

# An Elliptic Curve Cryptography based Authentication and Key Agreement Protocol for Wireless Communication \*

M. Aydos, B. Sunar, and Ç. K. Koc  
Electrical & Computer Engineering  
Oregon State University  
Corvallis, Oregon 97331

## Abstract

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.

## 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 [9].

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 [8, 15, 18]. Several authentication protocols have been proposed for wireless mobile communication systems. Among them, the protocol of Beller, Chang, and Yacobi [5] 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 [2] also proposed a similar authentication protocol, which does not require pre-computation. These two protocols along with several others [6, 16, 20, 22, 17, 23] 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

---

\*This research was supported in part by Intel Corporation.

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 [24]. However, some of the security algorithms of the GSM architecture have recently been cryptanalyzed [26, 10].

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 §2, and then give a brief introduction to elliptic curve cryptography in §3. We give the details of the proposed protocol in §4, a discussion of its implementation in §5, and its comparison to the Beller-Chang-Yacobi and Aziz-Diffie protocols in §6. The conclusions are found in §7.

## 2 Security Requirements

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

### 2.1 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 [17]. 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.

### 2.2 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.

### 2.3 Confidentiality

In today's technology, commercially available scanners can easily intercept radio signals traveling over the air in wireless mobile communication systems [25]. 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.

## 2.4 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 they get their certificate from the Certificate Authority (CA). Then, the assigned identities can be used in that domain. The proposed protocol in this paper hides the identity of the portable device from an eavesdropper on the communication channel.

## 2.5 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 [4]. 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 memory required to store the necessary data should also be kept at a minimum in order to have an efficient protocol.

# 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 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 [19, 12].

## 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 scalar 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 scalar 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  $H(M_1, M_2)$ .

### 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 [11].

**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  $x_1 \in GF(2^k)$ , 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 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, for example, in the beginning of each month after paying the monthly charge.

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.

## 4.1 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 1.

**Figure 1:** Network Server Initialization.

SERVER		CERTIFICATION AUTHORITY
<ul style="list-style-type: none"> <li>• Choose <math>d_s \in [2, n - 2]</math></li> <li>• <math>Q_s = d_s \times P</math></li> <li>• Send</li> </ul>	$\xrightarrow{Q_s}$	<ul style="list-style-type: none"> <li>• Choose <math>k_s \in [2, n - 2]</math></li> <li>• <math>R_s = k_s \times P</math></li> <li>• Receive</li> <li>• Choose unique <math>I_s</math></li> <li>• <math>r_s = R_s \cdot x</math></li> <li>• <math>s_s = k_s^{-1}(H(Q_s \cdot x, I_s, t_s) + d_{ca} \cdot r_s)</math></li> <li>• Send</li> </ul>
<ul style="list-style-type: none"> <li>• Receive</li> <li>• <math>e_s = H(Q_s \cdot x, I_s, t_s)</math></li> <li>• Store <math>Q_s, Q_{ca}, I_s, (r_s, s_s), e_s, t_s</math></li> </ul>	$\xleftarrow{Q_{ca}, I_s, (r_s, s_s), t_s}$	

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 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 2.

**Figure 2:** User Terminal Initialization.

USER		CERTIFICATION AUTHORITY
<ul style="list-style-type: none"> <li>• Choose <math>d_u \in [2, n - 2]</math></li> <li>• <math>Q_u = d_u \times P</math></li> <li>• Send</li> </ul>	$\xrightarrow{Q_u}$	<ul style="list-style-type: none"> <li>• Choose <math>k_u \in [2, n - 2]</math></li> <li>• <math>R_u = k_u \times P</math></li> <li>• Receive</li> <li>• Choose unique <math>I_u</math></li> <li>• <math>r_u = R_u \cdot x</math></li> <li>• <math>s_u = k_u^{-1}(H(Q_u \cdot x, I_u, t_u) + d_{ca} \cdot r_u)</math></li> <li>• Send</li> </ul>
<ul style="list-style-type: none"> <li>• Receive</li> <li>• <math>e_u = H(Q_u \cdot x, I_u, t_u)</math></li> <li>• Store <math>Q_u, Q_{ca}, I_u, (r_u, s_u), e_u, t_u</math></li> </ul>	$\xleftarrow{Q_{ca}, I_u, (r_u, s_u), t_u}$	

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.

## 4.2 Mutual Authentication Between Terminal and Server

The protocols shown in Figures 1 and 2 are executed off-line. The mutual authentication and key agreement protocols between the terminal (user) and the server need to be executed in realtime. We give the combined protocol in Figure 3.

