Improved Authentication and Key Agreement Protocol Using Elliptic Curve Cryptography

A. Chandrasekar haichandruu@gmail.com
Research Scholar,
Anna University,
Chennai, India.

V.R. Rajasekar vrrsekar@yahoo.com
Lecturer – Information Technology,
Al Musanna College of Technology,
Al Muladha, 314, Sultanate of Oman.

V. Vasudevan vasudevan_klu@yahoo.co.in
Senior Professor & HOD, Information Technology,
A.K. College of Engineering,
Krishnankoil, India.

Abstract

The Elliptic Curve Cryptosystem (ECC) is an emerging alternative for traditional Public-Key Cryptosystem like RSA, DSA and DH. It provides the highest strength-per-bit of any cryptosystem known today with smaller key sizes resulting in faster computations, lower power consumption and memory. It also provides a methodology for obtaining high-speed, efficient and scalable implementation of protocols for authentication and key agreement. This paper provides an introduction to Elliptic Curves and how they are used to create a secure and powerful cryptosystem. It provides an overview of the three hard mathematical problems that provide the basis for the security of public key cryptosystems used today: the Integer Factorization Problem (IFP), the Discrete Logarithm Problem (DLP), and the Elliptic Curve Discrete Logarithm Problem (ECDLP). It explains the proposed protocol which is improved to reduce the storage requirements for establishing a shared secret key between two parties, to sign and verify a document and to establish a mutual authentication between two parties. The result of implementation is also discussed.

Keywords: ECC, ECDLP, IFP, Authentication, Key Agreement

1. INTRODUCTION

Elliptic Curve Cryptography (ECC) was first proposed by victor Miller [13] and independently by Neal Koblitz [10] in the mid-1980s and has evolved into a mature public-key cryptosystem. Compared to its traditional counterparts, ECC offers the same level of security using much smaller keys. This result in faster computations and savings in memory, power and bandwidth those are especially important in constrained environments. More significantly, the advantage of ECC over its competitors increases, as the security needs increase over time. Recently the National Institute of standards and Technology (NIST) approved ECC for use by the U.S. government [12]. Several standards organizations, such as Institute of Electrical & Electronics Engineers (IEEE), American National Standards Institute (ANSI), Open Mobile Alliance (OMA)

and Internet Engineering Task Force (IETF), have ongoing efforts to include ECC as a required or recommended security mechanism.

2. ELLIPTIC CURVE CRYPTOGRAPHY

At the foundation of every public-key cryptosystem is a hard mathematical problem that is computationally intractable. The relative difficulty of solving that problem determines the security strength of the corresponding system. The well known public-key cryptosystems like RSA, Diffie-Hellman and Digital Signature Algorithm (DSA) can all be attacked using sub-exponential algorithms, but the best known attack on ECC requires exponential time. For this reason, ECC can offer equivalent security with substantially smaller key sizes [1].

Public-key schemes are typically used to transport or exchange keys for symmetric-key ciphers. Since the security of a system is only as good as that of its weakest component, the work factor needed to break a symmetric key must match that needed to break the public-key system used for key exchange. Table 1 shows NIST guidelines [11] on choosing computationally equivalent symmetric and public key sizes.

Symmetric	ECC	RSA/DH/DSA	MIPS Yrs to attack	Protection Lifetime
80	160	1024	10^{12}	Until 2010
112	224	2048	10^{24}	Until 2030
128	256	3072	10^{28}	Beyond 2031
192	384	7680	10^{47}	Beyond 2031
256	512	15360	10^{66}	Beyond 2031

Table 1: Equivalent key sizes (in bits)

The use of 1024-bit RSA does not match the 128-bit or even 112-bit security level now used for symmetric ciphers. This underscores the need to migrate to larger RSA key sizes in order to deliver the full security of symmetric algorithms with more than 80-bit keys. Recent work by Shamir and Tromer [2] on integer factorization suggests that the migration needs to happen sooner than previously thought necessary. They estimate that a specialized machine capable of breaking 1024-bit RSA in less than one year can be built for $10 - $15 million dollars. Consequently, RSA Laboratories now considers 1024-bit RSA to be unsafe for data that must be protected beyond 2010 and recommends larger key for longer term protection [3]. At higher key sizes, RSA performance issues become even more acute. Since the performance advantage of ECC over RSA grows approximately as the cube of the key size ration, wider adoption of ECC seems inevitable.

