrightarrow{Q_u, g_u}$	• Receive
• $Q_k = d_u \times Q_s = (d_u \cdot d_s) \times P$		• $Q_k = d_s \times Q_u = (d_s \cdot d_u) \times P$
• $Q_k.x$: The mutually agreed key		• $Q_k.x$: The mutually agreed key
		• Generate a random number g_s
		• $C_0 = E(Q_k.x, (e_s, (r_s, s_s), t_s$ $g_u, g_s))$
• Receive	$\xleftarrow{C_0}$	• Send
• $D(Q_k.x, C_0)$: Is g_u present?		
• $C_1 = E(Q_k.x, (e_u, (r_u, s_u)$ $t_u, g_s))$		
• Send	$\xrightarrow{C_1}$	• Receive
		• $D(Q_k.x, C_1)$
		• If g_s and t_u are valid, then
• $c = {s_s}^{-1}$		• $c = {s_u}^{-1}$
• $u_1 = c \cdot e_s$		• $u_1 = c \cdot e_u$
• $u_2 = c \cdot r_s$		• $u_2 = c \cdot r_u$
• $R = u_1 \times P + u_2 \times Q_{ca}$		• $R = u_1 \times P + u_2 \times Q_{ca}$
• $v = R.x$		• $v = R.x$
• If $v \neq r_s$, then abort		• If $v \neq r_u$, then abort
• $k_m = h(Q_k.x, g_s, g_u)_{msb-64}$		• $k_m = h(Q_k.x, g_s, g_u)_{msb-64}$
• k_m: The unique secret key		• k_m: The unique secret key

cation. However, there is one more step to complete our goal to have a full secure protocol: To generate a secret key known by each side to encrypt the conversation.

They do already have a secret key $Q_k.x$; however, this key cannot be used since it will be the same during the valid time limits of their certificates. Therefore, we need to add a new key exchange step to agree on a unique key to be used for communication during each session. However, we do not prefer to execute another key agreement process due to previously stated power and memory limitations. Instead, we will use the previously generated random numbers g_u and g_s which are known by both sides to generate a new secret key without using the channel again. Both the server and the user perform a hash operation to obtain the new secret key, which we call k_m. This key now can be used for encrypting the data sent through the channel.

4.5 Comparisons

Below we compare certain properties of the proposed protocol to those of Aziz-Diffie [12] and Beller-Chang-Yacobi [16] protocols. The performance values for the Aziz-Diffie and Beller-Chang-Yacobi protocols are found in [13], summarized in Table 4.1.

- The proposed protocol requires less bandwidth compared to other public-key based protocols, e.g., Aziz-Diffie and Beller-Chang-Yacobi protocols. The total number of bits exchanged in the real-time portion of the protocols is given as follows:

Beller-Chang-Yacobi:	8320 bits (1024-bit key)
Aziz-Diffie:	8680 bits (1024-bit key)
Proposed:	1666 bits (160-bit key)

In our protocol, the parameter lengths are as follows:

Q_u, Q_s:	$161 - bit$
e_u, e_s:	$160 - bit$
$(r_u, s_u), (r_s, s_s)$:	$320 - bit$
t_u, t_s, g_u, g_s:	$64 - bit$

- The proposed protocol has low storage requirements for the user side, which makes it suitable for smartcards and other handheld computing devices:

Beller-Chang-Yacobi:	5120 bits (1024-bit key)
Aziz-Diffie:	2176 bits (1024-bit key)
Proposed:	1440 bits (160-bit key)

- The proposed protocol has modest computational load on the user side for real-time execution:

Beller-Chang-Yacobi:	2 PKE (1024-bit) + 1 PKD (1024-bit) + Precomputation
Aziz-Diffie:	3 PKE (1024-bit) + 2 PKD (1024-bit)
Proposed:	1 eP (160-bit) + 1 ECDSAV (160-bit) +
	2 SKE (800-bit data)

The meaning of the above symbols are as follows:

PKE:	Public Key Encryption
PKD:	Public Key Decryption
eP :	Point Multiplication
ECDSAV:	Elliptic Curve Digital Signature Algorithm Verification
SKE:	Secret Key Encryption or Decryption

4.6 Conclusions

We have described an authentication and key agreement protocol for wireless communication based on elliptic curve cryptographic techniques. The proposed protocol is a public-key type with the feature of off-line certification procedure. It is a well-known fact that the public-key cryptography concept solves the key distribution and storage problems, which are typical in secret-key settings. The protocol provides certain security services, e.g., nonrepudiation, anonymity of user, service expiration mechanism using time certificates, as most recent secret and public-key based protocols also provide. In the proposed protocol, we protect the message content and

Table 4.1. Parameter Lengths (in bits) in Beller-Yacobi and Aziz-Diffie.

Beller-Chang-Yacobi		Aziz-Diffie	
Hashed certificate	384	Exchanged messages	8680
Public key	1024	Public key	1024
Certificate content	1536	Hashed messages	1030
Random number	1024	Random number	1024
Random challenge	128	Identity	512

certain system parameters using a secret-key cryptographic algorithm, e.g., RC4, IDEA, DES, or 3DES.

The protocol is based on the elliptic curve digital signature algorithm (ECDSA), and inherits the security and implementation properties of the elliptic curve cryptosystems which seem to offer the highest cryptographic strength per bit among all existing public-key cryptosystems. With a 160-bit modulus, an elliptic curve system seems to offer the same level of cryptographic security as DSA or RSA with 1024-bit moduli. The smaller key sizes result in smaller system parameters, smaller public-key certificates, bandwidth savings, faster implementations, lower power requirements, and smaller hardware processors [55].

The RSA-based protocols have significant problems in terms of the bandwidth and storage requirements. Currently, the RSA algorithm requires that the key length be at least 1024 bits for long term security, however, it seems that 160 bits are sufficient for elliptic curve cryptographic functions. Thus, the use of the ECC in wireless communication system is highly recommended. The proposed protocol is a step in this direction.

Chapter 5
AN END-TO-END WIRELESS SECURITY PROTOCOL

"You do not really understand something unless you can explain it to your grandmother."

A. Einstein

5.1 Introduction

One of the reasons cryptography is so interesting and challenging is that 99 cryptographers can look at an algorithm, protocol, or system and see nothing wrong, but a single cryptographer with a new insight can sometimes break it. This is why cryptanalysis of security protocols is necessary operations and it is mostly used to find out the flaws in security protocols. In many cases it is the wireless security protocols that benefit and/or suffer the most from these analyses.

Mobile communications is more vulnerable to security attacks such as interception and unauthorized access than fixed network communications. Therefore services for securing mobile communications are vital in guaranteeing authentication and privacy of legitimate users [59]. *Authentication service* is required to counteract masquerading, *confidentiality and integrity services* are required to protect information against unauthorized eavesdropping and modification, *authorization* is required to control resource access and usage, and finally *nonrepudiation and auditing services* are also required to complete the big security picture. The security and privacy methods [42] in existing wireless systems are given in Table 5.1.

There are many networks with different level of protection and some of them are more vulnerable than others to attacks from intruders or insiders. Fraud will happen once the security parameters are compromised, either by an intruder or insider. The security parameters sent from a user's home domain to a visited domain may be

intercepted by an insider. In some cases the damage is severe. For example, if the (K_c, R, S) tuple in GSM [1], KS in DECT [2], or the SSD in USDC [26] are known by the attacker, he can use it to impersonate a legal network (base station) to establish a connection with the corresponding subscriber, and hence draw some private information of the conversation, e.g., the identities of parties involved in the communication [51]. He can also pretend to be the other party of the call to gather further information until such impersonation is detected. This attack can be repeatedly launched in GSM because the mobile unit is not designed to detect used security parameters. Even with public-key approach, some subscriber-specific sensitive information could also be found in a visited network, e.g, the common key k in MSR+DH [15]. With knowledge of k, the attacker can always impersonate the legal subscriber within this specific network.

The interception mentioned above can occur both legal and illegal purposes. So, it is proper at this point to define and differentiate legal and illegal interception categories [32].

- *Illegal interceptions*: Illegal interceptions are those at the initiative of a person, group, organizations, or foreign power. These are practiced by individuals and agencies, private or not, to illicitly obtain information: Personal, commercial, industrial, economic, political, etc., for various purposes: *disloyal competition, blackmail, espionage, etc.*

- *Legal interceptions*: Judicial authority can order interceptions **in criminal matter** under certain conditions prescribed by the law. It can be also authorized exceptional interceptions for **security reasons** upon decision and under the responsibility of the authority concerned.

Under some situations, an old session key can also be derived without having to break a mobile unit or a service provider. With this compromised session key with other recorded information, an adversary can make fraudulent calls or masquerade as a legitimate-serving network to establish a false connection with the subscriber.

Security related concerns for current GSM security algorithms are also growing [41]. The approximate design of A5/1 was leaked in 1994. In 1999 the exact design of both A5/1 and A5/2 was reverse-engineered by Briceno. The results were confirmed against official test vectors [6]. The worst news came from Israel in December 1999. Alex Biryukov and Adi Shamir have claimed that they developed an efficient way to attack A5/1 algorithm [17]. By using a single PC with 128 MB RAM and two 73 GB hard disks, they have been able to find the key, used in the algorithm, in less than a second by analyzing the output of the A5/1 algorithm in the first two minutes of the conversation. Compared the previous attacks on A5/1 algorithm this is a significant improvement in terms of necessary power and equipment size needed to break the algorithm.

One of the main concerns of the GSM and CDPD authentication approaches is that these methods rely on the assumption that the "fixed network" is secured [62]. Therefore, messages are transmitted in a clear text form between MSCs which trust each other. However, this trust policy can not be assumed for large heterogeneous network environment managed by different administrative authorities. Thus, a security architecture with minimal assumptions about the security of intermediate transport networks is needed.

Another point of contention with GSM is the manner of distributing user authentication information. The home domain is expected to generate a set of challenge/response pairs on the fly [62]. The set is then used by the foreign domains in successive authentication flows with the end-user. The solution is inefficient in terms of both bandwidth consumption and the overhead incurred in the home domain.

Mobility brings new possibilities to the services, but places at the same time additional requirements on the service deployment. Because of the fundamental limitations on power and available spectrum, wireless networks have less bandwidth, more latency, less stability, and less predictability availability [28]. The wireless devices have restricted power consumption, less memory, smaller and lower quality displays, and different input devices [53]. Therefore, the protocols proposed should bring the minimum computational and memory requirements on mobile devices.

Table 5.1. Security and Privacy in Existing Wireless Systems.

MIN/ESN	AMPS: 10 digit mobile ID, 32 bit equipment serial number. All data sent in clear, systems share info on bad MIN/ESN.
Shared Secret Data	TDMA/CDMA cellular: Secret shared between mobile station and system
Security Triplets	GSM: challenge/response pairs plus privacy key. Home system generates 3-5 for visited system; one used per connection.
Public Key	PACS proposed as an option. Avoids need for communication with home system

The intention of this chapter is to explore better-suited security solutions for rapidly growing mobile environment. We mainly focus on the following goals:

- Overcome the known flaws of the existing private and public key security protocols.
- Improve the resiliency and interoperability of the security protocols by carefully selected design approaches.
- Make it possible for wireless handheld devices to use the up-to-date new technology applications with an acceptable level of security without worrying too much on processing and memory limitations.

In this chapter we fully define and analyze end-to-end security protocols for mobile users. The protocol given in chapter 4 provides an efficient method of authenticating users and servers, however it does not address end-to-end user authentication issues. The protocol given in this chapter improves the previous one in terms of security and interoperability.

Table 5.2. Abbreviations and Notation.

MS, BS	Mobile and base station
SIM	Subscriber identity module
IMEI	International mobile equipment id.
MSC	Message switching center
BSC	Base station controller
HN, VN	Home and visiting network
HLR, VLR	Home and Visiting location register
CA	Certification authority
AC	Authentication center
MSR	Modular square root
ECC	Elliptic curve cryptosystems
SKE, PKE	Secret and public key encryption
SKD, PKD	Secret and public key decryption
DES	Data Encryption Standard
SHA	Secure hash algorithm
$\mathbf{Q_A, d_A}$	Public and private key of A
$\mathbf{ID_{Atemp}}$	Temporary identity of A
$\mathbf{Cert_A}$	Certificate of A
E(K, M)	Encryption on M using key K
D(K, M)	Decryption on M using key K
Sign(M)	Signature operation on M
Ver(M)	Signature verification on M
EIR	Equipment identity register
ISDN	Integrated Services Digital Network
PSTN	Public Switched Phone Network
PSDN	Public Switched Data Network

The proposed protocol has significantly lower computational overhead and memory requirements due to the use of elliptic curve cryptography and the selected design method. The chapter is organized as follows: In section 5.2, we define the abbreviations and notations that are used throughout the chapter. The general desired security features and implementation requirements in mobile environments are given in section 5.3. Section 5.4 defines the mobile environment, protocol goals, design criteria and the end-to-end user mutual authentication and key agreement protocols. The protocol's achievement on the proposed goals and the security analysis of it are given in section 5.5. In section 5.6 we compare the bandwidth and memory requirements of this improved protocol once again to the well-known Aziz-Diffie and Beller-Chang-Yacobi protocols. Finally the conclusion is given in section 5.7.

5.2 Abbreviations and Notation

The short definitions used throughout the chapter are given in Table 2. For the remaining of this chapter, these abbreviations and notation will be used.

5.3 Desired Security and Implementation Requirements

The definitions and discussion of the desired security features and implementation requirements are given in 4.2. Here we just list the features and discuss the mobile device security.

- Mutual Authentication
- Nonrepudiation of Service
- Confidentiality
- Anonymity of User
- Physical Requirements

- Mobile Terminal Security

  Cloning refers to the unauthorized modification of the terminal identity, or the allocation of an existing identity to a fake terminal that does not conform to the standards and directives for type approved terminals [3]. Terminals that have been stolen are given new identities and often are exported to other countries for legitimate use. This is a major problem. The network can detect the existence of cloned terminals but there is no way of distinguishing between the legitimate and cloned terminal, or the process of achieving that is inefficient and therefore avoided. The terminal identity should therefore be stored in such a way that it is easy to read but very difficult or meaningless to change. The terminal identity should securely be stored in terminals during the manufacturing process. For example IMEI's of GSM terminals are stored by distributing the identity among random data. However, a link between a valid SIM card and the terminal in use should be established in order to be able to detect, prevent and/or make useless the cloning of terminals. Furthermore user defined access code and biometrics solution could associate a terminal to its legal owner.

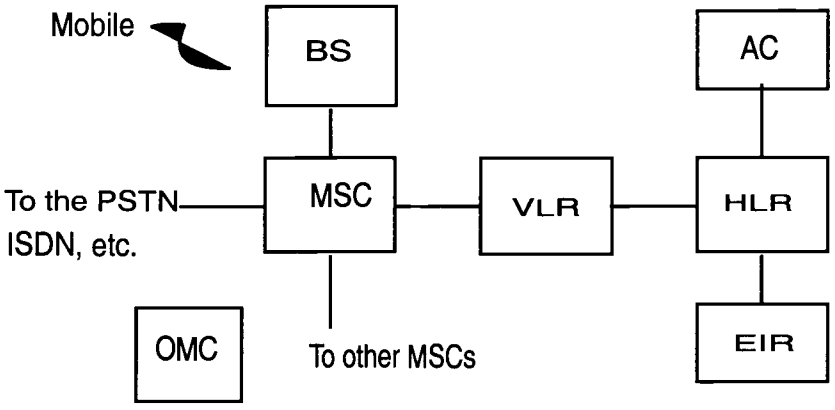
  From the network side, these numbers are stored in a central database EIR, which contains three lists: *White*, *grey*, and a *black* list. The *white list* contains all the terminals which are to be admitted for use on the network. The *black list* contains barred terminals that have no rights to access the network. Stolen terminals are examples of items in such a list. The terminals in the *grey list* are the ones that have to be monitored as they are used to access the network.

5.4 Proposed Protocols

5.4.1 Mobile Environment

A simple mobile computing environment is shown in Figure 5.1. Terminals access the network via a mobile network system. The network system consists of BS, LR(VLR, HLR), MSC, AC, etc. The home location register contains information related

Figure 5.1. Mobile Network Environment.



to the location and subscription of the users in its domain. We assume that the authentication server is present in every domain. The authentication servers store confidential information and are assumed to be physically protected. The mobile stations can move from one domain to another or within the same domain.

The Home Location Register and the Visitor Location Register are electronic directories which work together. They permit both local operation and roaming outside the local service area. Most often these two directories are located in the same place.

The VLR or visitor location registry contains roamer information. Once the visited system detects the mobile passing through another carrier's system, its VLR queries the terminal's assigned home location register. The VLR makes sure the mobile is a valid subscriber, then retrieves just enough information from the now distant HLR to manage the mobile's call. It temporarily stores the last known location area, the power the mobile uses, special services it subscribes to and so on. The cellular network now knows where the terminal is and can direct calls to it.

The AC is the Authentication Center, a secured database handling authentication and encryption keys. (GSM, PCS 1900, and certain cellular systems support these features.) As we'll see later authentication verifies a mobile customer with a complex challenge and reply routine. The network sends a randomly generated num-

ber to the mobile. The mobile then performs a calculation against it with a number it has stored and sends the result back. Only if the switch gets the number it expects does the call proceed. The AC stores all data needed to authenticate a call and to then encrypt both voice traffic and signaling messages.