**Figure 3:** Mutual Authentication and Key Agreement.

USER		SERVER
• Receive	$\xleftarrow{Q_s}$	• Send
• Send	$\xrightarrow{Q_u}$	• 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 $g$
		• $C_0 = E(Q_k.x, (e_s, (r_s, s_s), t_s, g))$
• Receive	$\xleftarrow{C_0}$	• Send
• $D(Q_k.x, C_0)$		
• $C_1 = E(Q_k.x, (e_u, (r_u, s_u), t_u, g))$		
• Send	$\xrightarrow{C_1}$	• Receive
		• $D(Q_k.x, C_1)$
		• If $g$ 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
• $k_m = Q_k.x + g$		• $k_m = Q_k.x + g$
• $k_m$ : The unique secret key		• $k_m$ : The unique secret key

According to the protocol, whenever there is a service request either by the user or by the server, there is an immediate key exchange. 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 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  $Q_k.x$ . The server encrypts the concatenation of its certificate  $e_s, (r_s, s_s)$ , the certificate expiration date  $t_s$ , and a random number  $g$  which will be used to obtain the final mutual key of the communication. 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 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 and the random number  $g$ . 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  $g$ . 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  $g$  and the time certificate are valid. If not, it aborts. This mechanism, specifically the use of  $g$ , 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.

### 4.3 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 communication. 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 number  $g$  which is known by both sides to generate a new secret key without using the channel again. Both the server and the user perform a scalar addition 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.

## 5 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 [13]. An efficient method for performing the field multiplication is the Montgomery method [21] 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 [7]. 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  $GF(p)$  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 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 [19] 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 [1] 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 [27, 14].

## 6 Comparisons

Below we compare certain properties of the proposed protocol to those of Aziz-Diffie [2] and Beller-Chang-Yacobi [6] protocols. The performance values for the Aziz-Diffie and Beller-Chang-Yacobi protocols are found in [3], summarized in Figure 4.

**Figure 4:** 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

- 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 realtime portion of the protocols is given as follows:

Beller-Chang-Yacobi:	8320 bits (1024-bit key)
Aziz-Diffie:	8680 bits (1024-bit key)
Proposed:	1602 bits (160-bit key)

- 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 realtime 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 ECDSA (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
ECDSA:	Elliptic Curve Digital Signature Algorithm Verification
SKE:	Secret Key Encryption or Decryption

## 7 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 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 [19].

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. We plan to implement the protocol and provide performance results in the near future.

## References

- [1] 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.

- [2] 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.
- [3] A. Basyouni. Analysis of wireless cryptographic protocols. Master’s thesis, Department of Electrical and Computer Engineering, Queen’s University, Kingston, Ontario, Canada, August 1997.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] R. E. Crandall. Method and apparatus for public key exchange in a cryptographic system. U.S. Patent Number 5,463,690, October 1995.
- [8] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.
- [9] V. K. Garg and J. E. Wilkis. *Wireless and Personal communications Systems*. Upper Saddle River, NJ: Prentice-Hall, 1996.
- [10] 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. New York, NY: Springer-Verlag, 1997.
- [11] IEEE P1363. Standard specifications for public-key cryptography. Draft version 7, September 1998.
- [12] N. Koblitz. *A Course in Number Theory and Cryptography*. New York, NY: Springer-Verlag, Second edition, 1994.
- [13] Ç. K. Koç. High-Speed RSA Implementation. Technical Report TR 201, RSA Laboratories, 73 pages, November 1994.
- [14] Ç. K. Koç, B. Sunar, and E. Savaş. Fast finite field arithmetic and elliptic curve cryptography over composite fields. Manuscript submitted for publication, February 1998.
- [15] 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.
- [16] 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. New York, NY: Springer-Verlag, 1994.

- [17] H.-Y. Lin and L. Harn. Authentication protocols for personal communication systems. *Computer Communication Review*, 25(4):256–261, 1996.
- [18] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.
- [19] A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Boston, MA: Kluwer Academic Publishers, 1993.
- [20] 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.
- [21] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, April 1985.
- [22] 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. New York, NY: Springer-Verlag, 1994.
- [23] C.-S. Park. On certificate-based security protocols for wireless mobile communication systems. *IEEE Network*, pages 50–55, September/October 1997.
- [24] S. M. Redl and M. K. Weber. *An Introduction to GSM*. Norwood, MA: Artech House, Inc., 1995.
- [25] S. Sampei. *Applications of Digital Wireless Technologies to Global Wireless Communications*. Upper Saddle River, NJ: Feher/Prentice-Hall, 1997.
- [26] B. Schneier. *Applied Cryptography*. New York, NY: John Wiley & Sons, Second edition, 1996.
- [27] E. De Win, A. Bosselaers, S. Vandenberghe, P. De Gerssem, 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. New York, NY: Springer-Verlag, 1996.