Implementation of Elliptic Curve Digital Signature Algorithms

1Temitope O.S. Olorunfemi, 2B.K. Alese, 3S.O. Falaki and 4O. Fajuyigbe
1Computer Centre, Adekunle Ajaysin University, Akungba-Akoko, Nigeria
2Department of Computer Science, Federal University of Technology, Akure, Nigeria

Abstract: Elliptic Curve Digital Signature Algorithms (ECDSA) have recently come into strong consideration, particularly by the standards developers, as alternatives to established standard cryptosystems such as the integer factorization cryptosystems and the cryptosystems based on the discrete logarithm problem. Crypto-algorithms are always the most important core tool in security applications. The elliptic curve-based digital signature algorithms were implemented in this study using the open source software from GNU’s Not Unix (GNU) Compiler collection version 3.4.4-1 (2005). The introduction of Cygwin version 1.5.18-1 (2005) release into this study, also enable users of Microsoft windows to make use of the software. ECDSA was examined and the main reason for the attractiveness of elliptic curve cryptography was brought out in the fact that there is no sub-exponential algorithm so far known to solve the elliptic curve discrete logarithm problem on a properly chosen elliptic curve. Hence, it takes full exponential time to solve while the best algorithms known for solving the underlying integer factorization and discrete logarithm problem both take sub-exponential time. The ECDSA have a smaller key size, which leads to faster computation time and reduction in processing power, storage space and bandwidth. This makes ECDSA ideal for constrained environments such as pagers, cellular phones and smart cards.

Keywords: Elliptic curve cryptosystems, digital signature, wireless, internet security

INTRODUCTION

Cryptology (from the Greek kryptós logos, meaning hidden word') is the study of codes. Cryptology is also the discipline of cryptography and cryptanalysis combined. Cryptology is of ancient origin. The Jewish writers sometimes concealed the meaning of their writing by alphabet reversal algorithm by using the last letter of the alphabet in place of the first, the next last for the second and so on. This system, called atbash, is exemplified in the Bible around 666BC, in Jeremiah 25:26, in which Sheshach is written for Babel (Babylon), using the second and twelfth letters from the end of the Hebrew alphabet instead of from the beginning (The Holy Bible, 2004).

Around 50 BC, Julius Caesar developed the idea of transposing letters in the alphabet in order to transmit military messages with relative security (James, 2005). Cryptography today might be summed up as the study of techniques and applications that depend on the existence of difficult problems. Cryptanalysis is the study of how to compromise (defeat) cryptographic mechanisms and to most people, cryptography is concerned with keeping communications secure. Indeed, the protection of sensitive communications has been the emphasis of cryptography throughout much of its history (Kalm, 1967).

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication.

Corresponding Author: Temitope O.S. Olorunfemi, Computer Centre, Adekunle Ajaysin University, Akungba-Akoko, Nigeria Tel: +234-803-6750017 +234-805-85-31677
Cryptography is according to Aydos (2000), in general, the science of concealing data. However, Alesa (2004) defined it as the art and science of concealing data which is further asserted by Rabah (2004) and this same view is shared by Olorunfemi (2006).

Encryption is the transformation of data into a form that is as close to impossible to read without the appropriate knowledge. Its purpose is to ensure security by keeping information hidden from anyone for whom it is not intended, even those who have access to the encrypted data. Traditionally, secrecy has meant security, however Olorunfemi and Oladipo (2005) stated that this might be so in theory but not necessarily in practice. Decryption is the reverse of encryption; it is the transformation of encrypted data back into an intelligible form.

**Digital Signature**

People use public-key cryptography to compute digital signatures by associating something unique with each person. When public-key cryptography is used to encrypt a message, the sender encrypts the message with the public key of the intended recipient. When public-key cryptography is used to calculate a digital signature, the sender encrypts the digital fingerprint of the document with his or her own private key. Anyone with access to the public key of the signer may verify the signature.