The EIR or Equipment Identity Register is another database. The EIR lists stolen phones, fraudulent telephone identity numbers, and faulty equipment. It is one tool to deny service or track problem equipment.

The OMC or operation maintenance center is network control. It monitors every aspect of a cellular system. A maintenance center may monitor several carrier's systems. Every OMC is staffed twenty four hours a day.

5.4.2 Motivation

Although a subscriber does not need to share a secret authentication key with the serving network as demonstrated in protocols [12, 15, 61] using public key techniques, these protocols inevitably involve a great deal of computation and the key certification/revocation problem. This is why the standard bodies have chosen to reject the public key based solutions and decided to implement private key based approaches. However, symmetric key based authentication and key agreement protocols have their own problems. Symmetric key methods rely on visited networks for protecting secret data. To reduce a roamer's trust on the visited network's capability of protecting sensitive data, public-key techniques should be incorporated [51]. Key management issue is another drawback for the private key only protocols. Furthermore, non-repudiation is an important security feature that can only be provided with digital signatures. Public key techniques provide such signatures.

In summary, today's wireless systems need security protocols involving public key based approaches to generate an authenticated key, and private key techniques to use this key to encrypt the communication channel. Considering the limited capabilities of mobile devices, the design should involve computationally efficient public key protocols and design methods.

It is our belief that by using the ECC based techniques, the desired security level can be provided with low computational load and memory/bandwidth requirements on the constrained mobile environment.

5.4.3 Design Criteria and Protocol Goals

In order to avoid the mentioned drawbacks of existing systems like GSM and the other proposed public-key based protocols, the solution must take into account the following criteria along with the general security features such as mutual authentication between the mobile and the server and unique encryption key for each session:

- *Domain Separation*: Domain specific secret or sensitive information should not be propagated between the home network and the visited network [62].
- *Transparency to Users*: Foreign domain authentication should have minimal impact on the user interface with respect to authentication in the home domain [62].
- *User Identity Confidentiality*: All the user identification information must be kept secret. Providing privacy of location using private-key techniques, such as in GSM, has a major drawback [15]. This type of protocols require synchronization, and has a protocol to reestablish identification when synchronization is lost. This synchronization is insecure. By contrast, this property can be provided with the public-key approach with no such synchronization requirement.
- *Minimal Overhead*: The number of messages exchanged between home domain and foreign domain should be kept minimal.
- *Interoperability*: This refers to the capability of using different shared key algorithms for domestic or foreign use of mobile devices.
- *Forward Secrecy*: Breaking the key of either the mobile or the server sometime in the future should not compromise the data that has been exchanged between the parties.

5.4.4 Security Levels

It should be possible for users and service providers to choose the security level for their needs. A regular person may not need any extra security while a company CEO or a government employee might feel the need for more security than they get from the services in use. Although many people feel confident and private in their mobile conversations and data exchanges, they start getting worried when they hear very simple possible threat scenarios to their privacy. In any case everybody needs some sort of security. However, the level of security should be chosen wisely for different types of users and applications, and for different geographical locations. It is a common belief that the security level should be chosen according to the value of the exchanged data. In some cases, the cost of an attack scenario on a target (a conversation, data exchange) might be greater then the target's value. There is no need to support such applications or instant communications with high level of security. A list of possible security levels are given as follows [42]. Our interest in this chapter is the first three levels. It is always possible to increase the key size of the public key cryptosystem (in our case ECC) significantly and claim that the application's security level is 3. However, there is always suspicion in crypto world that even the best known public key based algorithms such as RSA, DSA or ECC, and private key based algorithms such as DES, 3DES, IDEA, RC5, etc., might very well be broken very easily with the special information that a high level office such as NSA might have.

- *Level 0: No Privacy*

  No encryption is provided.

- *Level 1: Equivalent to Wireline*

  For routine conversations a decent level authentication and privacy can be provided with the assumption that the level of effort to break the protocol should take around 1 year.

- *Level 2: Commercially Secure*

  1024-bit RSA and 160-bit ECC should provide this level of security. In fact, in recently written paper [47], Lenstra gives the lower bounds for computationally equivalent key sizes and corresponding hardware cost and number of years to break a key. According to his table for the year 2000; the key sizes should be around 70 for secret key algorithms, 952 for classical public key algorithms such as RSA, and 132 for ECC. For the year 2020, the numbers are given as 86, 1881 and 151 respectively. The corresponding lower bound hardware cost and number of years for the year 2000 are $1.39 * 10^8$ US Dollars and $1.58 * 10^7$ years, and for the year 2020 are $5.55 * 10^8$ US Dollars and $6.54 * 10^{11}$ years, respectively. Note that the corresponding number of years were calculated on 450MHz PentiumII PC.

- *Level 3: Government/Military Secure*

  We do not make any assumption on this level of security. Security Level 2 may very well fall in this category.

  The protocols defined in this chapter target the Security Level 2 category. However, realizing that some environments may need low level of security, a simpler version of the protocol with less security is also shown.

5.4.5 Terminal and Server Initializations

We now explain our protocol step by step starting with the initialization of users and servers. This section shows the creation of public and private keys and the acquisition of certificates.

In order to receive a certificate, the terminal sends its public key Q_s together with other relevant or necessary data through a secure and authenticated channel to the CA. The CA uses its private key to sign the hashed value of the concatenation of the public key, the user identity I_s, and the certification expiration date t_s. The CA then sends the signed message through the secure and authenticated channel to the terminal as shown in Figure 5.2.

Figure 5.2. Base/Terminal Initialization.

SERVER		CERTIFICATION AUTHORITY
• Choose $d_s \in [2, n-2]$		• Choose $k_s \in [2, n-2]$
• $Q_s = d_s \times P$		• $R_s = k_s \times P$
• Send	$Q_s, server_{info} \longrightarrow$	• Receive
		• Choose unique I_s
		• $r_s = R_s.x$
		• $M = H(Q_s.x, I_s, t_s)$
		• $s_s = k_s^{-1}(M + d_{ca} \cdot r_s)$
		• $cert_s = (r_s, s_s)$
• Receive	$Q_{ca}, I_s, cert_s, t_s \longleftarrow$	• Send
• $e_s = H(Q_s.x, I_s, t_s)$		
• Store $Q_s, Q_{ca}, I_s, cert_s, e_s, t_s$		

Establishing a secure channel from the certification authority to the terminal is a common and accepted assumption in almost all authentication protocols. In practice the CA may use the postage system as the secure channel to distribute the signed messages and temporary identities stored within a smart card. The signed message is the certificate of the terminal which is used in future authentication and key generation processes. By repeating the very same process the user acquires its certificate just as given in Figure 5.2.

The certificates, $cert_s$ and $cert_u$, consists of a pair of integers which is denoted as (r_s, s_s) for the server and (r_u, s_u) for the user. Here r_u and r_s are the x coordinates of the (distinct) elliptic curve points R_u and R_s, respectively. As mentioned earlier, the proposed protocol is based on the ECDSA.

Figure 5.3. User Registration to a New Domain.

USER		SERVER
• Send	$I_{u.temp} \longrightarrow$	• Receive
• Receive	$\longleftarrow Q_s, I_s$	• Send
		• Send $I_{u.temp}$ to HLR for user location update
		• Retrieve Q_u, I_u from HLR
• $Q_k = d_u \times Q_s = (d_u \cdot d_s) \times P$		• $Q_k = d_s \times Q_u = (d_s \cdot d_u) \times P$
• $Q_k.x$: The mutually agreed key		• $Q_k.x$: The mutuallly agreed key

5.4.6 Mutual Authentication Between Terminals and Servers

The protocol shown in Figure 5.2 is executed off-line. It is or assumption that users and servers obtain their certificates and other related data from a trusted CA.

The mutual authentication and key agreement protocols between the terminal (user) and the server need to be executed in real-time when the user or the server initiates a communication session. However, before requesting a service from the server, the user needs to let the server know about its presence and its temporary identity if it is in a domain other then its home domain to which it is registered. According to the protocol there is an initial registration process between portable device (cellular phone, wireless workstation, etc.) and the access point (base station in a mobile phone system, a network server, etc.). We assume that the active user device informs the server about its presence by sending its temporary identity $I_{u.temp}$. In the case of the user being in a visited network, the visited network communicates to the user's home network. This enables the visited network to inform the home network register (HLR) about the latest location of the user, and also it enables the visited network to obtain the user-sensitive data such as the user's public key,

identity, etc. It is up to the application and to the home and visited networks to perform a check on the revocation list at this point to see if this specific user is legal to receive service. As it will be shown later, this check can also be performed when the user requests a service.

To ease the on-line authentication and key agreement process, the user and the server may perform an off-line initial key agreement. Once both sides have the other party's public key, they simultaneously generate a secret key to encrypt the data required to have a mutual authentication. This task is accomplished by multiplying the other party's public key Q_2 with this party's private key d_1 as shown in Figure 5.3. This should happen only once when the user visits another network the first time. After the first successful authenticated and encrypted service, the user and the server can now be sure about authenticity of the key they created. It is up to the application, to the service providers and to the users (in this key-sensitive case) to use this initial key to start future authentication sessions while the user's stay in the visited network. This process, if necessary, may become a part of the on-line authentication process at the cost of an elliptic curve point multiplication at both sides.

We assume that the users already know the associated public key of the server. Otherwise the server can broadcast its public key in a certain signaling format so that the user can tune in and obtain this key. This process can be monitored by legal devices so that the illegal and fake keys being broadcast can be detected. Note that this can be a standard procedure for a wireless network. This initial key agreement can also be performed on-line as mentioned before. The user can acquire the server's public key when it requests a service. We will discuss security concerns of both approaches in section 5.5.2.

The initial registration process is shown in Figure 5.3. Note that it is only the legal server that can associate the $I_{u.temp}$ to Q_u through a secured wired channel with the home network.

The on-line mutual authentication and key agreement protocol between user and the server is shown in Figure 5.4. According to the protocol, whenever there is a

service request either by the user or by the server, there is an immediate challenge by the initiated party. The initiated party will send a random challenge to the initiating party. The user may be either an initiating or an initiated party. The protocol is designed in a way that the user has to perform the same amount of operation in each case. In Figure 5.4, we consider the case that it is the user who initiates the server.

Upon receiving a challenge from the server, the user generates a random number that is its own challenge to the server. Then the user performs a signature operation (ECDSA) on the hash of both parties' public keys and challenge numbers. Including all these numbers are security related and will be discussed in section 5.5.2. The next step is to encrypt the newly generated signature, the user's certificate, the certification expiration date, and finally the user's challenge to the server, g_u. In addition to this data, a 64-bit parameter S is also encrypted. S includes the information of possible shared key encryption algorithms that the mobile terminal is capable of performing. The initiated party will have the priority to select one of the algorithms to be an encryption key for the upcoming voice or data communication. Note that there is a default encryption algorithm in use for the purpose of completing the authentication process. This ciphertext, called C_0, is then sent to the server. To this point, it was the user who initiated the server, and therefore it was trying to authenticate itself to the server.

Upon receiving the challenge response from the user, the server first decrypt C_0 to obtain the data hidden inside of it. The server checks the certification expiration date, t_u, first. If the date is valid, the server verifies the user's signature Sig_u by first computing the hash value K. Depending on the verification result, the server may continue or abort the authentication process. Then, it can either first verify the user's certificate and then generate the response Sig_s to the user's challenge, or vice versa. Being the initiated party, the server decides on the encryption algorithm and encryption mode which will be used in the communication after the authentication and key agreement process is completed. The decision is called S', and it is included in C_1. Next, C_1 is sent to the user as the challenge response by the server if the user's certificate is valid. As an example, the certificate verification steps using EC

Figure 5.4. Real-Time Execution of the Protocol.

USER		SERVER
• Send call initiation	$\xrightarrow{I_{u.temp}}$	• Receive
• Receive	$\xleftarrow{g_s}$	• Send
• Generate a random number g_u		
• $Sig_u = Sign(H(Q_s, Q_u, g_s, g_u))$		
• $C_0 = E(Q_k.x, (Sig_u, cert_u, t_u$		
$g_u, S))$		
• Send	$\xrightarrow{C_0}$	• Receive
		• $D(Q_k.x, C_0)$. Is t_u valid?
		• Calculate $K = H(Q_s, Q_u, g_s, g_u)$
		• Verify Sig_u, continue or abort.
		• Verify $Cert_u$, continue or abort.
		• Calculate $Sig_s = Sign(K)$
		• Assign new $I_{u.temp}$, notify HLR
		• $C_1 = E(Q_k.x, (Sig_s, cert_s, t_s,$
		$I_{u.temp}, S'))$
• Receive	$\xleftarrow{C_1}$	• Send
• $D(Q_k.x, C_1)$. Is t_s valid?		
• Verify Sig_s, continue or abort.		
• Verify $Cert_s$, continue or abort.		
• Store $I_{u.temp*}$ as new $I_{u.temp}$.		
• $k_m = h(Q_k.x, g_s, g_u)_{msb-64}$		• $k_m = h(Q_k.x, g_s, g_u)_{msb-64}$
• k_m: The unique secret key		• k_m: The unique secret key

Figure 5.5. Certificate Verification Steps.

USER	SERVER
• Given $cert_s = (r_s, s_s)$, Q_{ca}	• Given $cert_u = (r_u, s_u)$, Q_{ca}
Compute	Compute
• $c = {s_s}^{-1}$	• $c = {s_u}^{-1}$
• $u_1 = c \cdot e_s$	• $u_1 = c \cdot e_u$
• $u_2 = c \cdot r_s$	• $u_2 = c \cdot r_u$
• $R = u_1 \times P + u_2 \times Q_{ca}$	• $R = u_1 \times P + u_2 \times Q_{ca}$
• $v = R.x$	• $v = R.x$
• If $v \neq r_s$, then abort	• If $v \neq r_u$, then abort

digital signature verification algorithm (ECDSAV) is given in Figure 5.5. The server also includes a new user temporary identity in C_1 to be used for the user's next service request. Assigning this temporary identity is considered legal since the user has successfully authenticated itself at this point. Next, the server forward this new temporary identity to the user's home network. The reason for encrypting the certificates and the other related data is given in the protocol's security discussion section.

The user decrypts C_1 and obtains the certificate of the server and the challenge response Sig_s. Next, the user verifies the server's signature. Depending on the verification result, it may continue or abort the communication. Then, it checks the validity of the server certificate, and if it is not valid, the user aborts the communication.

Note that it may be a good idea to generate multiple g values in advance so that the protocol could save some time. However, storing these multiple random numbers will increase the storage requirement of the protocol, which is undesirable.

5.4.7 Challenge

The authentication process consists of a challenge and response dialog that runs between the HLR/AC and the wireless device. In our protocol two types of challenge defined:

- **Unique Challenge**: The challenge occurs during call attempts only, because it uses the voice channel. Unique challenge presents an authentication to a single phone during call origination and call delivery.
- **Global Challenge**: The challenge occurs during registration, call origination and call delivery. Global challenge presents an authentication challenge to all MSs that are using a particular radio control channel. It is called global challenge because it is broadcast on the radio control channel and the challenge is used by all mobile users accessing that control channel.

5.4.8 Key Agreement

Please refer to the section 4.4.4. for the details of generating a unique session key to encrypt the voice and data communication. Obtaining the session key in this unique way is discussed in section 5.5.2 from the security point of view.

5.4.9 End-to-End User Security Protocol

So far in this chapter we concentrated on the authentication and key agreement protocols between the user and the server. Either in call originations or in call terminations mobile users are now able to securely communicate over an authenticated wireless channel. We now consider the inter-domain end-to-end authentication and key agreement protocols. The process of establishing a secure and authenticated channel between two mobile users is summarized in Figure 5.6. Let us make the assumption that mobile station 1 is the initiating party and the mobile station 2 is the initiated party. Upon completing the protocol given in Figure 5.4 with the

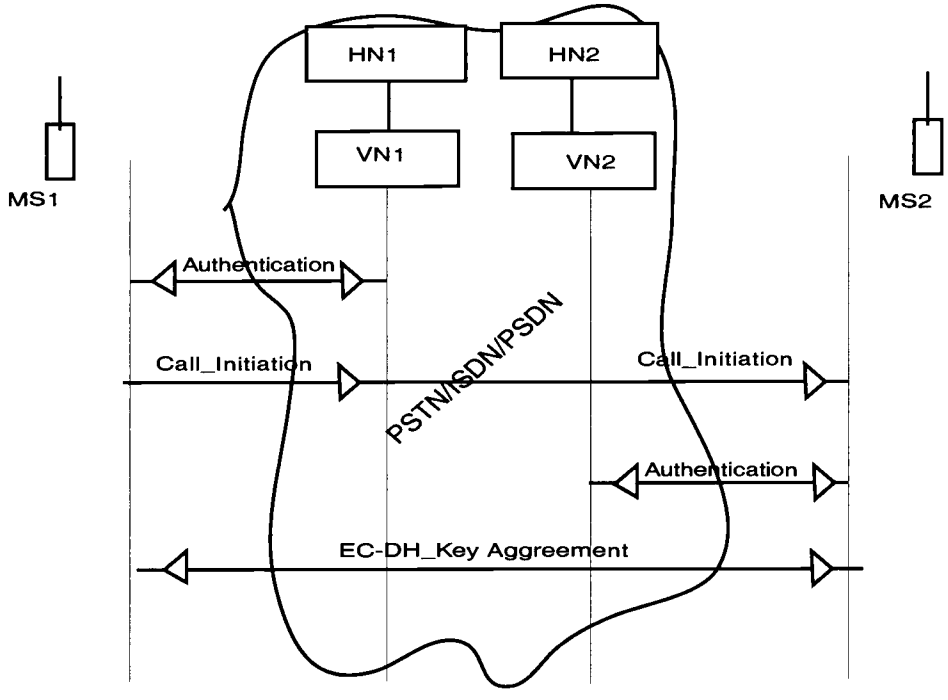
closest VN, MS1 request a channel to be set up with MS2. Therefore, MS1 and VN1 do not perform the session key agreement given in Figure 5.4 since MS1 intends to have a private communication with MS2, not with the server in VN1. Note that the location of MS2 is known by the mobile network and is stored in the VLR and/or HLR of the network, which MS2 is in.

