# AN EFFICIENT MODIFIED ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM

**Tilahun Kiros**
**Department of Computer Science and Engineering**
**Mekelle Institute of Technology**

**and**

**Kumudha Raimond**
**Department of Electrical and Computer Engineering**
**Addis Ababa University**

## ABSTRACT

*Many digital signatures which are based on Elliptic Curves Cryptography (ECC) have been proposed. Among these digital signatures, the Elliptic Curve Digital Signature Algorithm (ECDSA) is the widely standardized one. However, the verification process of ECDSA is slower than the signature generation process. Hence, the main objective of this work is to study ECDSA in order to improve its execution time. The method of the improvement is focused on the mathematical relationships of the algorithm in a manner that its verification process can be efficient. As a result, without affecting the underlying mathematical problem – the Elliptic Curve Discrete Logarithmic Problem (ECDLP) - a related efficient scheme is developed. The signature verification algorithm of the modified scheme is found to be faster than the verification process of ECDSA by 45%.*

***Keywords:*** *Digital signature, ECDSA algorithm, Elliptic curve cryptography, Scalar multiplication, Signature generation, Signature verification.*

## INTRODUCTION

With the advent of information technology, ensuring network and data communication security has become a crucial issue. Though the information technology provides us with various versatile tools for data manipulation and data storage, it is not without different facets of security attack. Thus, it is crucial to have tools that can insure the integrity of data, the confidentiality of data, and authenticity of any form of data communication.

To meet the requirements of network and data communication security, the cryptography science plays a great role. A variety of researches and applications of cryptography are developed in parallel with the advancement of the IT facilities. As a result, algorithms and techniques have been introduced to offer a better security mechanism. Algorithms like Rivest-Shamir-Adleman (RSA) [1, 6, 8], Digital Signature Algorithm (DSA)[1, 15], Diffie-Hellman (DH) [1, 7], and Elliptic Curve Cryptography (ECC) based schemes [4, 10] - like the ECDSA - are a few of the known cryptographic systems that are being employed in various applications. However, among these known cryptographic systems, ECC is emerging as an attractive and better alternative to the public-key cryptosystems [11, 12, 13, 14, 17, 18, 23]. ECC offers equivalent security with smaller key sizes resulting in faster computations.

The use of elliptic curves in modern public key cryptography was independently introduced by Neal Koblitz and Victor Miller in 1985 [2, 3, 4]. Since then, a lot of researches have been conducted in order to challenge its security strength and find out efficient ways of implementing ECC based cryptosystems. ECC has got increasing attention by the research community, as it offers equivalent security but shorter key size when it is compared with previously known systems like RSA and Discrete Logarithm (DL) – based cryptosystems. Though the confidence level of ECC is not equal to RSA as RSA has been around for above thirty years, it is widely believed that 160-bit ECC offers equivalent security of 1024-bit RSA.

In general, ECC has better per-bit security, and hence, suitable in constrained environments like smart cards and hand-held devices. ECC has less storage, power, and bandwidth requirements, and improved performance [20].

The rest of the paper is organized as follows: section 2 presents related works of elliptic curve digital signature. Preliminaries on ECC and ECDSA are presented in section 3. The proposed modified scheme is described in section 4.

Section 5 presents alternative form of the proposed scheme. Performance comparison of elliptic curve digital signature and the proposed modified scheme is presented in section 6, while section 7 concludes the paper.

## RELATED WORKS

Leading mathematicians and scientists have done a lot to ensure the robustness and correctness of many of the cryptographic schemes [9]. However, in [17], it is discussed that none of the mathematical problems – like the Integer Factorization Problem (IFP), the Discrete Logarithm Problem (DLP), and the Elliptic Curve Discrete Logarithm Problem (ECDLP) – are proven to be intractable. This article [17] underlines that it is based on our belief of their intractability that we rely on these algorithms, as no efficient algorithms are found to solve them. This article, [17], also assures that no sub-exponential algorithm is found to solve ECDLP.