Elliptic Curve (EC) as algebraic and geometric entities that have been studied extensively for the past 150 years and from these studies has emerged a rich and deep theory. Neal Koblitz as applied to cryptography first proposed EC systems in 1985 independently from the university of Washington and victor miller. EC are not ellipses. These are the curves described by cubic equations which are similar to those used for calculating the circumference of an ellipse. In simple, an ellipse curve is defined by an equation in z variables with coefficients. The cubic equations for EC's take the form

$$y^2+axy+by=x^3+cx^2+dx+e \qquad (1)$$

Where a, b, c, d and e are coefficients and x and y are variables. For cryptography the variables and coefficients are restricted to elements in a finite field. ECC operates over a group of points on an elliptic curve defined over a finite field. Its main cryptography operation is scalar multiplication, which computes $Q = k_P$ (a point P multiplied by an integer k resulting in another point Q on the

curve). Scalar multiplication is performed through a combination of point-additions and point-doublings. The security of ECC relies on the difficulty of solving the Elliptic Curve Discrete Logarithmic Problem (ECDLP), which states that given P and Q = k_P, it is hard to find k. Besides the curve equation, an important elliptic curve parameter is base point, G, which is fixed for each curve. In ECC, a large random integer k acts as private key, while the curve's base point G serves as the corresponding public key.

Every elliptic curve offers strong security properties and for some curves the ECDLP may be solved efficiently [9]. Since a poor choice of the curve can compromise security, standards organizations like NIST and Standard for efficient Cryptography Group (SECG) have published a set of curves [4, 12] that possess the necessary security properties. The use of these curves is also recommended as a means of facilitating interoperability between different implementations of a security protocol.

3. ELLIPTIC CURVE DIFFIE-HELLMAN

Elliptic Curve Diffie-Hellman 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 secret key to a secret-key encryption algorithm. Figure 1 Shows ECDH protocol.

User	Server
Choose $d_u \in [2,n-2]$	Choose $d_s \in [2,n-2]$
$Q_u = d_u \times P$	$Q_s = ds \times P$
Send (Q_u)	Receive (Q_u)
Receive (Q_s)	Send (Q_s)
$K = d_u \times Q_s = d_u\, d_s \times P$	$K = d_s \times Q_u = d_s\, d_u \times P$

Figure 1: Elliptic Curve Diffie-Hellman

The improved version given in Figure 2 provides a little more flexibility in the sense that the established value can be pre- selected by the user and sent to the server. The protocol steps can be modified slightly for sending a secret value from the server to the user.

User	Server
Choose $d_u \in [2,n-2]$	Choose $d_s \in [2,n-2]$
$e_u = d_u^{-1} \bmod n$	$e_s = d_s^{-1} \bmod n$
$Q = d_u \times K$	$Q = d_u \times K$
Send (Q)	
	Receive (Q)
	$R = d_s \times Q = d_s d_u \times K$
	Send (R)
Receive (R)	
$S = e_u \times R = e_u d_s d_u \times K = d_s \times K$	
Send (S)	
	Receive (S)
	$T = e_s \times S = e_s d_s \times K = K$

Figure 2: Elliptic Curve Diffie-Hellman (Improved)

4. ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM

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 the 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. The outline if the Elliptic Curve Digital Signature Algorithm (ECDSA) is given below, details of which can be found in [6].

4.1 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 key of the user A are (E, P, n, Q) and d, respectively.

4.2 ECDSA Signature Generation

The user A signs the message m using these 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 $s = k^{-1} (H(m) + dr) \bmod n$, Where H is the SHA.
   If s=0 go to step1.
5. The signature for the message m is the pair of integers(r,s).

4.3 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 $u_1 = H(m)c \bmod n$ and $u_2 = r\,c \bmod n$.
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.

5. PROPOSED PROTOCOL

Almost in all the security protocols, we assume that there is a certification 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. It is 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 protocol should be designed in such a way that the users can easily be authenticated on-line via their home networks.

5.1 Server initialization

In order to receive a certificate, the Server sends its public key Qs and 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 Is, and the certification expiration dates. The CA then sends the signed message through the secure and authenticated channel to the user as shown in Figure 3.

Server Action

Choose $d_s \in [2,n-2]$
$Q_s = d_s \times P$
Send (Q_s)
Receive $Q_{ca}, I_s, (r_s,s_s), t_s$
$e_s = H(Q_s.x, I_s, t_s)$
Store $Q_s, Q_{ca}, I_s, (r_s,s_s), e_s, t_s$.