There is a couple of issues at this point. Does VN1 need to forward the certificate of MS1 to MS2 through VN2 so that MS2 can also be sure about the real identity of MS1? Or, does VN1 and VN2 need to establish some kind of trust so that they permit the call forwarding and allocate the channels for the service? Though both issues can easily be solved with the proposed design methods, it is our assumption that if there is a roaming agreement between VN1 and VN2, they also have some kind of trust management procedure and therefore they are responsible for securely forwarding calls and service request to each other.

The authentication process between the initiating party MS1 and VN1 is given in Figure 5.4 with the exception that there is no session key generated. The authentication process between the initiated party MS2 and VN2 is as same as the previous one except that VN2 also forwards the public key of VN1 (Q_{s1}) to MS2. MS1 and MS2 should now have a session key for their privacy in their voice or data communication. The process of generating a session key between two users under the observation of the network server without revealing the secret to the third parties including the participant server is given in Figure 5.7. The outsider threats such as adversaries and eavesdroppers on the wireless channel and the insider threats such as adversaries and interception attackers on the fixed network between VN1 and VN2 are capable of intercepting the parameters $Q_s, Q_s d_2, Q_s d_1$, however incapable of calculating the session key value $K = Q_s d_2 d_1$. To be more precise it is infeasible to calculate K given the other parameters. Readers should refer back to section 5.4.4 for the estimated cost and time requirements to solve a discrete logarithm problem over a given elliptic curve for different key sizes.

At this point, MS1 and MS2 have successfully completed the necessary authentication and key agreement procedures. Some very sensitive and very high priority

Figure 5.6. End-to-end User Authentication Flow Diagram.



applications may require users to verify each other's certificates. Having generated a common secret, the users can now exchange their certificates in an encrypted form. The certification verification steps are shown in Figure 5.5.

The protocol flow chart is given in Figure 5.8. The subject that we did not address yet is the authorization issue. Authorization basically indicates what a user can and can not do if given a service for an application. To give an example, consider a group of people working on a project in a company. The need for a secure authenticated access to the project files is obvious. However, some members of the group may have *write* and *alter* access to certain files while some others may only be allowed to *read* the files. This task can be accomplished by the use of special purpose certificates that contain reserved sections such as *redelagation* and *rights*. The Simple Public Key Infrastructure [18] provides such certificates.

Figure 5.7. Elliptic Curve DH between End-to-End Users.

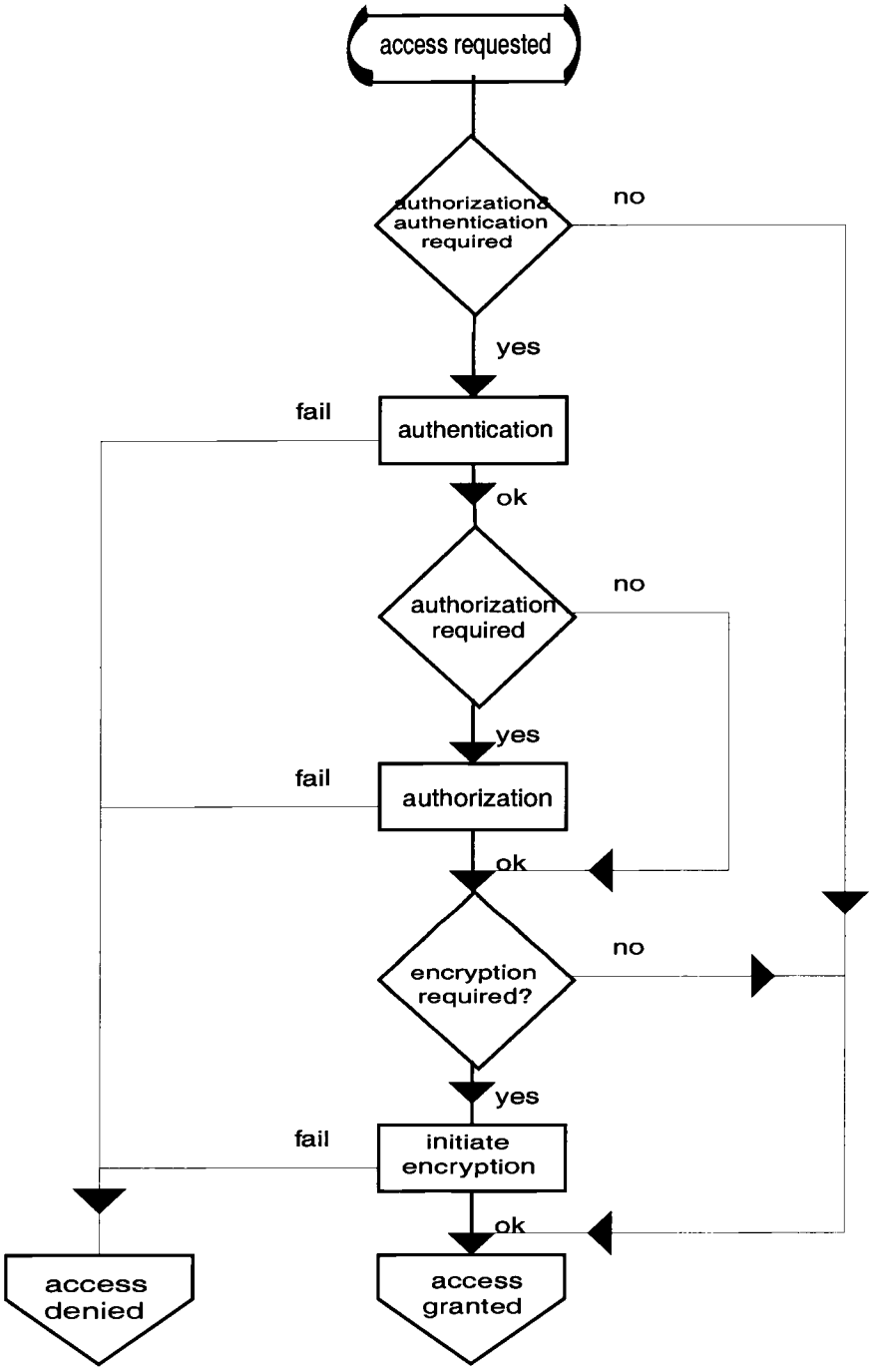
MOBILE 1	SERVER	MOBILE 2
• Receive Q_s	$\overset{Q_s}{\longleftrightarrow}$	• Receive Q_s
• Calculate $Q_s d_1$		• Calculate $Q_s d_2$
• Send $Q_s d_1$		• Send $Q_s d_2$
• Receive $Q_s d_2$	$\overset{Q_s d_2, Q_s d_1}{\longleftrightarrow}$	• Receive $Q_s d_1$
• Calculate $K = Q_s d_2 d_1$		• Calculate $K = Q_s d_1 d_2$

The other issues can be addressed as follows:

- **Crossing Domain Boundaries**

  In a highly-dynamic wireless environment where users frequently cross domain boundaries in the middle of communication, it is crucial to transfer the necessary state between domains in a manner transparent to user. The same problem also occurs when users migrate among different cells within the same domain. However, in the latter case, authentication is not an issue [62]. In GSM, the home domain issues the foreign domain A with a set of challenge/response pairs. Each pair is good for one-time authentication of the user. The foreign domain A is allowed to forward an unused pair to the foreign domain B as the user travels to the domain B. This lets the domain B to authenticate the user (or, rather, the user's SIM) without having to contact the home domain [62]. This is one-time only. The next time the user request an access from the domain B, a fully-fledged authentication procedure involving the home domain will take place. A similar approach can be applied to our protocols. While crossing domain boundaries in the middle of a communication, the associated VN or HN will authenticate the user to the new network on its behalf.

Figure 5.8. Protocol Flow Chart.



- **Visiting other networks**

  We assume here that each network maintains a database of public keys of other networks with whom a roaming agreement has been set up. The certificate of the MS is prepared and signed by either its home network or by a single trusted certification authority, CA , while each network's certificate is issued by a CA.

- **Operating with Multiple CAs**

  For a large network, a single CA cannot serve all the nodes. In this case, a hierarchy of CAs is employed [12]. X.509 [72], for example, describes such CA hierarchies. The classical certificates in the *Public Key Infrastructures (PKIs)* are verified by verifying each CAs certificate given in the certificate path. This brings significant computational complexity to mobile devices. However a recent proposal [48] describes a new *PKI* called the *Nested PKI (NPKI)*. This requires just one verification of the root certificate along with the hash content comparisons of the other CAs certificates. However we should also note the mobile network design should require mobile devices verify only the serving network's certificate for efficiency reasons. This means all the mobile service providers with a roaming agreement should agree on a trusted CA for creation of their certificates. This will enable mobile users to verify only one signature rather than verifying multiple certificate paths.

5.5 Protocol Analysis

In this section, we discuss the protocol's achievement on the proposed goals and the protocol's strength to possible attacks.

5.5.1 Protocol's Achievement on the Proposed Goals

- Mutual authentication of the server and the user:

  Both the server and the user challenge each other with fresh random numbers, and request the other party to sign this challenge. They also force each other

to encrypt the signed response with a key that can only be created by the real public key holders. They also verify each other's certificate by using the trusted CA's public key. These certificates contain the hashed universal identities and the public keys of the server and the user. By the time they come to the point of generating a unique session key, both parties are sure about the other party's identity.

- Agreement between users and servers on a secret authentication key to protect the data used in mutual authentication process:

  According to our protocol the users are assigned a temporary identity for each call. This of course happens after a mutually authenticated session with the servers. Our assumption is that there is a physical and secure connection between the visited network and the user's home network. Therefore, whenever provided with a $I_{u.new}$, the server knows that this is either a call initiation or a presence acknowledgement. In either case, it is the visited server's task to decode $I_{u.new}$ to obtain route code to the user's home network. By using $I_{u.new}$, the server acquires the public key and the universal identity I_u of the user. We also assume that a visiting device knows or acquires the server's public key. Now, both parties can compute a key to hide the certificates, the challenge response, and any other related data without giving away any clues to adversaries on the channel. All adversaries see during the whole communication session is just the public key and the identity of the server, the temporary identity of the user, and a challenge number by the server. Breaking the key with this amount of knowledge is just not feasible. The computation of this key is given in Figure 5.3.

- Non-repudiation of origin by the user and the server for relevant data sent from the user to the server and from the server to the user:

  Non-repudiation of origin by user for relevant data is usually provided by performing on-line signature operations by users. The idea, here, is that these signatures contain the private keys of the users, which are only known by

them. In our protocol, on-line signature generation is performed by both the user and the server on newly generated random numbers along with additional data. Therefore the protocol achieves the desired goal. One can always claim that he did not make the call or receive a service, however, the fact that his private key has been involved during process to sign messages can always be proved by the recorded data at the server site.

- Agreement between the user and the server on a secret session key, which will be used to encrypt data sent by each party:

  After the authentication process, each party has a couple of fresh random challenge numbers, g_u and g_s, and the initial mutual authentication key $Q_k.x$. An eavesdropper is only capable of knowing g_s since this is the only number sent in plaintext form. In the protocol, both parties create a unique session key k_m as given in Figure 5.4.

- Interoperability :

  The protocol allows negotiation of the symmetric key algorithm. This allows for enhancing the protocol in the future to better algorithms as well as interoperability between domestic and foreign versions of the product if they require different encryption algorithms for exportability purposes.

- User identity confidentiality: Hiding the identity of the portable device from an eavesdropper on the communication channel:

  In our protocol, the user's public key, the universal identity and any other user sensitive data are never revealed in open-air interface. In addition, the user is assigned a new temporary identity after each call. This temporary identity is sent to the user in an encrypted form. Therefore, analyzing call traffic patterns and the temporary identities will not help the adversaries at all. This protocol can also be modified to hide the real identity of the users from the visited network. The visited network may just play the role of passing the information between the user and the user's home domain. The home network

then has to authorize the visited network to give service to the user and has to manage the billing and tracing operations. As long as the home and the visited networks have a strong and secured connection, considering the capabilities of the networks compared to the user's this is a reasonable assumption, it can always be done.

5.5.2 Protocol's Security Analysis

The previous section should give some insight to security related concerns. Here, we summarize the possible attack scenarios and the protocol's resistance to them.

- While roaming, a visited network cannot know the next session key until a visiting user makes a request. Possible exposure [51] of the roamer's security parameters stored in the visited network as found in GSM, DECT, and USDC is thus eliminated.

- The mutual authentication is effectively performed and the security of the resulting protocol can be formally proved using BAN logic [52]. For the cases of mobile station registrations and terminations, the protocol is secure unless the attacker can compromise either the user or the server and at the same time break the public key algorithm, which means solving the elliptic curve discrete logarithm problem.

- The initial registration process is shown in Figure 5.3. Note that it is only the legal server that can associate the $I_{u.temp}$ to Q_u through a secured wired channel with the home network. If the user does not know the public key of the visited domain server, it can tune in to a previously defined frequency or format to acquire the frequently broadcast key. An attacker's attempt to send the legal server's public key is fruitless. The user may not know that it is an attacker who sent the Q_s, but the attacker does not gain anything either because he does not have the associated private key of the server to create the mutual key.

- Assuming the public keys are valid and belong to the key holders, the initial encryption/decryption key can only be generated by the real user/server.

- The certificates are usually sent in clear in almost all the other authentication protocols. In our protocol, we do not reveal the content of the certificate, which may be useful for spoofing attacks. This increases the encryption time only slightly since the certificate is not very long (on the order of kilobits) and the encryption algorithms are very fast (on the order of megabits per second).

- The user decrypts C_1 and obtains the certificate of the server and the challenge response Sig_s. Verifying the signature from the server confirms the freshness of the message and prevents the reply attacks. Next, the user checks the validity of the certificate, and if it is invalid, the user aborts the communication. This mechanism, specifically the use of g_s, defeats *spoofing* attacks by the user side and also prevents unnecessary computation. Next, the server checks the validity of the certificate and accordingly grants or aborts the service.

- The user's response to the server's challenge is Sig_u. Note that the user not only signs the challenge g_s, but also signs its own challenge g_u. It also includes g_u in C_0. The server first decrypts C_0, and obtain g_u. Then verifies the user's signature. The user can never claim he did not choose g_u in this scenario. And the server has one more assurance that g_u is generated by the real user. Including the public keys of both parties in the signatures is just another precautionary step to ensure each party that they are really communicating with a party for whom they have really intended to.

- The unique session key is generated by performing a one-way function on the previously obtained data. Both parties contribute same amount of entropy in the generated key. The reason for using a one-way hash function is that for any reason if the key is broken by an attacker, he should not able to run the protocol backwards to obtain useful data. That is the key should not reveal any information about the special format or the number of 1's or 0's of the challenge

number g_u and the initial mutual authentication key $Q_k.x$. $Q_k.x, g_u, g_s$ are all 64-bit numbers. The adversary only knows g_s. The output of the hash function is 160-bit long. The key is the most significant 64 bits of the hashed output.

- Many existing private and public key protocols were designed in a way that after each authentication the certificates should be destroyed. This is to prevent network personnel from obtaining the certificate by impersonating the mobile user. There is no such threat in the protocols defined here. Acquisition of a certificate is just not enough to impersonate a subscriber.

- Forward Secrecy:

  The protocol also provides for good forward secrecy, meaning that if the private keys of either the base or the mobile should be compromised at some point in the future, this does not necessarily compromise the wireless link data that has been exchanged between the party whose private key is compromised and the other party. The protocol requires the compromise of both the base and the terminal in order to compromise communication between them without breaking the encryption key. ECC key pair generation, given a valid set of domain parameters, is about as simple as possible to conceive, while RSA key pair generation is much more complex [40]. This means that achieving the security attribute of forward secrecy is relatively straightforward when using ECC, but is more complicated to achieve and suffers a significant performance cost when using RSA.

5.6 Comparisons

For the sake of obtaining some comparison results, once again we compare certain properties of this improved protocol to those of Aziz-Diffie [12] and Beller-Chang-Yacobi [16] protocols. The comparison is based on the mutual authentication and key agreement between the user and the server. This means we do not consider the end-to-end user protocol as the comparison basis.