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. Digital signatures are much more secure than hand-written signatures. There is no known way to forge a digital signature. Another advantage that digital signatures have over hand-written signatures is that the signature is applied to the entire message. The signing key perturbs every bit of the digital message. A hand-written signature is applied to the bottom of a paper document. There is nothing that prevents alteration of the text appearing above the penned signature while leaving the signature intact. Such alterations are not possible with digital signatures. Changing even a single bit of a signed message will cause the verification procedure to fail.

Diffie and Hellman introduced a key exchange protocol in their first publication along with their ideas of public key cryptography. Their protocol is known as Diffie-Hellman key exchange. Diffie and Hellman (1976a and b) introduced the concept of a digital signature in 1976. Although the idea of a digital signature was clearly articulated, no practical realization emerged until the 1978 paper by Rivest, Shamir and Adleman (Rivest et al., 1978). Digital signatures appear to have been independently discovered by Merkle earlier but not published until 1978 (Merkle, 1978, 1979). Other early research was due to Lamport (1979), Rabin (1978, 1979) and Matyas (1979).

Diffie and Hellman (1976a) suggested that public key schemes would ultimately be of more importance to the business community than the confidentiality services for which cryptography had traditionally been used. The digital signature schemes in use today can be classified according to the hard underlying mathematical problem, which provides the basis for their security:

**Integer Factorization (IF) Schemes**

They base their security on the intractability of the integer factorization problem. Examples include RSA (Rivest et al., 1978; Rabin, 1979) Signature Schemes.

**Discrete Logarithm (DL) Schemes**

They base their security on the intractability of the (ordinary) discrete logarithm problem in a finite field. Examples of this include the (ElGamal, 1985; Schnorr, 1991; NIST, 1994; Nyberg and Rueppel, 1993, 1996).
Elliptic Curve (EC) Schemes

They base their security on intractability of the elliptic curve discrete logarithm problem. An example is the Elliptic Curve Digital Signature Algorithm (Varstone, 1992; Johnson and Menezes, 1999) being implemented in this study. The elliptic curve scheme is definitely the most recent of all the various schemes (Alese, 2004, Rabah, 2005a, b).

ELLIPITC CURVE CRYPTOGRAPHIC SYSTEMS

Elliptic curve cryptosystems were first proposed independently by Miller (1986) and Koblitz (1987a) in the mid-1980s. Elliptic curve cryptography constitutes a fundamental and efficient technology for public key cryptosystems (Konstantinos et al., 2003). At a high level, they are analogs of existing public-key cryptosystems in which modular arithmetic is replaced by operations defined over elliptic curves.

The elliptic curve cryptosystems that have appeared in the literature can be classified into two categories according to whether they are analogs to the RSA system or to discrete logarithm based systems. Just as in all public-key cryptosystems, the security of elliptic curve cryptosystems relies on the underlying hard mathematical problems. The security of such systems depends on the following hard problem: Given two points G and Y on an elliptic curve such that Y = kG (that is, Y is G added to itself k times), find the integer k. This problem is commonly referred to as the elliptic curve discrete logarithm problem.

Presently, the methods for computing general elliptic curve discrete logarithms are much less efficient than those for factoring or computing conventional discrete logarithms. As a result, shorter key sizes can be used to achieve the same security of conventional public-key cryptosystems, which might lead to better memory requirements and improved performance. One can easily construct elliptic curve encryption, signature and key agreement schemes by making analogs of ElGamal, DSA and Diffie-Hellman. These variants appear to offer certain implementation advantages over the original schemes and they have recently drawn more and more attention from both the academic community and the industry (Rohshaw and Lin, 1997; Menezes, 1995).

Mathematicians have studied elliptic curves for more than a century. Besides their recent cryptographic applications, they are used in primality testing and integer factorization. The majority of the products and standards that use public key cryptography for encryption and digital signatures use RSA. However, the bit length for secure RSA use has increased over recent years and this has increased the processing load on applications using RSA.

Recently, the Elliptic Curve Cryptology (ECC) has begun to challenge RSA. The security of ECC rests on the discrete logarithm problem over the points of an elliptic curve. When the curve is defined over a finite field, the points of the curve form an abelian group. The addition of the points can be implemented efficiently both in software and hardware.