Certification Authority action

Choose $k_s \in [2,n-2]$
$R_s = k_s \times P$
Receive (Q_s)
Choose Unique I_s
$r_s = R_s.x$
$S_s = k_s^{-1}(H(Q_s.x, I_s, t_s)+d_{ca}.r_s)$
Send $(Q_{ca}, I_s, (r_s,s_s), t_s$

Figure 3: Server Initialization

5.2 User Initialization

Establishing a secure channel from the CA to the server is a common and accepted assumption in almost all authentication protocol. 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 user which is used in future authentication and key generation process. By repeating the very same process the user acquires the certificate as shown in figure 4.

User Action

Choose $d_u \in [2, n-2]$
$Q_u = d_u \times P$
Send (Q_u)
Receive $Q_{ca}, I_u, (r_u, s_u), t_u$
$e_u = H(Q_u.x, I_u, t_u)$
Store $Q_u, Q_{ca}, I_u, r_u, s_u, e_u, t_u$.

Certification Authority action

Choose $k_u \in [2,n-2]$
$R_u = k_u \times P$
Receive (Q_u)
Choose Unique I_u
$r_u = R_u.x$
$s_u = k_u^{-1}(H(Q_u.x, I_u, t_u)+d_{ca}.r_u)$
Send $(Q_{ca}, I_u, (r_u, s_u), t_u$

Figure 4: User Initialization

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 elliptic curve points R_u and R_s respectively. As mentioned earlier, the proposed protocol is based on the ECDSA.

5.3 Mutual Authentication between User and Server

The mutual authentication and key agreement protocols between the user and the server need to be executed in real-time. The above two protocols User initialization and Sever initialization are combined together as a single protocol, which is given in Figure 5. In this protocol a secret-key encryption algorithm is used to encrypt the data in the protocol. A conventional stream cipher or a block cipher 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 cipher text C are denoted as C= E(K,M) and M= D(K,C), respectively.

In this 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 be sending random challenge to the initiating party. Sending the public keys does not introduce any threat to the security of the system. Once both the sides have the other party's public key, they simultaneously generate a secret key to encrypt the data required to have a mutual authentication. To protect the certificates, it is necessary to send the certificates in encrypted form. To encrypt certificates the protocol uses a secret key cipher which is a mutually agreed secret key. The server encrypts the concatenation of its certificate, the certificate expiration date and a random number 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 in clear in almost all the other authentication protocols. In the proposed protocol the encryption time of the certificate increases slightly.

The encrypted message is then sent to the user. The user then decrypts and obtains the certificate of the servers, the random number and the challenge 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, the certificate expiration date, and the random number. This encrypted data is sent to the server.

User	Server
Receive (Q_s)	Send (Q_s) Public Key of Server
Send (Q_u) Public key of User	Receive (Q_u)
$Q_k = d_u \times Q_s \times P$	$Q_k = d_s \times Q_u \times P$
$Q_k.x$: The Mutually agreed Key	$Q_k.x$: The Mutually agree Key
	Generate random number g
	$C_0 = E(Q_k.x, (e_s,(r_s,s_s),t_s,g))$
Receive (C_0)	Send (C_0)
$D(Q_k.x, C_0)$	
$C_1 = E(Q_k.x, (e_u,(r_u,s_u),t_u,g))$	
Send (C_1)	Receive (C_1)
	$D(Q_k.x,C_1)$
	If g and t_u are valid, then
	$c = s_u^{-1}$
$c = s_s^{-1}$	Compress (g)
$u_1 = c.e_s$	$u_1 = c.\ e_u$
$u_2 = c.\ r_s$	$u_2 = c.\ r_u$
$R = u_1 \times P + u_2 \times Q_{ca}$	$R = u_1 \times P + u_2 \times Q_{ca}$
$v = R.x$	$v = R.x$
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

Figure 5: Mutual Authentication between Server and User

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 and checks whether the random number generated by the server and the time of the certificate are valid. If not, it aborts. This mechanism, specifically the use of random number, defeats spoofing attacks by the user side and also prevents unnecessary computation. Then, the server checks the validity of the certificate and accordingly grants or aborts the service. It may be a good approach to generate multiple random numbers in advance so that the protocol could save some time. However, storing these multiple random numbers will increase the storage requirement of the protocol. This is the main drawback which is available in the existing protocols [7,8]. The above draw back is removed in the proposed protocol by applying the message compression technique. A compression technique is applied to reduce the storage of the random numbers.

5.3.1 Key Agreement

Once the verification procedure is completed by the user and the server, then a secret key known by each side to encrypt the communication is to be generated. A new key exchange step to agree on a unique key to be used for communication during each session. We will use the previously generated random number which is known by both the 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; this key is used for encrypting the data sent through the channel.