The performance values for the Aziz-Diffie and Beller-Chang-Yacobi protocols are found in [13], summarized in Figure 5.9.

- The proposed protocol requires less bandwidth compared to other public-key based protocols, e.g., Aziz-Diffie and Beller-Chang-Yacobi protocols. The total number of bits exchanged in the real-time portion of the protocols is given as follows:

Beller-Chang-Yacobi:	8320 bits (1024-bit key)
Aziz-Diffie:	8680 bits (1024-bit key)
Proposed:	1888 bits (160-bit key)

  In our protocol, the parameter lengths are as follows:

Q_u, Q_s:	$161 - bit$
e_u, e_s:	$160 - bit$
$Cert_u = (r_u, s_u), Cert_s = (r_s, s_s)$:	$320 - bit$
$I_u, I_s, t_u, t_s, g_u, g_s, S, S'$:	$64 - bit$

- The proposed protocol has low storage requirements for the user side, which makes it suitable for smart cards and other handheld computing devices:

Beller-Chang-Yacobi:	5120 bits (1024-bit key)
Aziz-Diffie:	2176 bits (1024-bit key)
Proposed:	1440 bits (160-bit key)

- The proposed protocol has modest computational load on the user side for real-time execution:

Beller-Chang-Yacobi:	2 PKE (1024-bit) + 1 PKD (1024-bit)
+ Precomputation	
Aziz-Diffie:	3 PKE (1024-bit) + 2 PKD (1024-bit)
Proposed:	1 ECDSA (160-bit) + 2 ECDSAV (160-bit) +
	2 SKE (848-bit data)

Figure 5.9. Parameter Lengths (in bits) in Beller-Yacobi and Aziz-Diffie.

Beller-Chang-Yacobi		Aziz-Diffie	
Hashed certificate	384	Exchanged messages	8680
Public key	1024	Public key	1024
Certificate content	1536	Hashed messages	1030
Random number	1024	Random number	1024
Random challenge	128	Identity	512

5.7 Conclusions

We have described an end-to-end authentication and key agreement protocol for wireless communication based on elliptic curve cryptographic techniques. The possible application areas of the proposed protocol are smartphones with a micro browser using XML based mobile services, personal digital assistants (PDAs) for retrieving e-mails, laptops accessing to a wireless network (WLAN), digital cameras with a GPS module for authenticated crime scene pictures exchanged with police labs over a wireless link, smartcards to access to a database for financial transactions and lastly but most importantly mobile phones.

The proposed protocols given in this chapter can be simplified for certain environments, which do not require a high-level security. For example for those environments in which masquerading and impersonation are not a big threat or not harmful, or too expensive to be avoided, ECC-DH key agreement can be deployed to provide confidentiality rather than implementing the authentication scheme. In this case the user should know the public key of the service provider in that area in advance. This type of protocol will require an elliptic curve point multiplication on the user side which can be done very fast compared the requirement in full authentication protocol in which the user has to perform one ECC signature generation, two ECC signature verifications.

The protocols given in this chapter provide mutual authentication, non-repudiation, user identity confidentiality (hiding the identity of the user from the adversaries on the wireless channel), a unique session key for each data, voice or video communication session. The users neqotiate on a secret key under the supervision of the server providers. However, these servers on the fixed channel do not get to see the exchanged data in cleartext. Therefore, a total confidentiality of the end-to-end user communication is supported. So insider and outsider attacks on the message integrity and confidentiality are fruitless.

Chapter 6
IMPLEMENTATION ISSUES OF ECC

" I am sitting here 93 million miles from the sun on a rounded rock which is spinning at the rate of 1000 miles an hour... and my head pointing down into space with nothing between me and infinity but something called gravity which I can't even understand, and which you can't even buy any place so as to have some stored away for a gravityless day... "

Russell Baker

6.1 Introduction

Elliptic curve cryptography provides a methodology for obtaining high-speed, efficient, and scalable implementations of network security protocols. In this chapter, we describe in detail implementation related issues, and the results of our implementation of the elliptic curve cryptography over the Galois field $GF(2^k)$, where $k = n.m$ is a composite number.

6.2 Background

6.2.1 Rings

A ring R is a set, whose objects can be added and multiplied, satisfying the following conditions

- Under addition, R is an additive (abelian) group.
- For all $x, y, z \in R$ we have,

$$x(y+z) = xy + xz; \tag{6.1}$$

$$(y+z)x = yx + zx. \tag{6.2}$$

- For all $x, y \in R$, we have $(xy)z = x(yz)$.
- There exists an element $e \in R$ such that $ex = xe = x$ for all $x \in R$.

The integer numbers, the rational numbers, the real numbers and the complex numbers are all rings.

An element x of a ring is said to be invertible if x has a multiplicative inverse in R, that is, if there is a unique $u \in R$ such that: $xu = ux = 1$. 1 is called the *unit element* of the ring.

6.2.2 Fields

A field is a ring in which the multiplication is commutative and every element except 0 has a multiplicative inverse. We can define the field F with respect to the addition and the multiplication if:

- F is commutative group with respect to the addition.
- $F \setminus \{0\}$ is a commutative group with respect to the multiplication.
- The distributive laws mentioned for rings hold.

6.2.3 Finite Fields

A finite field or *Galois field* denoted by $GF(q = p^n)$, is a field with characteristic p, and a number q of elements. Such a finite field exists for every prime p and positive integer n, and contains a subfield having p elements. This subfield is called *ground field* of the original field. For every non-zero element $\alpha \in GF(q)$, the identity $\alpha^{q-1} = 1$ holds. Furthermore, an element $\alpha \in GF(q^m)$ lies in $GF(q)$ itself if and only if $\alpha^q = \alpha$.

For the rest of this thesis, we will consider only the two most used cases in cryptography: $q = p$, with p a prime, and $q = 2^m$. The former case, $GF(p)$, is denoted as the prime field, whereas the latter, $GF(2^m)$, is known as the finite field of characteristic two or simply binary field.

6.3 Implementation Related Issues

The speed of the elliptic curve operations, e.g., the point addition and point multiplication, depends on the arithmetic of the underlying finite field. The drafted IEEE standard proposes the use of the fields $GF(p)$ and $GF(2^k)$. The implementation of the field $GF(p)$ requires that we implement modular arithmetic with respect to a prime modulus p. Due to the security requirements, the size of p is at least 100 bits, usually around 160 bits. The large number arithmetic has been extensively studied in the context of the RSA algorithm, and efficient algorithms for field multiplication have been designed [45]. An efficient method for performing the field multiplication is the Montgomery method [58] which effectively performs modulo 2^k multiplication instead of modulo p multiplication, where $2^k > p > 2^{k-1}$. Selection of a Mersenne prime (a prime of the form $2^k - 1$) or a prime in the special form of $2^k - c$, where c is a small odd integer has also shown to be useful [23]. With the use of these methods, it is possible to obtain high-speed implementations of the elliptic curve cryptographic algorithms. However, the arithmetic operations in $GF(p)$ in hardware are significantly slower than those in $GF(2^k)$. For example, a 160-bit elliptic curve point multiplication operation requires about 200 milliseconds on a 200 MHz Pentium II processor.

Due to the high latency in hardware implementations over $GF(p)$, a better choice is to use the field $GF(2^k)$ as the underlying field, where there is more room for improvement. The carry free nature of the field $GF(2^k)$ yields higher performance and less area consumption. Moreover, the structural properties of the field $GF(2^k)$ can be exploited to further improve the performance. The use of special irreducible polynomials greatly improves the efficiency. All-one-polynomials and trinomials are among the most popular irreducible polynomials. The basis of the representation is also an important factor. The optimal normal basis is especially popular in hardware implementations. For example, it was reported in [55] that Newbridge Microsystems in conjunction with Cryptech Systems implemented a chip which is able to perform public-key operations over the field $GF(2^{593})$. The clock rate of this chip is 20 MHz,

and it can perform a field multiplication in 0.065 ms and a field inversion in 2.5 ms. The elliptic curve point multiplication operation with the use of the projective coordinate system was estimated to be less than a second on this chip. Another interesting implementation was reported in [5] for the field $GF(2^{155})$. In the design, an irreducible trinomial was used to perform field multiplications. With a clock rate of 40 MHz, the chip performs the multiplication and inversion operations in about 0.004 ms and 0.095 ms, respectively. Using these timings, we estimate the computation time for the elliptic curve point multiplication as around 20 ms.

Another approach would be to use a standard signal processor or a microcontroller, which provides flexibility since the size of the field can be changed in software more easily than in hardware. Recently, there is a growing interest to develop high performance software libraries for elliptic curve cryptography. Software designs achieving timings under 20 ms for the field $GF(2^{176})$ on a 200 MHz Pentium II processor have been reported in [71]. Recently a practical implementation of elliptic curve crytosystems over $GF(p)$ on a 16-bit microcomputer was performed by Hasegawa, Nakajima and Matsui [36]. They report that their implementation of a cryptographic library supports 160-bit elliptic curve DSA (ECDSA) signature generation, verification and SHA-1 on the processor. This library also includes general integer arithmetic routines for applicability to other cryptographic algorithms. The ECDSA signature generation and verification timings of their implementation are given as 150ms and 630ms, respectively.

6.4 Elliptic Curves over Composite Fields

An important category of elliptic curve cryptographic algorithms is defined over the finite field $GF(2^k)$. Elliptic curve cryptographic applications require fast hardware and software implementations of the arithmetic operations in $GF(2^k)$ for large values of k. Recently, there has been a growing interest to develop software methods for implementing $GF(2^k)$ arithmetic operations and elliptic curve cryptographic operations [68, 71]. An area of particular interest is the development of efficient software

implementations of the arithmetic and elliptic curve operations in $GF(2^k)$, where k is a composite number as $k = nm$. An implementation method for this case was presented in [71], where the authors propose to use the logarithmic table lookup method for the ground field $GF(2^n)$ operations. The field $GF(2^{nm})$ is then constructed using the polynomial basis, where the elements of $GF(2^{nm})$ are polynomials of degree $m-1$ whose coefficients are from the ground field $GF(2^n)$. The field multiplication is performed by first multiplying the input polynomials, and reducing the resulting polynomial by a degree-m irreducible trinomial.

Here, we propose a similar methodology for implementing the arithmetic operations in $GF(2^{nm})$. Our main difference is that we use an optimal normal basis in $GF(2^m)$ to represent the elements of $GF(2^{nm})$ by taking the ground field as $GF(2^n)$. The resulting field operations, multiplication and squaring are quite efficient, and they do not involve modular reductions. Our implementation results indicate that the arithmetic operations in the proposed method are faster than those of [71]. Detailed algorithms for performing field multiplication and inversion operations are reported in [66]. Here, we give a brief overview of our methodology, and report the timing results for the finite field and elliptic curve operations in the special case of $k = 176$, $n = 16$, and $m = 11$.

It is customary to view the finite field $GF(2^k)$ as a k-dimensional vector space defined over the field $GF(2)$. The field over which the vector space defined is generally called the ground field. If we take the ground field as $GF(2)$, then the elements of the k-dimensional vector space are bit strings of length k, i.e., $A = (a_0, a_1, \ldots, a_{k-2}, a_{k-1})$, where $A \in GF(2^k)$ and the entries $a_i \in GF(2)$. However, if $n > 1$ divides k, then it is also possible to select the ground field as $GF(2^n)$. If we take $nm = k$, then, effectively we are constructing an m-dimensional vector space over $GF(2^n)$. The elements of the ground field are represented as bit-strings of length n; the elements of $GF(2^{nm})$ are represented as

$$A = (A_0, A_1, \ldots, A_{m-2}, A_{m-1}) \ , \tag{6.3}$$

where the entries $A_i \in GF(2^n)$. These representations, the nm-dimensional vector

space over $GF(2)$ and the m-dimensional vector space over $GF(2^n)$, are equivalent. However, the latter representation is based on n-bit words, and it is more advantageous for implementation on microprocessors, particularly when n is selected properly. For example, $n = 8$ and $n = 16$ have been suggested in [35, 71].

The methodology in [71] uses the values of n as 8 or 16. A basis for $GF(2^8)$ or $GF(2^{16})$ can be chosen; however, this is not important since the field arithmetic is performed using the logarithmic table lookup method. An element of $GF(2^{nm})$ is represented using a degree-$(m-1)$ polynomial whose coefficients are from the field $GF(2^n)$. The multiplication in $GF(2^{nm})$ is performed by first multiplying the operands to obtain the twice-sized polynomial, and then reducing the resulting polynomial by a degree-m irreducible polynomial. In general, this degree-m irreducible polynomial needs to have its coefficients from the field $GF(2^n)$, however, it is well-known [50] that an irreducible polynomial over $GF(2)$ of degree m remains irreducible over $GF(2^n)$ if and only if $\gcd(n, m) = 1$. Therefore, one can select an irreducible polynomial of degree m with coefficients from $GF(2)$ rather than $GF(2^n)$ if $\gcd(n, m) = 1$. Furthermore, the use of a special irreducible polynomial is suggested in [71]. This special polynomial is a trinomial of the form $x^m + x^t + 1$, where $t \leq \lfloor m/2 \rfloor$. Such special trinomials are easy to find; the paper [71] lists them for a few values of m.

Our methodology is similar to the one proposed in [71] in terms of representing the elements of and performing the arithmetic in $GF(2^{nm})$. We also select n as 8 or 16. However, we represent the elements of $GF(2^{nm})$ using an (optimal) normal basis for the field $GF(2^m)$. In this representation, the elements of $GF(2^{nm})$ are m-dimensional vectors whose entries are in $GF(2^n)$. A degree-m irreducible polynomial is selected such that the root of this polynomial generates an optimal normal basis [60]. Furthermore, if $\gcd(n, m) = 1$, then the selected degree-m irreducible polynomial will have its coefficients from $GF(2)$ rather than $GF(2^n)$. This selection provides a much simpler multiplication method. Since $\gcd(n, m) = 1$ and $n = 8$ or 16, we need to have an odd m. This requires that we use an optimal normal basis of

type II in the field $GF(2^m)$. For optimal normal bases of type I, m would be even because $m + 1$ is a prime number.

The normal bases are thought to be inefficient for composite fields in this setting [71]. The advantages of (optimal) normal bases seem to disappear for $n > 1$, particularly for squaring operation. However, it is our conclusion that if the ground field operations are computed fast (e.g., using the table lookup method), then the composite field operations can be performed very efficiently in the normal basis. The mathematical and algorithmic details of our methodology can be found in [66].

6.5 Implementation Results

We have implemented the addition, multiplication, and inversion operations in $GF(2^{176})$, and also the elliptic curve point doubling, addition, and multiplication operations over $GF(2^{176})$. The programs were written in C++ using Microsoft Visual C++ Version 5.0, and executed on a PC with the 300-MHz Pentium II processor, running Windows NT 4.0. Our timing results are given in the first column of Table 6.1. The elliptic curve operations are performed using the affine coordinate system in which a point is represented using two field elements $P = (x, y)$. The field parameters $a, b \in GF(2^{176})$ are selected randomly. The point multiplication algorithm uses the canonical recoding binary method in which the signed-digit recoding of the 176-bit randomly chosen integer e is used.

We also include the timing results of [71] for comparison. The original timing results of [71] were obtained on a 133-MHz Pentium. We have re-developed the programs of [71] by investing reasonable efforts to optimize the code. The timing results given in the second column are our reproduction of their method. Since the 300-MHz Pentium II processor is about 2.25 times faster than the 133-MHz Pentium, it seems that our reproduction software of the methods of [71] is about 50 % faster than their implementation. In the third column, we also give their original timings on the 133-MHz Pentium processor.

Table 6.1. The Timing Results for the Field and Elliptic Curve Operations.

Operation	Our Method (300-MHz P-II)	Reproduced [71] (300-MHz P-II)	Original [71] (133-MHz P-I)
Field Multiplication	12 μsec	15 μsec	(62.7+1.8) μsec
Field Squaring	1.5 μsec	2.5 μsec	(5.9+1.8) μsec
Field Inversion	60 μsec	63 μsec	160 μsec
EC Addition	80 μsec	83 μsec	306 μsec
EC Doubling	80 μsec	85 μsec	309 μsec
EC Multiplication	25 msec	30 msec	72 msec

We have obtained all our timings by actual implementation. The results in Table 6.1 shows that the proposed methodology is faster than our reproduction of their method [71], and about 50 % faster than their reported timings, taking into account the speed difference of the processors.

6.6 Conclusions

We provided the results of our implementation of elliptic curve cryptography over the field $GF(2^k)$, where k is a composite number $k = nm$. This implementation uses table lookup techniques to perform the operations in the ground field $GF(2^n)$, while the large field is constructed using the optimal normal basis. The reported timing results only contain the arithmetic operations and the elliptic curve operations. The timings of an entire protocol can be estimated by using these values. In the next chapter we will present the timing performances of the proposed protocols on a microprocessor.

Finally, we note that certain attacks on elliptic curve cryptosystems over composite fields $GF(2^{nm})$ have recently been developed [29, 70]. These attacks are applicable to elliptic curves whose coefficients are defined in the ground field $GF(2^n)$,

providing the speedup factor of $\sqrt{2m}$. Furthermore, these attacks are highly efficient on so-called binary anomalous curves, in which the curve coefficients are restricted to the field $GF(2)$. In our proposed implementation, we do not make any assumption that the curve coefficients would be restricted to the ground field $GF(2^n)$. In fact, the timing results summarized in Table 6.1 are obtained by selecting the parameters a and b randomly in the large field $GF(2^{176})$.