As it is in the case with the integer factorization problem and the discrete logarithm problem modulo p, no efficient algorithm is known to solve the elliptic curve discrete logarithm problem. Of the three problems, the integer factorization and the discrete logarithm problem modulo p both admit general algorithms that run in sub-exponential time. This means the problem is still hard, but not as hard as those problems that admit only fully exponential algorithms. The best general algorithm for the elliptic curve discrete logarithm problem is fully exponential time. Therefore, cryptosystems that rely on the elliptic curve discrete logarithm problem provide higher strength-per-bit than the other cryptosystems that rely on the integer factorization problem and the discrete logarithm problem modulo p.

The use of elliptic curve in cryptography means having shorter key lengths which means smaller bandwidth and memory requirements, which is a crucial factor in some applications such as...
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 (Menezes, 1993).

The security of many public-key cryptosystems relies on the apparent intractability of the computational problem they are based upon. In a cryptographic setting, it is prudent to assume that the adversary is very powerful. Thus, a computational problem is said to be easy or tractable if it can be solved in (expected) polynomial time, at least for a non-negligible fraction of all possible inputs. In other words, if there is an algorithm that can solve a non-negligible fraction of all instances of a problem in polynomial time, then any cryptosystem whose security is based on that problem must be considered insecure.

**Mathematics of Elliptic Curves over Real Numbers**

The Weierstrass form of an elliptic curve equation (Silverman, 1985):

\[ Y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \]  \hspace{1cm} (1)

The variables \( x \) and \( y \) cover a plane. In fact, \( x \) and \( y \) can be complex, real, integers, polynomial basis, optional normal basis, or any other kind of field element. This same equation can be given over prime field and binary field as follows:

\begin{align*}
\text{over } \mathbb{GF}(p), & \quad \text{with } a_i, b \in \mathbb{GF}(p) \text{ and } 4a^3 + 27b^2 \neq 0; \\
\text{over } \mathbb{GF}(2^m), & \quad \text{with } a_i, b \in \mathbb{GF}(2^m) \text{ and } b \neq 0
\end{align*}  \hspace{1cm} (2)

The interest of this study is only in a few special cases of Eq. (1), hence we stick to some aspects of the mathematics. In real numbers on the real plane, a simple form of Eq. (1), which will be applicable:

\[ Y^2 = x^3 + a_4x + a_6 \]  \hspace{1cm} (3)

As an example, the plot for the curve, \( a_4 = -7, a_6 = 5 \), for \( x \) and \( y \) in the set of real numbers. To plot the curve, we only need to find:

\[ y = \sqrt{x^3 - 7x + 5} \]  \hspace{1cm} (4)

and plot both negative and positive values of \( y \) for the same \( x \). (Fig. 1)

The basic idea is to find a way to define addition of two points that lie on the curve such that the sum is another point on the curve. If we can do that and invent an identity element, then we have algebra—that is, a higher level of abstraction but following the same basic rules.

The identity element \( O \) is the point that, added to any point on the curve, gives the same point back:

\[ P + O = P. \]  \hspace{1cm} (5)

It is also called the point at infinity. Under normal conditions we will never use this point in real code. The formulas to be presented later won't work if \( O \) is an input.
Mathematics of Elliptic Curves Over Prime Fields

All calculations are based on modulo a large prime (Morain, 1991) instead of using floating-point arithmetic. This method has been investigated by the academic community and is mentioned in the IEEE P1363 (2000) crypto standard (Beth and Schaefer, 1991; Koyama \textit{et al.}, 1991). The major advantage of choosing this for computing elliptic curves is that the number of points on the curve can be computed easily.

The Order of an Elliptic Curve

Since the mathematics of a finite field only covers a finite set of points, the total number of points on an elliptic curve defined over a finite field is fixed. This number is sometimes called the cardinality of an elliptic curve.