6. IMPLEMENTATION RELATED ISSUES

Elliptic curve cryptographic algorithms is defined over the finite filed GF(2^k). ECC applications require fast hardware and software implementations of the arithmetic operations in GF(2^k) for large values of k. An implementation method for this case was presented in [13], where the authors propose to use the logarithmic table lookup method for the ground field GF (2^n) operations. The filed GF(2^{nm}) is then constructed using the polynomial basis, where the elements of GF(2^{nm}) are polynomials of degree m-1 whose coefficient are from the ground filed 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 the similar methodology for implementing the arithmetic operations in GF(2^{mn}) is used. The only difference is that an optimal normal basis in GF(2^m) to represent the elements of GF(2^{nm}) by taking the ground filed 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 which are given in [5].

Addition, multiplication and inversion operations are implemented 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++ and executed on the PC with 548 MHz, Pentium III Processor, running Windows Xp. The timing results are given below in the Figure 6. The results of implementation were compared with the results of [7,8], which was implemented on the PC with 300 MHz, Pentium II Processor. It result shows that both the proposed protocol and the protocol given in [8] having the same timings but the proposed system takes a lower storage requirement for the user side the protocols proposed in [7].

Operation	Proposed-Timings	Timings given in [8]
EC Addition	80µsec	80 µsec
EC Doubling	80 µsec	80 µsec
EC Multiplication	25 msec	25 msec

Protocols	Storage
Proposed	1120 bits
Protocol Proposed in [7]	1440 bits

Figure 6: Result comparison with [7] & [8]

7. CONSLUSION & FUTURE WORK

The proposed protocol for Authentication and key agreement is based on ECC, which is a public-key type. The public key cryptography concept solves the key distribution and storage problems. The protocol provides certain security services like non-repudiation, anonymity of user and service expiration mechanism using time certificates. The RSA-based protocols have significant problems in terms of the storage requirements. The use of ECC will decrease the storage requirements for the execution of the protocols. The use of ECC with compression techniques will further reduce the storage requirements and it is highly recommended for the future developments with regard to the network security protocols, the proposed protocol is a step in this direction. The future work of this paper will be implementing the protocol in real-time and providing the performance results.

8. REFERENCES

[1] A. Lenstra and E. Verheul, "Selecting Cryptographic Key Sizes", Journal to Cryptology 14 (2001) pp. 255 – 293, http:/www.cryptosavvy.com/

[2] A. Shamir and E. Tromer, "Factoring Large Numbers with the TWIRL Device", Crypto 2003, LNCS 2729, Springer-Verlag, Aug.2003.

[3] B. Kaliski, "TWIRL and RSA Key size", RSA Laboratories Technical Note, May 2003. http://rsasecurity.com/rsalabs/technotes/twirl.html.

[4] Certicom Research, "SEC 2: Recommended Elliptic Curve Domain Parameters", Standards for efficient Cryptography, Version 1.0, Sep. 2000.

[5] E. De Win. A. Bosselars, S. Vandenberghe P. De Gersem and J. Vandewalle. A fast software implementation for arithmetic operations in GF (2n). In K. Kim and T. Matsumoto, editors, Advances in Cryptology – ASIACRYPT 96, Lecture notes in computer Science, N0. 1163, Pages 65 – 76. New York, NY: Springer – Verlag, 1996.

[6] IEEE P 1363. Standard Specifications for Public-Key Cryptography. Draft version 7, September 1998.

[7] M. Aydos, B. Sunar and C.K. Koc, "An Elliptic Curve Cryptography based Authentication and Key agreement Protocol for wireless communication", 2nd International workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Dallas, Texas, October, 30, 1998.

[8] M. Aydos, E. Savas and C.K. Koc, "Implementing Network Security Protocols based of Elliptic Curve Cryptography", Proceedings of the fourth symposium on computer networks, Pages 130 – 139, Istanbul, Turkey, May 20 – 21, 1999.

[9] N. Smart, "How secure are elliptic curves over composite extension fields?", EUROCRYPT 2001, LNCS 2045 Springer-Verlag, pp. 30- 39, 2001.

[10] N.Koblitz, "Elliptic curve cryptosystems", Mathematics of Computation, 48:203-209, 1987.

[11] NIST, "Special Publication 800-57: Recommendation for Key Management. Part 1: General Guideline", Draft Jan.2003.

[12] U.S. Dept of Commerce/NIST, "Digital Signature Standard (DSS)", FIPS PUB 186-2, Jan. 2000.

[13] V. Miller, "Uses of elliptic curves in cryptography", Crypto 1985, LNCS218: Advances in Cryptology, Springer-Verlag, 1986.