Chapter 7
IMPLEMENTATION OF THE PROTOCOLS OVER GF(P)

" Who are we? We find that we live on an insignificant planet of a humdrum star lost in a galaxy tucked away in some forgotten corner of a universe in which there are far more galaxies than people. "

Carl Sagan

7.1 Introduction

The need for information security in today's digital technology has been growing so fast that it is necessary to implement cryptographic security techniques in almost every digital electronic system including digital cellular communication systems, wireless or wired digital computer networks, smart cards, etc. Today's cryptographic techniques are mostly based on public key cryptosystems due to their implementation efficiency and easier key management protocols.

The rapid progress in wireless mobile communication technology, personal communication systems and smart card technology has brought new challenges to be met by engineers and researchers who work and study the security aspects of these new technologies. Using software public key cryptosystems on high speed hardware platforms such as PCs rarely brings problems such as excessive computation times and code size. However in restricted hardware environments with limited computational power and small memory such as smart cards and cellular phones, these factors become significantly important. The need for a public key cryptosystem with higher strength-per-bit is clear. The benefits of this higher strength-per-bit include higher speeds, lower power consumption, bandwidth savings, and smaller certificates. These advantages are particularly beneficial in applications where bandwidth, processing

capacity, power availability, or storage are constrained (including chip cards, electronic commerce, web servers, cellular telephone, and pagers). The obvious answer for this kind of beneficial public key cryptosystem is Elliptic Curve Cryptography (ECC).

Elliptic curves have been used to factorize numbers and in primality proving. More recently, elliptic curves attracted attention for their part in the solution to Fermat's Last Theorem. However, elliptic curves have also inspired a new type of cryptosystem providing alternatives for implementation of digital signatures, key exchange and confidentiality [55]. As mentioned earlier, ECC requires a much shorter key than that required by RSA. Based on the current knowledge, the key size required to obtain acceptable security level is about $160 - 180$ bits for ECC while it is 1024 or more bits for *RSA*.

Certicom's SigGen smart card [20] is a good example of ECC software implementation over a field of characteristic 2. SigGen is a prototype smart card with 8-bit microprocessor that generates digital signatures using a conventional I/C from Motorola (68SC28). Developed in cooperation with Schlumberger, SigGen combines Multiflex card technology with the Certicom Elliptic Curve Engine, resulting in strong, fast, public-key operations. This card demonstrates that effective digital signature applications can be implemented on standard processors: Digital signatures are generated in less than 600ms while using only 90 Bytes of RAM. It has been implemented in less than 4K-code space. SigGen is ideally suited for applications requiring end-user identification and strong authentication. A cryptographic co-processor is not required because (CE) permits the efficient use of elliptic curves for digital signatures in small footprint environments such as smart cards.

Another practical implementation of elliptic curve cryptosystems over $GF(p)$ on a 16-bit microcomputer was introduced by Mitsubishi's Information Technology R&D Center (Hasegawa et al.) [36]. They have designed a practical cryptographic library, which supports EC arithmetic operations, EC digital signature generation and verification and secure Hash Algorithm SHA-1. Their target processor was a Mitsubishi's 16-bit microcomputer M16C (10MHz) which has been used in various

engineering applications such as mobile telecommunication systems (cellular phones, pagers, etc.) They designed two independent integer arithmetic modules: one is for executing modular arithmetic modulo a fixed prime characteristic p for high speed computation, and the other is for general integer routines that accept any positive integers with arbitrary length for wider applicability. The goal was here to support not only ECC but also RSA. They have reported a speed of $150msec$ for generating a 160-bit ECDSA signature and $630msec$ for verifying an ECDSA signature on M16C. Total code size was 4Kbyte including SHA-1.

Our goal is to design a practical and scalable cryptographic library that supports ECDSA signature generation and verification. This library also contains SHA and DES algorithms that are necessary for the implementation of our authentication protocol. We implemented the protocol on a 32-bit ARM7TDMI microprocessor using the ARM software development toolkit. We achieved the timings of $46.4msec$ ECDSA signature generation and $92.4msec$ ECDSA signature verification for 160-bit ECC. We also calculated the total protocol execution timings, which are given in this chapter.

The chapter is organized as follows. In section 7.2, we present the details of arithmetic operations used in $GF(p)$. In section 7.3, we give a brief explanation of the proposed wireless authentication protocols, which were implemented in software to obtain timing results. In §7.4, we introduce a 32-bit ARM microprocessor. The software architecture, programming style and library structure is presented in §7.5. Finally, we show our timing results and possible enhancements in detail in §7.6.

7.2 Efficient Elliptic Curve Operations over $GF(p)$

The speed of the elliptic curve operations such as the point addition and point multiplication, depends on the arithmetic of the underlying finite field. The drafted IEEE standard proposes the use of the fields $GF(p)$ and $GF(2^k)$. The implementation of the field $GF(p)$ requires that we implement modular arithmetic with respect to a prime modulus p. Due to the security requirements, the size of p is at least 100 bits,

usually around 160 bits. The large number arithmetic has been extensively studied in the context of the RSA algorithm, and efficient algorithms for field multiplication have been designed [45]. An efficient method for performing the field multiplication is the Montgomery method [58] which effectively performs modulo 2^k multiplication instead of modulo p multiplication, where $2^k > p > 2^{k-1}$. In our implementation we use the field of $GF(p)$.

In the following we will summarize several different coordinate systems used to represent elliptic curves. This is important because for each system the total number of field multiplications is different resulting in different speed for point additions and doublings. We will now show the addition formulas in all these coordinates.

7.2.1 ECC Arithmetic Using Affine Coordinates

An elliptic curve over real numbers may be defined as the set of points (x, y) which satisfy the elliptic curve equation form:

$$y^2 = x^3 + ax + b \tag{7.1}$$

where x, y, a and b are real numbers. Note that the condition $a, b \in F_q$, $4a^3+27b^2 \neq 0$ where F_q is the underlined field over which the elliptic curve is defined should be met.

The addition formulas in *affine coordinates* are given as follow [37].
Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$ and $K = P + Q = (x_3, y_3)$ be points on elliptic curve $E(F_p)$.

- *EC addition formulas when* $(P \neq \pm Q)$:

$$x_3 = \lambda^2 - x_1 - x_2 \tag{7.2}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \tag{7.3}$$

where

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \tag{7.4}$$

- *EC doubling formulas when* $(P = Q)$

  Let $P = (x_1, y_1)$ a point on the curve. Then, $2(x_1, y_1) = (x_2, y_2)$, where

$$x_2 = \lambda^2 - 2x_1 \tag{7.5}$$

$$y_2 = \lambda(x_1 - x_2) - y_1 \tag{7.6}$$

  where

$$\lambda = \frac{3x_1^2 + a}{2y_1} \tag{7.7}$$

7.2.2 ECC Arithmetic Using Projective Coordinates

It may be useful to keep track of numerators and denominators in $GF(p)$ division operation if division within this field is relatively expensive [37]. This is used to avoid the costly modular inverse operation which is required in the *affine coordinates.* The *projective coordinates* are often used to reduce the number of modular inversions [36]. The *projective coordinates* X, Y, and Z are given by

$$x = \frac{X}{Z^2}, \tag{7.8}$$

$$y = \frac{Y}{Z^3} \tag{7.9}$$

Actually there are more kinds of projective coordinates; however, the one mentioned here provides the fastest arithmetic on elliptic curves [37].

The equations given above are used for converting a finite point from projective coordinates to affine coordinates. The formulas for converting it back to projective from affine are given by

$$X = x, Y = y, Z = 1$$

The addition formulas in *projective coordinates* are given as follows [36, 37]. Let $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$ and $K = P + Q = (X_3, Y_3, Z_3)$ be points on elliptic curve $E(F_p)$.

- *EC addition formulas when* $(P \neq \pm Q)$:

$$X_3 = (Y_1 Z_2^3 - Y_2 Z_1^3)^2 - (X_1 Z_2^2 + X_2 Z_1^2)(X_1 Z_2^2 - X_2 Z_1^2)^2 \tag{7.10}$$

$$A = (X_1 Z_2^2 + X_2 Z_1^2)(X_1 Z_2^2 - X_2 Z_1^2)^2 - 2X_3 \tag{7.11}$$

$$2Y_3 = A(Y_1 Z_2^3 - Y_2 Z_1^3) - (Y_1 Z_2^3 + Y_2 Z_1^3)(X_1 Z_2^2 - X_2 Z_1^2)^3 \tag{7.12}$$

$$Z_3 = Z_1 Z_2 (X_1 Z_2^2 - X_2 Z_1^2) \tag{7.13}$$

Or equivalently (another version of representing the equations);

$$U_1 = X_1 Z_2^2 \tag{7.14}$$

$$S_1 = Y_1 Z_2^3 \tag{7.15}$$

$$U_2 = X_2 Z_1^2 \tag{7.16}$$

$$S_2 = Y_2 Z_1^3 \tag{7.17}$$

$$W = U_1 - U_2 \tag{7.18}$$

$$R = S_1 - S_2 \tag{7.19}$$

$$T = U_1 + U_2 \tag{7.20}$$

$$M = S_1 + S_2 \tag{7.21}$$

$$Z_3 = Z_1 Z_2 W \tag{7.22}$$

$$X_3 = R^2 - TW^2 \tag{7.23}$$

$$V = TW^2 - 2X_3 \tag{7.24}$$

$$2Y_3 = VR - MW^3 \tag{7.25}$$

- *EC doubling formulas when* $(P = Q)$.

Let $P(X_1, Y_1, Z_1)$ be a point on the curve. Then, $2(X_1, Y_1, Z_1) = (X_2, Y_2, Z_2)$, where

$$X_2 = (3X_1^2 + aZ_1^4)^2 - 8X_1 Y_1^2 \tag{7.26}$$

$$Y_2 = (3X_1^2 + aZ_1^4)(4X_1Y_1^2 - X_2) - 8Y_1^4 \tag{7.27}$$

$$Z_2 = 2Y_1Z_1 \tag{7.28}$$

Or equivalently (another version of representing the equations);

Let $P(X_1, Y_1, Z_1)$ be a point on the curve. Then, $2(X_1, Y_1, Z_1) = (X_2, Y_2, Z_2)$, where

$$M = 3X_1^2 + aZ_1^4 \tag{7.29}$$

$$Z_2 = 2Y_1Z_1 \tag{7.30}$$

$$S = 4X_1Y_1^2 \tag{7.31}$$

$$X_2 = M^2 - 2S \tag{7.32}$$

$$T = 8Y_1^4 \tag{7.33}$$

$$Y_2 = M(S - X_2) - T \tag{7.34}$$

Before we go to next section it is quite important to note that in the case of $a = p - 3$ the number of field multiplications in elliptic curve addition is fewer than the normal case. For additional details please refer to P1363 Standard [37].

7.2.3 ECC Arithmetic Using Modified Jacobian Coordinates

As mentioned before the elliptic curve equation in *affine coordinates* are given by

$$y^2 = x^3 + ax + b$$

For *Jacobian coordinates*, x and y are set as $x = X/Z^2$ and $y = Y/Z^3$ respectively. The new equation, then, becomes the following form in $E_J(F_p)$

$$Y^2 = X^3 + aXZ^4 + bZ^6 \tag{7.35}$$

It is a known fact that the *modified Jacobian coordinates* provides the fastest possible doublings. The addition formulas for both Jacobian and *Modified Jacobian coordinates* are given in [22]. Here, we will only show the equations for the latter one since it is the one that we have used in our software implementation.

First, the *Jacobian coordinates* are represented as a quadruple (X, Y, Z, aZ^4) which is called *modified Jacobian coordinates.* Let $P = (X_1, Y_1, Z_1, aZ_1^4)$, $Q = (X_2, Y_2, Z_2, aZ_2^4)$ and $K = P + Q = (X_3, Y_3, Z_3, aZ_3^4)$ be points on elliptic curve $E_J(F_p)$.

- *EC addition formulas when* $(P \neq \pm Q)$ **[22]**:

$$U_1 = X_1 Z_2^2 \tag{7.36}$$

$$S_1 = Y_1 Z_2^3 \tag{7.37}$$

$$U_2 = X_2 Z_1^2 \tag{7.38}$$

$$S_2 = Y_2 Z_1^3 \tag{7.39}$$

$$H = U_1 - U_2 \tag{7.40}$$

$$\tag{7.41}$$

Then, we write the *modified Jacobian coordinates* as follows:

$$X_3 = -H^3 - 2U_1 H^2 + r^2 \tag{7.42}$$

$$Y_3 = -S_1 H^3 + r(U_1 H^2 - X_3) \tag{7.43}$$

$$Z_3 = Z_1 Z_2 H \tag{7.44}$$

$$aZ_3^4 = aZ_3^4 \tag{7.45}$$

- *EC doubling formulas when* $(P = Q)$

$$S = 4X_1 Y_1^2 \tag{7.46}$$

$$U = 8Y_1^4 \tag{7.47}$$

$$M = 3X_1^2 + (aZ_1^4) \tag{7.48}$$

$$T = -2S + M^2 \tag{7.49}$$

Then, we write the *modified Jacobian coordinates* as follows:

$$X_3 = T \tag{7.50}$$

$$Y_3 = M(S - T) - U \tag{7.51}$$

$$Z_3 = 2Y_1Z_1 \tag{7.52}$$

$$aZ_3^4 = 2U(aZ_1^4) \tag{7.53}$$

7.3 Authentication based on ECC

The protocol described in this chapter depends on the security of the so-called *elliptic curve primitives*, e.g., key generation, signature generation, and signature verification. These operations utilize the arithmetic of *points* which are elements of the set of solutions of an elliptic curve equation defined over a finite field. The security of the protocol depends on the intractability of the elliptic curve analogue of the discrete logarithm problem which is a well known and extensively studied computationally hard problem.

The mathematical background of the elliptic curve cryptography can be found in [55, 44].

Please refer to elliptic curve digital signature algorithm (ECDSA) given in chapter 4 before start reading the next section.

The authentication protocol given in this chapter was originally intended for mobile phones. However, it is also suitable for all the handheld devices and smart cards. This makes the protocol a very strong security algorithm candidate to be deployed in the next generation cellular phones and smart cards. Although the 160-bit key length is considered secure enough for now and near future, the algorithms were implemented in a way that the key length can easily be increased to 176, 192, 208,...,256, etc. This scalability makes our implementation a unique one. It is possible to make a single chip that will support variable key lengths, $GF(p)$ and $GF(2^k)$ arithmetic on the curve, as well as different algorithms such as RSA.

The terminal and the user acquire the corresponding certificates as shown in chapter 4 in Figures 4.1 and 4.2. The certificates consist of a pair of integers which is denoted as (r_s, s_s) for the server and (r_u, s_u) for the user. Here r_u and r_s are the x coordinates of the (distinct) elliptic curve points R_u and R_s, respectively. As mentioned earlier, the proposed protocol is based on the ECDSA.

The protocol steps and its resistance to several attacks have been explained in chapters 4 and 5.

The number of exchanged messages of the protocol (given in chapter 4) over the air is four. It is important to keep the number of messages as small as possible since the propagation delay will make customers wait for some time until the call is setup. For low bit transmission channels the transmission time will be the dominant factor. On the other hand, for high rate transmission channels the bottleneck will be the encryption and decryption operations.

The protocol in chapter 4 consists of exchanging public keys, generating random challenge numbers, exchanging encrypted certificates and the other necessary data using the special key, and then verifying the certificates in order to complete mutual authentication process. The computational cost until this point on the user side is just a point multiplication on the curve (eP operation), generating random number, a secret key encryption and a secret key decryption (DES, 3DES, RC5, or IDEA), and finally an ECC signature verification operation. The timing figures of these operations will increase as we increase the ECC key length from $160-bit$ to $176-bit$, $192-bit$, etc. Having a scalable cryptographic architecture like this one will ensure the users be confident on long term investments. It is not uncommon to assume that the big credit/smart card or cellular phone providers would not put money on the products with just only one key length option.

The last part of the protocol is just a session key establishment between the user and the server. The one-time unique key is obtained by hashing several previously obtained data. This key will be used to encrypt the data sent through the channel.

7.4 32-bit ARM Software Development Toolkit

We have implemented a scalable elliptic curve cryptosystem on a 32-bit ARM software development product designed by ARM Inc. ARM offers several microprocessor cores among which ARM7TDMI, a 32-bit RISC processor, is our concern because of its applicable features to cellular phones and other handheld devices.

The ARM7TDMI is a microprocessor core designed by Advanced RISC Machines in Cambridge, England. It is optimized for the best combination of die size, performance and power consumption. The processor uses a three-stage pipeline: fetch, decode and execute [39].

A pure RISC processor executes each instruction in a single cycle. However, no nonsuperscalar commercial RISC processors actually achieve this goal. ARM7 takes one cycle to perform most data processing operations, which account for three cycles, and stores two cycles. Load and store multiples can take up to 18 cycles. Overall, ARM7 achieves an average CPI (clock cycles per instruction) of around 1.8 [69].