The number of points on an elliptic curve over a finite field must satisfy Hasse’s Theorem. Given a field, $F_q$, the order of the curve ($N$) will satisfy this equation (Menezes, 1993; Koblitz, 1987b):

$$|N - (q + 1)| \leq 2\sqrt{q}$$  \hspace{1cm} (6)

Another way to put it is:

$$q + 1 - 2\sqrt{q} \leq N \leq q + 1 + 2\sqrt{q}$$  \hspace{1cm} (7)

So the number of points on the curve is approximately the field size. Both order and cardinality of a curve mean the total number of points that satisfy a specific equation. The order of a point is defined by the number of times we can add the point to itself until we get to the point at infinity. The order of any point on a curve will evenly divide the order of the curve. If the order of the curve is smooth, meaning lots of small factors, it is easier to crack.

Koyama \textit{et al.} (1991) gave a description of how to pick elliptic curves such that finding their order is trivial while Rosing (1999) gave a good coverage on the implementation of elliptic curve in cryptography.

Digital Signature Algorithm (DSA)

The DSA is due to Kravitz (1993) and was proposed as a Federal Information Processing Standard in August of 1991 by the U.S. National Institute for Science and Technology. It became the
Digital Signature Standard (DSS) in May 1994, as specified in FIPS-86, NIST1994; Smid and Brunstad (1992) commented that the DSA was selected based on a number of important factors: the level of security provided, the applicability of patents, the ease of export from the U.S., the impact on national security and law enforcement and the efficiency in a number of government and commercial applications. They provide a comparison of the computational efficiencies of DSA and RSA and address a number of negative responses received during the FIPS public comment period. Naccache et al. (1995) describe a number of techniques for improving the efficiency of the DSA. B'eguin and Quisquater (1995) show how to use an insecure server to aid in computations associated with DSA signature generation and verification. The method accelerates the computation of modular multiplication and exponentiation by using an untrusted auxiliary device to provide the majority of the computing. As such, it also applies to schemes other than DSA. Arazi (1993) shows how to integrate a Diffie-Hellman key exchange into the DSA.

**Elliptic Curve Digital Signature Algorithms (ECDSA)**

The Digital Signature Algorithm (DSA) was specified in a United State Government Federal Information Processing Standard (FIPS) called as the Digital Signature Standard (DSS). DSA’s security relies on the discrete logarithm problem in the prime-order subgroups of $\mathbb{Z}_p^*$. ECDSA was first proposed by Vanstone (1992) in response to National Institute of Standards and Technology’s (NIST) request for public comments on their first proposal of Digital Signature Standard (DSS). It was accepted in 1998 as an International Standard Organization (ISO) standard (ISO 14888-3, 1998), it was also accepted in 1999 as an American National Standard Institute (ANSI) standard (ANSI X9.62, 1999) and accepted in 2000 as an Institute of Electrical and Electronics Engineers (IEEE) standard (IEEE 1363-2000) and a FIPS standard, FIPS 186-2 (NIST, 2000).

Digital signature schemes are designed to provide the same functions of handwritten signatures for the digital environment. A digital signature is a number that is calculated from a secret known only by the signer and from the contents of the message that is signed. The signature should be verifiable by other parties without having any knowledge of the secret key. The signature generated should be unforgeable to prevent the signer from repudiating a signature he/she created and others from claiming that the signature is his/her signature.

The digital signature schemes can provide data integrity, data origin authentication and non-repudiation, but not used to provide confidentiality.

**ECDSA Key Generation**

The user A follows these steps where p is a large prime:

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

The other parties can check if the public key is valid by:

- Checking that $Q \neq 0$.
- Checking that $x_Q$ and $y_Q$ are properly represented elements of $F_q$.
- Checking that Q is on the elliptic curve defined by a and b.
- Checking that $nQ = Q$.