ARM has thirty-one, 32-bit registers. At any given time, sixteen are visible; the other registers are used to speed up exception processing. All the register specifiers in ARM instructions can address any of the 16 registers.

The ARM7TDMI is a very simple RISC processor. The core is fully 32-bit including 32-bit ALU, barrel shifter, data and address bus. Although the 4GB of address range is rarely used in wireless applications, it does have the advantage of simplifying the decode logic by using the upper address lines as chip selects. Some of the characteristics of ARM7TDMI processor are as follows [38]:

- *Shortest instruction execution time*:
  - 800ns (at f = 80MHz)
- *Registers*:
  - 30 general purpose registers
  - 6 status registers

  - A program counter

- *Instruction Set*: 48 instructions

  - load and store instructions
  - data processing instructions
  - multiply instructions
  - coprocessor instructions
  - branch instructions

Portable and handheld products require processors that consume less power than those in desktop and other powered applications. RISC processors such as ARM7TDMI have some extra strengths as far as the power concerned. A modern 32-bit RISC architecture can provide software compatibility between a range of products. This kind of modern microcontroller family is also very easy to implement. These microprocessors are available as small cores which makes them easy to integrate with. Another advantage is on-chip debug support. These advantages make this family a good fit for embedded applications such as cellular phones.

Another advantage of the ARM7TDMI is the fact that it has two instruction sets. The ARM7TDMI implements both the traditional 32-bit wide ARM instruction set and the new Thumb instruction set which is only 16-bit wide. The Thumb instruction set was added to remove the limitations of code density and performance from narrow memory. Effectively, the traditional 32-bit ARM instruction set was compressed into the Thumb 16-bit instruction set. Thumb instructions are then decompressed at execution time to produce a traditional 32-bit wide ARM instruction, which is then executed on the core as normal. As the ARM decode phase is relatively simple, it is possible to do the Thumb decompression on the fly without taking any additional cycles. The special use of ARM thumb instructions enables ARM to evaluate real GSM, DECT and D-AMPS code from the leading wireless players. There are three main components while benchmarking the code [34]:

- Code Density : Shows how much memory is required for given high level "C" code. The smaller size will result in reduced cost.

- Performance : The processors' clock speed is an important factor. The smaller the clock rate to execute given algorithms, the less the power consumed. This will also lead to easier designs. The 32-bit RISC controllers will spend most of its time in an idle mode resulting in saving power.

- Power Consumption : It is one of the most important factors in wireless technology. The lower power consumption will make the batteries life longer, the size smaller and the price cheaper. ARM7TDMI consumes only 1.85mW per MHz while StrongARM runs up to 233MHz but only consumes 900mW [34].

ARM7TDMI is widely accepted and proposed to be used in cellular phone and smart phone technology due to its cost and power efficiencies. The future prospects show that ARM9TDMI will probably replace ARM7TDMI. Integrating the DSP module with ARM7 family will produce the new ARM9 family [33].

7.5 Parameter Lengths

Table 7.1 shows the protocol parameter lengths for variable ECC key lengths. Table 7.2 shows the corresponding bandwidth and storage requirements for variable elliptic curve digital signature key lengths.

7.6 ECC Library

7.6.1 The Software Architecture

This practical cryptographic library implementation of ECC over prime characteristic was designed to perform ECDSA signature generation and signature verification which is being standardized in the ANSI X9F1 and IEEE P1363 standards committees. The IEEE-P1363 describes the algorithms in detail for elliptic point addition, doubling, multiplication, etc.

Table 7.1. The Protocol Parameter Lengths for Various ECC Key Lengths in Bits.

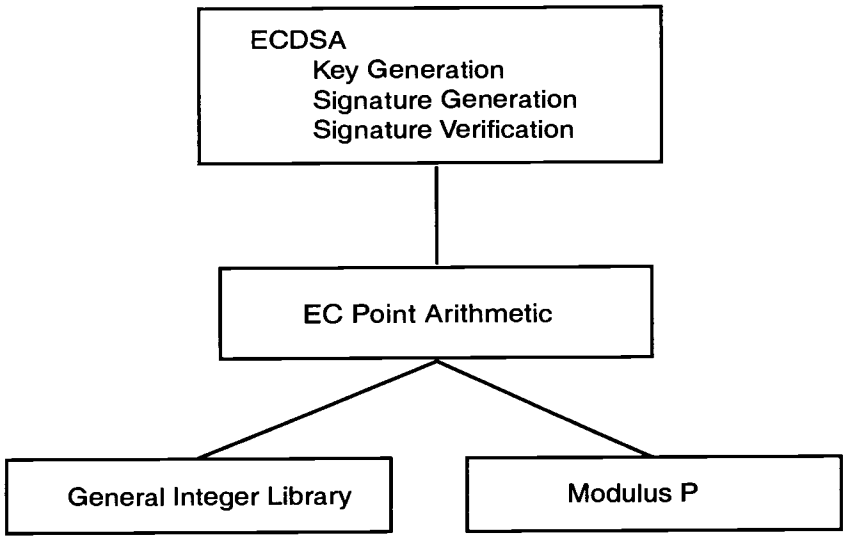
Parameters	160-bit ecc	176-bit ecc	192-bit ecc	208-bit ecc	256-bit ecc
Q_u, Q_s	161	177	193	209	257
e_u, e_s	160	160	160	160	160
$(r_{u,s}, s_{u,s})$	320	352	384	416	512
$t_{u,s}, g_{u,s}$	64	64	64	64	64

Table 7.2. The Protocol Bandwidth and Storage Requirements in Bits.

Parameters	160-bit ecc	176-bit ecc	192-bit ecc	208-bit ecc	256-bit ecc
Bandwidth	1730	1826	1922	2018	2306
Storage	1408	1520	1632	1744	2080

In creation of our library, we did not make any assumption on the elliptic curve parameters to be used. Elliptic curves can be generated randomly. Note that some ECDSA implementations fix the constant term **a** of the curve equation to $p - 3$ to speed up the elliptic doubling. In our case, the curve parameters and the base point (P_x, P_y) are generated randomly. Our library allows users to choose different curves with different key lengths, therefore our library is scalable. The machine word size is 32-bit on ARM microprocessor. The library is implemented in 27Kbyte code size and the *Modified Jacobian Coordinates* are used to represent the points on the curves since it gives the fastest point doubling timings. The block diagram of our library is shown in Figure 7.1. Short definitions of the modules are given as follows.

Figure 7.1. Software Architecture.



- *Modulo p Integer Library*

  This module contains modular operations such as modular addition, subtraction, multiplication and inversion operations modulo p, where p is the characteristic of the base field over which the elliptic curve is defined. In overall ECDSA signature generation operation, these routines are the most time consuming algorithms. Especially point multiplication operation dominates the timing performance of an EC signature. To improve the performance we use the improved Montgomery multiplication algorithm in our library.

- *General Integer Library*

  This library contains general operation routines. These routines accept variable length inputs.

- *EC Point Arithmetic Library*

  This library consists of point addition, point doubling, point multiplication and point inversion routines. Point addition and doubling routines perform the corresponding elliptic curve operations in modified *Jacobian* coordinates.

- ECDSA Key and Signature Generation/Verification

  This is the root module of our software architecture. Elliptic curve parameters and key generation are performed here. Upon creating these parameters, this top module can interact with other modules to generate signatures or to verify signatures. Note that our library does not contain a digest algorithm such as SHA-1 or MD5. We use randomly generated 160-bit message values, which is assumed to be the output of a hash function algorithm, to test the modules.

7.6.2 Possible Improvements

A list of possible enhancements for further speeding up and/or reducing code size can be given as follows.

- The scalar multiplication of the base point can be performed in more efficient way by having a precomputed look-up table in ROM area.
- The finite field multiplication operations dominate the performance of signature generation and verification. Even small improvements on the existing multiplication routines may improve overall ECDSA performance significantly.
- The use of trinomial and pentonomial algorithms will make the reduction operation faster and the complexity of the software programming smaller.
- The 16-bit wide *Thumb* instruction set of ARM7TDMI can be used to reduce the code size.

7.7 Implementation Results

In this section, we present our implementation timing results. The elliptic curve signature generation and verification timings are listed for variable key lengths just to give an idea about how fast these operations could be done in today's technology if needed. As we mentioned before the online mutual authentication protocol given in chapter 4 requires users to perform a point multiplication, a random number

Table 7.3. The Timing Results of the Cryptographic Operations

Parameters	160-bit ecc	176-bit ecc	192-bit ecc	208-bit ecc	256-bit ecc
point_mul	44.8	63.4	69.2	93.6	150.2
RNG	N/A	N/A	N/A	N/A	N/A
DES	0.25	0.25	0.25	0.25	0.25
SHA	2	2	2	2	2

generation, two secret key operations (DES) with $672-bit$ data for $160-bit$ ECC (one encryption + one decryption), one elliptic curve signature verification and finally one hash operation with $288-bit$ data. Table 7.3 shows the timings for these operations for variable ECC key lengths. Table 7.4 is also provided to see the ECDSA signature generation and verification timings clearly. The total protocol execution timings are given for the protocols given in chapter 4 and 5. Pro_1 refers to the protocol given in chapter 4 and Pro_2 refers to the protocol given in chapter 5.

Note that our library does not have a random number generator (RNG). Generating a random number is very fast therefore its timing value is negligible compared to the other operations such as point multiplication and signature generation. Similarly SHA operations can be executed very fast. Hasegawa, et al, reported $2msec$ for processing one-block of information (512-bits) with SHA-1 in their recently published paper [36]. It is a hardware implementation on a 16-bit Mitsubishi microprocessor (M16C). In our protocol the input size to the SHA-1 is given as $k + 128$ where k is being the implemented elliptic curve key length. The largest k value shown in the table is 256-bits for which the input size for SHA-1 is 384-bits. Therefore, for each key length given in the table 7.3, the SHA-1 input length in our protocol should be padded to reach 512-bit block size. We assume that in the worst case scenario we will obtain $2msec$ timing value for processing 1-block SHA-1 if implemented in our library.

Table 7.4. Signature Generation/Verification Timings in Milliseconds over $GF(p)$.

Operations	160-bit ecc	176-bit ecc	192-bit ecc	208-bit ecc	256-bit ecc
sign_gen	46.4	65.4	71.3	96.2	153.5
sig_ver	92.4	131.3	148.3	194.3	313.4
pro1_exe	139.7	197.2	220	290.4	466.1
pro2_exe	231.7	328.5	368.4	485.3	780.8

Generating random numbers are very fast so that it can be ignored when compared to other latency figures for point multiplication, signature generation and verification.

7.8 Conclusions

In this chapter, we presented a practical implementation of ECC over prime characteristic. The field and elliptic curve operation algorithms in the library were written in a way that the implemented design will permit the use of increased key lengths. In his recently published paper [47], Lenstra indicates that 1024-bit RSA and 139-bit EC offer computationally equivalent security. This claim is better than the generally believed security comparison in which it was said that 1024-bit RSA and 160-bit EC offer similar security performance. We should note that the latter statement is believed to be true for cost equivalent security. The smaller key size results in smaller system parameters, smaller certificates, bandwidth savings, faster implementations, lower power requirements, and smaller hardware processors [55].

In our implementation, we created an EC library, which is capable of performing EC signature generation and verification operations. More importantly, the implementation permits users to select different elliptic curves with longer key sizes. This scalable architecture of the design enables ECC service providers such as govern-

ments, banks, telecommunication giants and all other digital service companies to offer long term security on their user applications such as mobile phones, smart cards, etc.

With this implementation it is possible to obtain timing results less than $100msec$ for both EC-160 signature generation and verification on a 32-bit ARM processor. To be specific, 160-bit EC signature generation and verification timings were obtained as $46.4msec$ and $92.4msec$, respectively. In addition, the timing performances were obtained for recently proposed wireless authentication and key agreement protocols [10] [7]. These protocols can be used in third generation wireless communication as security protocols due to their bandwidth and storage efficiency and fast execution timing performances.

Chapter 8
IMPLEMENTATION OF THE PROTOCOLS OVER $GF(2^k)$

" Personally, I don't think there's intelligent life on other planets. Why should other planets be any different from this one? "

Bob Monkhouse

8.1 Introduction

In this chapter we present the real-time execution timings of the protocols given in chapters 4 and 5. The timings were obtained on ECC over $GF(2^k)$ using once again ARM software development toolkit. ARM7TDMI was chosen as the microprocessor core among several other ARM processors. As mentioned before software implementations over $GF(p)$ are faster than those over $GF(2^k)$. On the other hand, hardware implementations over $GF(2^k)$ produce better results than those over $GF(p)$ due to the carry-free nature of $GF(2^k)$ arithmetic.

In chapter 7, we showed $GF(p)$ arithmetic operations for efficient implementation, and then we presented ECDSA signature generation, verification and protocol execution timings over this field with prime characteristic. In this chapter we first provide some background information on $GF(2^k)$ arithmetic operations and elliptic curves over this type of fields. Then, we present our implementation timing results for the protocols proposed in this thesis.

ARM7TDMI was selected among several others as the core processor for the implementation. It operates at 80MHz. This processor is commonly used in wireless applications, automotive technologies, etc. It is a low power processor with many useful features.

In this chapter, we give some background on arithmetic operations in $GF(2^m)$. The selection of the underlying finite field and the elliptic curve groups over $GF(2^m)$

are given. We show the arithmetic equations for the operations such as field addition, field multiplication, field substraction, and so on. Then several methods for representing elliptic curve elements over $GF(2^m)$ are given. An example of a small field is also provided.

At the end of the chapter, the implementation issues for both type of fields and our implementation results are provided.

8.2 Background

8.2.1 Choice of Underlying Field GF(q) and Elliptic Curve E

When setting up an elliptic curve cryptosystem, there are three basic decisions that need to be made [21]:

- Selection of the underlying finite field $GF(q)$.
- Selection of the representation for the elements of $GF(q)$.
- Selection of the elliptic curve **E** over $GF(q)$.

In the following, we discuss how the choices of the underlying field, its representation, and elliptic curve, may effect the intractability of the ECDLP and therefore the security of the elliptic curve cryptosystem.

Generally in practical implementations, the two most common choices for the underlying finite field are $GF(p)$ and $GF(2^m)$ where p is an odd prime. The ECDLP is equally difficult for instances which use both choices under the assumption that the sizes 2^m and p of the fields are approximately equal. It is noted that there have not been any mathematical claims to date which suggest that the ECDLP for elliptic curves over $GF(2^m)$ may be any easier or harder than the ECDLP for the curves over $GF(p)$.

There are many ways in which the elements of $GF(2^m)$ can be represented if this field is selected as the underlying finite field. The two most efficient ways are *optimal normal basis representation* and a *polynomial basis representation.* It is

possible to convert the elements from one basis to another using an appropriate change-of-basis matrix, the intractability of the ECDLP is not affected by the choice of the representation.

There are some algorithms that yield algorithms for the ECDLP when the elliptic curve is supersingular. It is easy to avoid this type of curves and in fact most of the curves are non-supersingular.

8.2.2 Elliptic Curve Groups over $GF(2^m)$

Elements of the field $GF(2^m)$ are m-bit strings. The rules for arithmetic in $GF(2^m)$ can be defined by either *polynomial representation* or by *optimal normal basis representation.* Since $GF(2^m)$ operates on bit strings, computers can perform arithmetic in this field very efficiently.

An elliptic curve with the underlying field $GF(2^m)$ is formed by choosing the elements $\mathbf{a}$ and $\mathbf{b}$ within $GF(2^m)$ with the condition that $\mathbf{b}$ is not zero. As a result of the field $GF(2^m)$ having characteristic 2, the elliptic curve equation is slightly adjusted for binary representation:

$$\mathbf{y^2 + xy} \;=\; \mathbf{x^3 + ax^2 + b}. \tag{8.1}$$

The elliptic curve includes all points (x, y) which satisfy the elliptic curve equation over $GF(2^m)$ (where $\mathbf{x}$ and $\mathbf{y}$ are elements of $GF(2^m)$). An elliptic curve group over $GF(2^m)$ consists of the points on the corresponding elliptic curve, together with a point at infinity, $\mathbf{0}$. There are finitely many points on such an elliptic curve.

8.2.3 Arithmetic in an Elliptic Curve Group over $GF(2^m)$

Elliptic curve groups over $GF(2^m)$ have a finite number of points, and their arithmetic involves no round off error. This combined with the binary nature of the field, $GF(2^m)$ arithmetic can be performed very efficiently by a computer. The following sections show the arithmetic operations over $GF(2^m)$.

8.2.3.1 *Adding Distinct Points P and Q*

The negative of the point $P = (x_P, y_P)$ is the point $-P = (x_P, x_P + y_P)$. If P and Q are distinct points such that P is not $-Q$, then $P + Q = R$ where

$$s = (y_P - y_Q)/(x_P + x_Q) \tag{8.2}$$

$$x_R = s^2 + s + x_P + x_Q + a \tag{8.3}$$

$$y_R = s(x_P + x_R) + x_R + y_P \tag{8.4}$$

The addition of P and $(-P)$ will give the point at infinity, that is:

$$\mathbf{P} + (-\mathbf{P}) = \mathbf{O} \tag{8.5}$$

In addition, $P + O = P$ for all points $\mathbf{P}$ in the elliptic curve group.

8.2.3.2 *Doubling the Point P*

If $x_P = 0$, then $2P = O$, provided that x_P is not 0.

The double of the point P is given as R:

$$\mathbf{2P} = \mathbf{R} \tag{8.6}$$

$$s = x_P + y_P/x_P \tag{8.7}$$

$$x_R = s^2 + s + a \tag{8.8}$$

$$y_R = x_P + (s + 1) * x_R \tag{8.9}$$

Note that **a** is one of the parameters chosen with the elliptic curve and **s** is the slope of the line through **P** and **Q**.

8.2.4 Polynomial Representation

The elements of $GF(2^m)$ are polynomials of degree less than **m**, with coefficients in $GF(2)$; that is,

$$a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_2x^2 + a_1x + a_0 \tag{8.10}$$

where a_i is either 0 or 1. These elements can be written in vector form as $(a_{m-1} \ldots a_1 a_0)$. $GF(2^m)$ has 2^m elements.

The main operations in $GF(2^m)$ are addition and multiplication. Some computations involve a polynomial

$$f(x) = x^m + f_{m-1}x^{m-1} + f_{m-2}x^{m-2} + \ldots + f_2 x^2 + f_1 x + f_0 \tag{8.11}$$

where each f_i is in $GF(2)$. The polynomial $f(x)$ must be irreducible; that is, it cannot be factored into two polynomials over $GF(2)$, each of degree less than $\mathbf{m}$.

8.2.4.1 *Addition*

The addition equation is given as:

$$(a_{m-1} \ldots a_1 a_0) + (b_{m-1} \ldots b_1 b_0) = (c_{m-1} \ldots c_1 c_0) \tag{8.12}$$

where each $c_i = a_i + b_i$ is over $GF(2)$. Addition is just the componentwise XOR of

$$(a_{m-1} \ldots a_1 a_0), (b_{m-1} \ldots b_1 b_0). \tag{8.13}$$

8.2.4.2 *Subtraction*

In the field $GF(2^m)$, each element $(a_{m-1} \ldots a_1 a_0)$ is its own additive inverse, since

$$(\mathbf{a_{m-1} \ldots a_1 a_0}) + (\mathbf{a_{m-1} \ldots a_1 a_0}) = (\mathbf{0 \ldots 00}), \tag{8.14}$$

is the additive identity. Thus addition and subtraction are equivalent operations in $GF(2^m)$.

8.2.4.3 *Multiplication*

The multiplication equation is given as:

$$(a_{m-1} \ldots a_1 a_0)(b_{m-1} \ldots b_1 b_0) = (r_{m-1} \ldots r_1 r_0) \tag{8.15}$$

where $r_{m-1}x^{m-1} + r_{m-2}x^{m-2} + \ldots + r_2x^2 + r_1x + r_0$ is the remainder when the polynomial

$$(a_{m-1}x^{m-1} + \ldots + a_2x^2 + a_1x + a_0)(b_{m-1}x^{m-1} + \ldots + b_2x^2 + b_1x + b_0) \quad (8.16)$$

is divided by the polynomial $f(x)$ over $GF(2)$. All the polynomial coefficients are reduced modulo 2.

8.2.4.4 *Exponentiation*

The exponentiation $(a_{m-1}x^{m-1} + \ldots + a_2x^2 + a_1x + a_0)^e$ is performed by multiplying together **e** copies of $(a_{m-1} \ldots a_1a_0)$

8.2.5 An Example of an Elliptic Curve Group over $GF(2^m)$

As a very small example, consider the field $GF(2^4)$, defined by using polynomial representation with the irreducible polynomial

$$f(x) = x^4 + x + 1. \quad (8.17)$$

The element $\mathbf{g} = \mathbf{(0010)}$ is a generator for the field. The powers of **g** are given in Table 8.1.

In a true cryptographic application, the parameter **m** must be large enough to preclude the efficient generation of such a table otherwise the cryptosystem can be broken. In today's practice, $\mathbf{m = 160}$ is a suitable choice. The table allows the use of generator notation (g^e) rather than bit string notation, as used in the following example. Also, using generator notation allows multiplication without reference to the irreducible polynomial

$$f(x) = x^4 + x + 1.$$

Consider the elliptic curve

$$\mathbf{y^2 + xy = x^3 + ax^2 + b}. \quad (8.18)$$

Table 8.1. The Representation of the Elements in $GF(2^4)$.

Element in $GF(2^4)$	Polynomial	Coordinates
0	0	(0000)
g	g	(0010)
g^2	g^2	(0100)
g^3	g^3	(1000)
g^4	$g+1$	(0011)
g^5	g^2+g	(0110)
g^6	g^3+g^2	(1100)
g^7	g^3+g+1	(1011)
g^8	g^2+1	(0101)
g^9	g^3+g	(1010)
g^{10}	g^2+g+1	(0111)
g^{11}	g^3+g^2+g	(1110)
g^{12}	g^3+g^2+g+1	(1111)
g^{13}	g^3+g^2+1	(1101)
g^{14}	g^3+1	(1001)
g^{15}	1	(0001)

Here $a = g^4$ and $b = g^0 = 1$. The point (g^5, g^3) satisfies the equation over $GF(2^m)$:

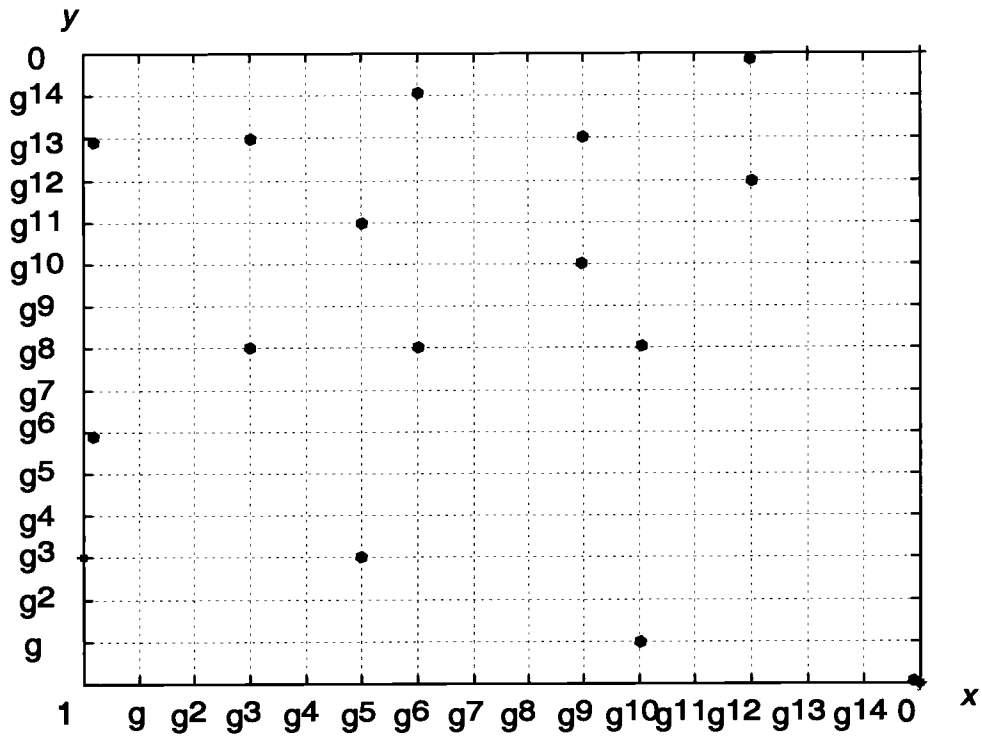
$$\begin{aligned} y^2 + xy &= x^3 + g^4x^2 + 1 \\ (g^3)^2 + g^5g^3 &= (g^5)^3 + g^4g^{10} + 1 \\ g^6 + g^8 &= g^{15} + g^{14} + 1 \\ (1100) + (0101) &= (0001) + (1001) + (0001) \\ (1001) &= (1001) \end{aligned} \tag{8.19}$$

The fifteen points which satisfy this equation are:

$(1, g^{13})$, (g^3, g^{13}) , (g^5, g^{11}) , (g^6, g^{14}) , (g^9, g^{13}) , (g^{10}, g^8) (g^{12}, g^{12}) , $(1, g^6)$, (g^3, g^8) , (g^5, g^3) , (g^6, g^8) , (g^9, g^{10}) (g^{10}, g) , $(g^{12}, 0)$, $(0, 1)$

All points which satisfy equation (8.18) are shown in Figure 8.1.

Figure 8.1. Elements in the Elliptic Curve of Equation (8.18)



8.3 Implementation Issues

Since the elliptic curve discrete logarithm problem appears to be harder than the discrete logarithm problem in Z_p, that is, the problem of factoring a composite integer n, one can use an elliptic curve group that is significantly smaller than Z_p. As it was explained before, an elliptic curve $E(GF(p))$ with a point $P \in E(GF(p))$ whose order is a 160-bit prime offers approximately the same level of security as DSA with a 1024-bit modulus p and RSA with a 1024-bit modulus n [21].

Computational efficiency of EC systems can be shown in the following [21]:

Assume that we compare the timings to compute:

- The scalar multiplication kP where $P \in E(GF(p))$, E is an elliptic curve, $p \sim 2^{160}$, and k is a 160-bit integer (this is an operation in ECDSA) and,

- $g^k mod p$, where p is a 1024-bit prime and k is a 160-bit integer (this is a DSA operation).

Let us assume that a field multiplication in $GF(p)$, where $log_2 p = l$, takes l^2 bit operations; then a modular multiplication in DSA takes

$$(1024/160)^2 \sim 41 \tag{8.20}$$

times longer than a field multiplication in ECDSA. On the average it takes 160 elliptic curve doublings and 80 elliptic curve additions to compute kP by repeated doublings and additions. From the addition formula given before, we see that an elliptic curve addition or doubling requires 1 field inversion and 2 field multiplications. Note that the cost of field addition is negligible, as is the cost of a field squaring if the field $GF(2^m)$ is used instead of $GF(p)$. We can also assume that the time to perform a field inversion is roughly equivalent to that of 3 field multiplications (this is what has been reported in practice for the case of $GF(2^m)$ [21]). Then it can be seen that computing kP requires the equivalent of 1200 field multiplications, or $1200/41 \sim 29$ 1024-bit modular multiplications. On the other hand, computing $g^k \ mod p$ by repeated squaring and multiplying requires, on average, 240 1024-bit modular multiplications. Thus, the operation in ECDSA can be expected to be 8 times faster than the operation in DSA.

Using smaller groups in elliptic curve cryptosystems makes it feasible low-cost implementations in restricted computing environments, such as wireless devices and smart cards. Although no security or standardization differences exist between the two types of underlying finite fields ($GF(p)$ and $GF(2^m)$), performance and cost differences can arise when implementing a smart card application.

In general software environments in which an arithmetic processor is already available for modular exponentiation, the performance of $GF(p)$ can be improved so that in some cases it exceeds the performance of $GF(2^m)$. This holds true for platforms such as those using Pentium processor or, in the case of smart cards, those having a crypto coprocessor to accelerate modular arithmetic [21].

Table 8.2. DES, SHA and eP Timings in Milliseconds over $GF(2^k)$.

Parameters	163-bit ecc	179-bit ecc	191-bit ecc	211-bit ecc	251-bit ecc
point_mul	71.7	101.4	110.7	149.8	240.4
DES	0.25	0.25	0.25	0.25	0.25
SHA	2	2	2	2	2

Point compression allows the points on the elliptic curve to be represented with fewer bits of data. In *smart card implementations* point compression is essential because it reduces not only the storage space for keys and certificates on the card, but also the amount of data that needs to be transmitted to and from the card.

$GF(2^m)$ hardware implementations offer significant performance and die size advantages over $GF(p)$ hardware implementations. In our case, we have obtained the ECDSA and protocol execution timings on Pentium processor and therefore the timing performance of our $GF(2^m)$ implementation is slower compared to the timings taken over $GF(p)$ given in previous chapter.

8.4 Implementation Results

In this section, we present our implementation timing results. The elliptic curve signature generation and verification timings are listed for variable key lengths just to give an idea about how fast these operations could be done in today's technology if needed.

Table 8.2 shows the timing results for point multiplication in $GF(2^k)$, secret key encryption for a block of data (512-bit) and hash operation for different curves.

Table 8.3 shows the signature generation and verification and protocol execution timings for the protocols given in chapters 4 and 5.

Table 8.3. The Protocol Timings in Milliseconds over $GF(2^k)$.

Operations	163-bit ecc	179-bit ecc	191-bit ecc	211-bit ecc	251-bit ecc
sign_gen	74.3	104.6	114.0	154.0	245.6
sig_ver	147.8	210.0	237.3	310.8	501.4
pro1_exe	222.0	313.9	350.5	463.1	744.3
pro2_exe	370.4	525.0	589.1	776.2	1248.9

Generating random numbers are very fast so that it can be ignored when compared to other latency figures for point multiplication, signature generation and verification.

8.5 Conclusions

In this chapter, we presented a practical implementation of ECC over a binary field. The field and elliptic curve operation algorithms in the library were written in a way that the implemented design will permit the use of different curves with variable key lengths.

In our implementation, we created an EC library, which is capable of performing EC signature generation and verification operations. More importantly, the implementation permits users to select different elliptic curves with longer key sizes.

As it can be seen from the results that the timing performance of the protocols is much slower for the curves defined over $GF(2^k)$ compared to the ones defined over $GF(p)$. However, we should note that with the use of precomputed tables, it is possible to improve these timing results in the order of 5-8 in magnitude. Unfortunately, table look-up methods bring excessive memory requirements that are not desirable in mobile environments.

Our conclusion is that using elliptic curves with prime characteristics over $GF(p)$ is much faster in software and the protocols using this type of fields can efficiently be implemented in software. However, the use of dedicated hardware implementations can even be faster than the software implementations.

Chapter 9
CONCLUSIONS

Public-key cryptographic systems have proven to be effective and more manageable than private-key systems in many areas. Security protocol designers need to make a choice between three types of public-key systems. These are integer factorization systems, discrete logarithm systems and elliptic curve cryptographic systems. Each of these systems provide the main security services: Confidentiality, authentication, data integrity and nonrepudiation. Although several well-known public-key cryptosystems have been implemented based on one of these hard mathematical problems, none of them have been proven to be intractable. Rather they are believed to be intractable since many years of intensive research to solve these problems by leading mathematicians and computer scientists have not produced efficient algorithms. These intensive work, however, has resulted in a widely accepted view that EC discrete logarithm problem is significantly more difficult than the other two problems. The additional strength-per-bit property of ECC provides significant efficiency savings in many applications in which computational power, bandwidth, and storage space are limited.

This chapter summarizes the results of the research, lists the most significant contributions and finally discusses future research directions in this highly popular area.

9.1 Discussion of Results

Chapter 4 is based on the paper [10]. In this chapter, we proposed an ECC based mutual authentication and key agreement protocol for wireless communication. The protocol efficiently uses the elliptic curve digital signature algorithm to generate and

verify the certificates issued by the universal certifying authority. *EC Diffie-Hellman key exchange* protocol is also used in the protocol for generating cryptographic keys. These keys are used to protect the data exchanged between terminals and users for authentication purposes. The proposed protocol generates unique session key to encipher voice conversation, e-mail, etc.

Considering the availability of digital scanners that can be used to capture communicated data over the air and the recent breakdown of GSM security algorithms [67, 31], it is incredibly important to improve today's cellular and digital mobile security. Quite importantly, the proposed protocol in this research is not limited to mobile communications. E-Commerce, smart card technology, wireless LAN's and all the other handheld devices are perfectly fit to this new power, bandwidth and space-efficient security protocol. We strongly believe that our protocol with a few modifications is a strong security protocol candidate for the upcoming third generation wireless communication technologies as well as the existing second-generation technologies. The physical requirements of the protocol as well as its comparison to the other well-known public-key authentication protocols are given in chapter 4.

Chapter 5 is based on the paper [7]. This chapter presents an end-to-end security protocol for mobile networks. The protocol proposed in this chapter is an improved version of the previous protocol proposed in chapter 4. This improved protocol contains an on-line signature operation being done by the users to address the non-repudiation issue more clearly. In addition, the public keys of the mobile devices requesting a service from the service providers are never revealed in this protocol providing the total user identity confidentiality over the wireless interface.

Unlike the former protocol given in chapter 4, this protocol provides more interoperability by presenting an on-line negotiation on the secret key algorithm so that the use of the foreign and domestic versions of the mobile product would be possible. In addition, the use of a mobile device within a different network environment using a different secret key algorithm could be done with this approach.

Chapter 5 also explains the process of crossing domain boundaries during a call session, visiting other networks and operating with multiple certification authorities.

The authorization issue as well as the protocol's achievement on the proposed goals and a complete protocol security analysis are also given in this chapter. Although the improved protocol involves a little more computational complexity compared the earlier version, it is still superior (in bandwidth and storage requirements) to the other public key based security protocols summarized in chapter 3. Like the previous protocol proposed in chapter 4, this new protocol was also compared to the couple of well known security protocols and the comparison results were given in this chapter.

Chapter 4 and chapter 5 have presented new security protocols for mobile networks. In these chapters it has been shown that with the use of ECC based techniques, it is now possible to provide the desired security features for mobile networks with smaller bandwidth and storage requirements. The rest of the thesis concentrates on the implementation issues of these protocols in order to show that fast and efficient realization of the proposed protocols is possible.