If any of these checks fail the public key Q is invalid, otherwise Q is valid. The following procedure describes how to generate the signature.
**ECDSA Signature Generation**

The user A signs the message \( m \) using the following steps as shown in Fig. 2:

- Select a pseudorandom integer \( k \in [1, n-1] \).
- Compute \( k \times P = (x_1, y_1) \) and \( r = x_1 \mod n \).
  If \( x_1 \in \text{GF}(2^m) \), it is assumed that \( x_1 \) is represented as a binary number.
  If \( r = 0 \) then go to Step 1.
- Compute \( k^{-1} \mod n \).
- Compute \( s = k^{-1}(H(m) + d \cdot r) \mod n \).
  Here \( H \) is the secure hash algorithm SHA-1.
  If \( s = 0 \) go to Step 1.
- 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 as shown in Fig. 3:

- Verify that \( r \) and \( s \) are integers in the interval \([1, n-1]\).
- Compute \( c = s^1 \mod n \) and \( H(m) \).

---

**Fig. 2: Signature generation schematic**
Fig. 3: Signature verification schematic

- Compute \( u_1 = H(m) \cdot c \mod n \) and \( u_2 = r \cdot c \mod n \).
- Compute \( u_1 \times P + u_2 \times Q = (x_i, y_i) \) and \( v = x_i \mod n \).
- Accept the signature if \( v = r \).

**ECC DOMAIN PARAMETERS**

The operation of public-key cryptographic schemes involves arithmetic operations on an elliptic curve over a finite field determined by some elliptic curve domain parameters. ECC domain parameters over \( F_q \) (where \( F_q \) is either \( F_p \) and \( F_2^p \)) are a septuple:

\[ T = (q, FR, a, b, G, n, h) \]

Consisting of a number \( q \) specifying a prime power (\( q = p \) or \( q = 2^p \)), an indicator \( FR \) (field representation) of the method used for representing field elements in \( F_q \), two field elements \( a \) and \( b \) in \( F_q \) that specify the equation of the elliptic curve \( E \) over \( F_q \) (i.e., \( y^2 = x^3 + ax + b \) in the case \( p > 3 \) and \( y^2 + xy = x^3 + ax^2 + b \) when \( p = 2 \)), a base point \( G = (x_0, y_0) \) on \( E(F_q) \), a prime \( n \) which is the order of \( G \) and an integer \( h \) which is the co factor \( h = \# E(F_q)/n \).
Operating System

Cygwin version 1.5.18-1 of 2005 which is an open source software was downloaded freely from www.cygwin.com and was ported in Microsoft Windows XP environment.

What Is Cygwin?
Cygwin is a Linux-like environment for Windows. It consists of two parts:

- A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API functionality.
- A collection of tools, which provide Linux look and feel.

The Cygwin DLL works with all non-beta, pre-release candidate, i686 32 bit versions of Windows since Windows 95, with the exception of Windows CE.

Software Language

Software was developed using the open source GCC (GNU Compiler Collection) 3.4.4-1 (2005) version that came with the cygwin downloaded on the internet. GNU is a recursive acronym for GNU's Not UNIX while GCC, the GNU Compiler Collection, includes front ends for C, C++, Objective-C, Fortran, Java and Ada, as well as libraries for these languages (libstdc++, libgcc,...).

GCC development is a part of the GNU Project, aiming to improve the compiler used in the GNU system including the GNU/Linux variant. The GCC development effort uses an open development environment and supports many other platforms in order to foster a world-class optimizing compiler, to attract a larger team of developers, to ensure that GCC and the GNU system work on multiple architectures and diverse environments and to more thoroughly test and extend the features of GCC. (http://www.gnu.org/ and http://gcc.gnu.org/).

Why Open Source?

It is beyond dispute that open source systems are potentially more secure than Windows, but the most important advantage do not come from openness per se. They come instead from the coincidence that open source systems, like Linux and BSD, are modeled on UNIX, which is designed in a more modular fashion than Windows. Such systems are more transparent to the user or administrator and have far fewer interdependencies - two factors that are exceptionally good for security.

The deep integration and multiple interdependencies among Windows components is a major security challenge in itself. It is the deep integration that makes the scores of exploits against the Internet Explorer browser so serious and difficult to fix. For example, the more modular architecture of UNIX and its open source codes enable developers to fix a major component without needing to re-work the kernel, or re-work other system components dependent on flawed code buried in the guts of the operating system.