Chapter 6 is based on the paper [9]. In this chapter, we deal with the software implementation details of the arithmetic operations that are used in ECC security protocols. Examples of these operations are field operations: Field multiplication, field squaring, field inversion , and EC arithmetic operations: EC point addition, point doubling and point multiplication.

The authentication protocols proposed in chapters 4 and 5 make use of these arithmetic operations. Therefore, the execution timings of the protocols are strongly based on fast and efficient implementation of these operations in both software and hardware environments. Elliptic curves over $GF(2^m)$ were used in the implementation because of their efficient implementations due to their carry-free nature. An area of particular interest is the development of efficient software implementations of the arithmetic and elliptic curve operations in $GF(2^k)$, where k is a composite number as $k = nm$. By using a logarithmic table lookup method for the ground field $GF(2^n)$, we use an optimal normal basis in $GF(2^m)$ to represent the elements of $GF(2^{nm})$. The resulting field operations, multiplication and squaring are quite efficient, and they do not involve modular reductions, which are costly operations. The timing results of these operations are given in the table in chapter 6. Our timing

results are faster than the results obtained by a similar implementation by De Win, et al [71].

Chapter 7 is based on the paper [11]. In this chapter, we give implementation timing results for ECC over a field of prime characteristic on a 32-bit ARM microprocessor. ARM7TDMI is selected as the ARM microprocessor core due to its simplicity. It is a low-power processor, therefore it is suitable for portable and handheld devices. Small cores make these type of microprocessors easy to integrate with. On-chip debug support and other hardware characteristics of this family make them a good fit for embedded applications such as mobile phones. ARM7TDMI family is widely accepted and proposed to be used in cellular phone and smart phone technology due to it's cost and power efficiencies.

In this part of the research we created a software library, which is capable of performing elliptic curve signature generation and verification. The arithmetic operation and elliptic curve operation algorithms were written with modified or improved versions of existing software codes. *Montgomery's multiplication method* was used in field multiplication to speed-up the process. *Modified Jacobian coordinates* were used to represent the points on the curve since they provide the fastest *doublings*. The library offers scalability meaning that ECC with variable key sizes can be supported. For investors looking for long-term security, this is a very important factor.

The timing results given in Chapter 7 shows that it is now possible to obtain signature generation and verification timings for less than $100msec$ on a 32-bit processor for 160-bit ECC. A recent implementation by Hasegawa, et al. [36] achieves the timings of $150msec$ and $630msec$ for signature generation and verification, respectively. They obtained the timings on a 16-bit Mitsubishi processor using special hardware and curve parameters. Our timings are $46.4msec$ and 92.4, respectively. The on-line execution timings of our new authentication and key agreement protocol were also obtained and provided in Chapter 7.

Chapter 8 is based on the paper [8]. This chapter basically follows the same approach taken in chapter 7. The implementation timing results for the protocols given in chapters 4 and 5 are obtained once again on ARM7TDMI microprocessor.

The underlying finite fields are chosen in the form of $GF(2^k)$ over which different elliptic curves with different key lengths are selected. As we know ECC arithmetic defined over $GF(2^k)$ will be slower in software implementation compared to the ones over $GF(p)$. Therefore, the signature generation and verification timings along with the protocol execution timings are slower in this type of fields. An introduction to the arithmetic operations on $GF(2^k)$ and the timing results of the protocols are given in this chapter.

9.2 Summary of Contributions

Below is the summary of the contributions made:

- Public key cryptographic systems RSA, DSA and ECC were fully analyzed and compared.
- Several public and private key authentication and key agreement protocols were discussed. These include the wireless protocols by Beller-Chang-Yacobi [15], by Aziz-Diffie [12], by Park [61], by Molva Samfat and Tsudik [62], by Mu and Varadharajan [59], and by Beller and Yacobi [16]. The flaws in these protocols were presented.
- A new mutual authentication and key agreement protocol based on elliptic curve cryptosystem was proposed. This protocol provides better performance in terms of bandwidth and storage compared to the others presented. The simple structure of the protocol makes it easy to implement it in wireless environments.
- An end-to-end mobile security protocol was presented. This is an improved version of the previous one, and provides better security features. The protocol targets not only cellular phones and smart cards but also the up-to-date wireless products such as smart phones using XML based wireless services, personal data assistants (PDAs) as well as secure web connections over the wired networks. The protocol permits users to negotiate on the secret key algorithm

for interoperability and supports SPKI certificates to provide authorization for the users to perform legal operations.

- The following arithmetic and elliptic curve operations were implemented in $GF(2^{176})$.
  - Field operations: Addition, multiplication and inversion.
  - Elliptic curve operations: Doubling, addition and multiplication.
    The implementation has provided better performances compared to a similar work [71].
- The proposed protocols were simulated on a ARM microprocessor using ARM software development toolkit. The protocol execution timings as well as elliptic curve signature generation and verification (ECDSA, ECDSAV) timings were obtained and provided in chapter 7 for the case of $GF(p)$ and in chapter 8 for the case of $GF(2^k)$, respectively. In this implementation, scalability, a very important requirement in long-term security, is supported. Our timing results were compared to a recently published work by Hasegawa, et al [36].
- Other application areas of ECC are being explored. New security protocols based on ECC for authenticating digital cameras and MPEG movies are being considered to be implemented.

9.3 Future Work

The work in this thesis implies that more effort must be spent in designing ECC based universal security protocols for wireless communication and smart card technology. We should note that private key systems offer faster encryption and decryption, however they lack supporting digital signatures, which are used to offer authentication, data integrity and nonrepudiation. Therefore, both systems should be combined in a way that public key system should be used to provide key exchange and authentication algorithms while private key systems could be used to keep the communicated

data private with encryption process. The existing public key security protocols for mobile networks were rejected in the standard bodies simply because they involve costly public key operations for constrained environments. However none of these public key based protocols are based on ECC. In this thesis, we show that the same level of security with smaller key lenghts is possible and it can be implemented very efficiently. Therefore more effort should be put in ECC based techniques to design even better protocols.

In addition, the current protocols should be analyzed with high technology to check the security of the cryptographic systems on which they are based. As mentioned before, none of the problems (integer factorization, discrete logarithm and EC discrete logarithm) have been proven to be intractable. One of the natural results of this fact is the necessity of having scalable architecture in which it should be possible to increase the key size if needed without changing the hardware.

Furthermore, to speed-up the arithmetic and EC operations even more, current algorithms to perform modular addition, multiplication and reduction should be improved. It is possible to obtain faster timings for these operations with special dedicated hardware and software implementations. We plan to concentrate more in this area in the future.

BIBLIOGRAPHY

[1] ETSI/TC Recommendation GSM 03.20. Security related network function. Version 3.3.2, January 1991.

[2] ETSI. ETS 300 175-7. October 1992.

[3] C. Nicolacacos A. Al-Adnani and W. Adi. USECA:UMTS Security Architecture. Panasonic technical report, June 1999.

[4] C. Adams and S. Farrell. Internet X.509 public key infrastructure certificate management protocols. RFC 2510, March 1999.

[5] G. B. Agnew, R. C. Mullin, and S. A. Vanstone. An implementation of elliptic curve cryptosystems over $F_{2^{155}}$. *IEEE Journal on Selected Areas in Communications*, 11(5):804–813, June 1993.

[6] Smartcard Developer Association. `http://www.scard.org`.

[7] M. Aydos and Ç. K.Koç. Designing security protocols for future mobile networks. Submitted for publication, April 2000.

[8] M. Aydos, F. Rodríguez-Henríquez, and Ç. K. Koç. A practical implementation of an ecc based mobile security protocol on an arm microprocessor over $GF(2^n)$. In progress, May 2000.

[9] M. Aydos, E. Savaş, and Ç. K. Koç. Implementing network security protocols based on elliptic curve cryptography. In S. Oktuğ, B. Örencik, and E. Harmancı, editors, *Proceedings of the Fourth Symposium on Computer Networks*, pages 130–139, Istanbul, Turkey, May 20–21 1999.

[10] M. Aydos, B. Sunar, and Ç. K. Koç. An elliptic curve cryptography based authentication and key agreement protocol for wireless communication. In *2nd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications Symposium on Information Theory*, Dallas, Texas, October 30 1998.

[11] M. Aydos, T. Yanık, and Ç. K.Koç. High-speed implementation of an ecc-based wireless authentication protocol on a 32-bit microprocessor. Submitted for publication, January 2000.

[12] A. Aziz and W. Diffie. A secure communications protocol to prevent unauthorized access: Privacy and authentication for wireless local area networks. *IEEE Personal Communications*, pages 25–31, First Quarter 1994.

[13] A. Basyouni. Analysis of wireless cryptographic protocols. Master's thesis, Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario, Canada, August 1997.

[14] A. Basyouni and S. Tavares. Public key versus private key in wireless authentication protocols. In *Proceedings of the Canadian Workshop on Information Theory*, Lecture Notes in Computer Science, No. 950, pages 41–44, Toronto, Canada, 1997.

[15] M. J. Beller, L.-F. Chang, and J. Yacobi. Privacy and authentication on a portable communications systems. *IEEE Journal on Selected Areas in Communications*, 11(6):821–829, August 1993.

[16] M. J. Beller and J. Yacobi. Fully-fledged two-way public key authentication and key agreement for low-cost terminals. *Electronics Letters*, 29(11):999–1001, May 27th 1993.

[17] A. Briyukv and A. Shamir. Real time cryptanalysis of the alleged A5/1 on a PC. `http://cryptome.org/a51-bs.htm`, December 1999.

[18] B. Lampson R. Rivest B. M. Thomas C. M. Ellison, B. Franz and T. Ylonen. Spki requirements, spki certificate theory, simple public key certificate, spki example. `http://www.clark.net/pub/cme/html/spki.html`, October 1998.

[19] U. Carlsen. Optimal privacy and authentication on a portable communications system. *Operating Systems*, June 1994.

[20] Certicom. SigGen Smart Card. `http://205.150.149.57/ce2/embed.htm`, 1997.

[21] Certicom. White papers on elliptic curve cryptography. `http://www.certicom.com`, 2000.

[22] H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In K. Ohta and D. Pei, editors, *Advances in Cryptology – ASIACRYPT 98*, Lecture Notes in Computer Science, No. 1514, pages 51–65. Springer-Verlag, Berlin, Germany, 1998.

[23] R. E. Crandall. Method and apparatus for public key exchange in a cryptographic system. U.S. Patent Number 5,463,690, October 1995.

[24] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, November 1976.

[25] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[26] EIA/TIA-IS-54-B.

[27] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, July 1985.

[28] WAP Forum. The wireless application protocol. `http://www.uplanet.com/pub/feb99WAPWP.pdf`, 1999.

[29] R. Gallant, R. Lambert, and S. Vanstone. Improving the parallelized Pollard lambda search on binary anomalous curves. Draft Article, April 7, 1998.

[30] V. K. Garg and J. E. Wilkis. *Wireless and Personal communications Systems.* Prentice-Hall, Upper Saddle River, NJ, 1996.

[31] J. Dj. Golić. Crytanalysis of alleged A5 stream cipher. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT 97*, Lecture Notes in Computer Science, No. 1233, pages 239–255. Springer-Verlag, Berlin, Germany, 1997.

[32] D. Guinier. From eavesdropping to security on the cellular telephone system GSM. Internet draft, June 1997.

[33] O. Gunasekara. Smart phone challenges. `http://www.arm.com/Documentation/WhitePapers/SmartPhone`, 1997.

[34] O. Gunasekara. Developing a digital cellular phone using a 32-bit microcontroller. `http://www.arm.com/Documentation/WhitePapers/CellPhone`, 1998.

[35] G. Harper, A. Menezes, and S. Vanstone. Public-key cryptosystems with very small key lengths. In R. A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT 92*, Lecture Notes in Computer Science, No. 658, pages 163–173. Springer-Verlag, Berlin, Germany, 1992.

[36] T. Hasegawa, J. Nakajima, and M. Matsui. A practical implementation of elliptic curve cryptosystems over $GF(p)$ on a 16-bit microcomputer. In H. Imai and Y. Zheng, editors, *First International Workshop on Practice and Theory in Public Key Cryptography*, Lecture Notes in Computer Science, No. 1431, pages 182–194. Springer-Verlag, Berlin, Germany, 1998.

[37] IEEE P1363. Standard specifications for public-key cryptography. Draft Version 7, September 1998.

[38] ARM Incorporated. *Advanced RISC Machines Architectural Reference Manual.* Prentice-Hall, New York, NY, 1996.

[39] D. Jaggar. ARM architecture and systems. *IEEE Micro*, pages 9–11, July/August 1997.

[40] D. B. Johnson. Elliptic curve cryptography, future resiliency and high security systems. `http://www.certicom.com/research.html`, 2000.

[41] B. Kasim and L. Ertaul. Evaluation of GSM security. Technical report, February 2000.

[42] R. H. Katz. Security and privacy in wireless systems. `http://www.cs.berkeley.edu/~randy/`, 1996.

[43] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, January 1987.

[44] N. Koblitz. *A Course in Number Theory and Cryptography.* Springer-Verlag, Berlin, Germany, Second edition, 1994.

[45] Ç. K. Koç. High-Speed RSA Implementation. Technical Report TR 201, RSA Laboratories, 73 pages, November 1994.

[46] H. Krawczyk. SKEME: A versatile key exchange mechanism for Internet. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 114–127, February 1996.

[47] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. In *The 3rd Workshop on Elliptic Curve Cryptography (ECC 99)*, Waterloo, Canada, November 1–3 1999.

[48] A. Levi and M. U. Caglayan. An efficient and trust preserving public key infrastructure. To appear on *2000 IEEE Symposium on Security and Privacy*, May 2000.

[49] A. Li and O. Bukhres. Overall system for secure wireless mobile networks. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT 94*, Lecture Notes in Computer Science, No. 950, pages 351–353. Springer-Verlag, Berlin, Germany, 1994.

[50] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications.* Cambridge University Press, New York, NY, 1994.

[51] H.-Y. Lin and L. Harn. Authentication protocols for personal communication systems. *Computer Communication Review*, 25(4):256–261, 1996.

[52] M. Abai M. Burrows and R. Needham. A logic of authentication. *ACM Transaction on Computer Systems*, vol. 8, No. 1, 18–36, February 1990.

[53] O. Marttila and P. Vuorimaa. XML based mobile services. `http://www.tcm.hut.fi/~pv/`, 2000.

[54] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 1997.

[55] A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, Boston, MA, 1993.

[56] V. Miller. Uses of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology — CRYPTO 85, Proceedings*, Lecture Notes in Computer Science, No. 218, pages 417–426. Springer-Verlag, Berlin, Germany, 1985.

[57] R. Molva, D. Samfat, and G. Tsudik. An authentication protocol for mobile users. In *Colloquium on Security and Cryptography Applications to Radio Systems*, pages 62, 4/1–4/7, June 1994.

[58] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, April 1985.

[59] Y. Mu and V. Varadharajan. On the design of security protocols for mobile communications. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT 94*, Lecture Notes in Computer Science, No. 950, pages 135–145. Springer-Verlag, Berlin, Germany, 1994.

[60] R. Mullin, I. Onyszchuk, S. Vanstone, and R. Wilson. Optimal normal bases in $GF(p^n)$. *Discrete Applied Mathematics*, 22:149–161, 1988.

[61] C.-S. Park. On certificate-based security protocols for wireless mobile communication systems. *IEEE Network*, pages 50–55, September/October 1997.

[62] D. Samfat R. Molva and G. Tsudik. Authentication of mobile users. *IEEE Network*, pages 26–34, March/April 1994.

[63] S. M. Redl and M. K. Weber. *An Introduction to GSM.* Artech House, Norwood, MA, 1995.

[64] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[65] S. Sampei. *Applications of Digital Wireless Technologies to Global Wireless Communications.* Feher/Prentice-Hall, Upper Saddle River, NJ, 1997.

[66] E. Savaş and Ç. K. Koç. Efficient composite field arithmetic. Work in progress, August 1999.

[67] B. Schneier. *Applied Cryptography.* John Wiley & Sons, New York, NY, Second edition, 1996.

[68] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck. Fast key exchange with elliptic curve systems. In D. Coppersmith, editor, *Advances in Cryptology — CRYPTO 95*, Lecture Notes in Computer Science, No. 973, pages 43–56. Springer-Verlag, Berlin, Germany, 1995.

[69] S. Segars. ARM7TDMI power consumption. *IEEE Micro*, pages 12–19, July/August 1997.

[70] M. J. Wiener and R. J. Zuccherato. Faster attacks on elliptic curve cryptosystems. Draft Article, April 8, 1998.

[71] E. De Win, A. Bosselaers, S. Vandenberghe, P. De Gersem, and J. Vandewalle. A fast software implementation for arithmetic operations in GF(2^n). In K. Kim and T. Matsumoto, editors, *Advances in Cryptology — ASIACRYPT 96*, Lecture Notes in Computer Science, No. 1163, pages 65–76. Springer-Verlag, Berlin, Germany, 1996.

[72] CCITT Recommendation X.509. The directory-authentication framework. 1988.

Index