Overall, the UNIX family of systems is designed to be immensely easier to monitor, to simplify and to administer for security. They feature fewer interdependencies, more transparency and better isolation of users.

So, while openness provides a couple of security advantages in itself, the chief reason why Linux and BSD offer superior security is not so much because they are open source, but because of Windows' method of providing security through obscurity. The idea, which is rooted in the fact that if people cannot see inside, they do not find the software's weaknesses, is rather inefficient and ineffective to hackers (Olorunfemi and Oladipo, 2005, http://www.addic3d.org/index.php?page=viewarticle&type=news&id=1284; www.serverwatch.com/tutorials/article.php/3526826).
Running of the Program

After successful compilation of the correct version of the software, an executable file is generated. After booting the system on to MS Windows XP environment, the cygwin command was invoked and thereafter we use cmd command of Windows XP. We now change to the appropriate directory and then type the executable file name generated initially. The results are as shown below.

ECDSA IMPLEMENTATION RESULTS

random point
x : ee85 233286c5 e13dbf14 3fe07ef9
y : 1dc37 fe616d43 4664353a 591e3972

Base point
x : 17a14 e4795736 42144d4c b4b6ab88
y : 4125 fc5ecc89 b894e8e9 48e459e

Signer's secret key : 6585 8b2a0b1f 16fd492f bf5f7149
Signers public key
x : 114d2 3dbe4192 7fa133c2 9b4352db
y : 5a7e e313412d 45ee5eae 1d5f276c

first component of dsa signature : 5eae a8032860 aaaa93fd4 237f4ace
second component of dsa signature : be37 14e04b1a d659603a fe4e83e9
Message Verifies

CONCLUSIONS

Elliptic Curve Digital Signature Algorithm (ECDSA) which is one of the variants of Elliptic Curve Cryptography (ECC) proposed as an alternative to established public-key systems such as Digital Signature Algorithm (DSA) and Rivest Shamir Adleman (RSA), have recently gained a lot of attention in industry and academia.

The main reason for the attractiveness of ECDSA is the fact that there is no sub-exponential algorithm known to solve the elliptic curve discrete logarithm problem on a properly chosen elliptic curve. Hence, it takes full exponential time to solve while the best algorithms known for solving the underlying integer factorization for RSA and discrete logarithm problem in DSA both take sub-exponential time. The keys generated by the implemented software is highly secured and it consumes lesser bandwidth because of small key size used by elliptic curves and this is also coupled with the introduction of open source software into this work, which is generally believed to be more secured than those traditionally available on closed source operating systems like Microsoft Windows.

Significantly smaller parameters can be used in ECDSA than in other competitive systems such as RSA and DSA, but with equivalent levels of security. Some benefits of having smaller key sizes include faster computation time and reduction in processing power, storage space and bandwidth. This makes ECDSA ideal for constrained environments such as pagers, personal digital assistants (PDAs), cellular phones and smart cards. These advantages are especially important in other environments where processing power, storage space, bandwidth, or power consumption are lacking.

The above properties makes communication to be more secure on the internet hence making electronic business and other transactions to be carried out with little or no fear of hackers.
ACKNOWLEDGMENT

Our sincere appreciation to Mike Rosing (Ph.D.) for his assistance and guidance.

REFERENCES

Digital Signature Algorithm.
EUROCRYPT ’91 (Berlin: Springer-Verlag), pp: 316-327.
Cygwin Dynamic Link Library available at www.cygwin.com
GNU Compiler Collection available at http://gcc.gnu.org/
Johnson, D. and A. Menezes, 1999. The elliptic Curve Digital Signature Algorithm (EDSA) Technical
on Elliptic Curves over the Ring Zn. In: EUROCRYPT ’91 (Berlin: Springer-Verlag, 1991),
pp: 252-266.
SRI International, Palo Alto.
Boston, MA.