Rosner discusses the implementation of GF[1] ($2^m$) based curves on a reconfigurable hardware [11]. It is shown that for GF ($2^{168}$), one point doubling operation takes 273 clock cycles. The work in [11] provides with some fundamental concepts for hardware implementation. The suitability of ECC based schemes for constrained devices and embedded systems is explained in [18]. Based on the per-bit security of ECC, this paper clarifies the advantage of ECC to achieve longer running battery operated devices with less heat, faster applications that consume less memory, and scaleable cryptographic applications. Moreover, the key-size comparison of ECC with RSA and DH based systems is given in [18]. In general, the advantage and performance comparison of ECC with RSA and DH schemes is provided in [12, 13, 14, 17, 18].

In [16], implementation of ECDSA on Advanced RISC Machines (ARM) processors for a curve on GF ($2^m$) is done. It is concluded that by using certain machine and curve specific techniques, the ECDSA signature can be made faster and optimized [16]. Similar work is done in [21] for a curve on GF ($p$). ARM processor implementation of curve $p$-224 is discussed in [21]. According to [21], it is concluded that 129.28ms was taken to perform point multiplication over the curve $p$-224 for C-based implementation. And the time was less for assembly language based implementation.

[1] Finite fields with $p^n$, for $p$ a prime integer and $n$ a positive integer are known as Galois Fields or GF.

Don B. Johnson, [12], has given an explanation of ECC suitability on high-security environment based on the underlying difficulty of ECDLP. It is explained that ECDLP is more difficult to solve than IFP and DLP [12] as currently known efficient algorithms to solve ECDLP are full exponential, whereas to solve IFP and DLP there are sub exponential algorithms. The article shown in [13], and the works discussed in [14] strengthened this idea. Moreover, in [13, 14], the suitability of ECC on smart cards is evidently explained, as ECC is more compact than RSA. Pietilainen [14] has compared ECC and RSA based on security, efficiency and space requirement by implementing both of them.

In [22], the authors provide basic alternatives to resolve the implementation issues of ECC on constrained devices like cellular phones. They indicated that curves over GF ($2^m$) are convenient for hardware implementation; whereas curves on GF ($p$) are suitable for software implementation. Finally, in [22], optimization of ECC based schemes is recommended as it is accepted as the next generation public-key cryptosystem.

Many of the works that aimed at improving performance of ECC based schemes either concentrated on improving the underlying mathematical operations, or concentrated on implementation of specific curves on a specific hardware platform. Little is done in designing different digital signature algorithms which may have a better performance than the existing ones.

In this work, after a thorough study of ECC based cryptosystems, areas of performance improvements of ECDSA has been examined. In ECDSA, the most expensive operation is the scalar multiplication or elliptic curve point multiplication. Another expensive operation is the modular inversion operation. Optimized techniques of scalar multiplication are given in [4, 16, 24]. Here, an attempt has been made to develop ECDSA related scheme in such a way that the number of elliptic curve point multiplications can be reduced during signature verification process.

## PRELIMINARIES

### Elliptic Curve Cryptography

Elliptic curves for cryptography are defined over finite algebraic structures such as finite fields. Let's assume prime fields $F_p$ of characteristics $p > 3$ [2, 4]. Such a curve is the set of geometric

solutions $P = (x, y)$ to an equation of the following form

$$E : y^2 = x^3 + ax + b(\text{mod } p) \qquad (1)$$

Where $a$ and $b$ are constants in $F_p$ ($p > 3$) satisfying $4a^3 + 27b^2 \neq 0(\text{mod } p)$. To have the points on $E$ to form a group, an extra point denoted by $O_\infty$ is included. This extra point is called the point at infinity and can be formulated as

$$O_\infty = (x, \infty) \qquad (2)$$

The point at infinity is the identity element for the group law formulated as

$$E = \{p = (x, y) \mid x, y \in F_p \text{ that solves (1)}\} \cup \{O_\infty\} \qquad (3)$$

This set of points form a group under a group operation which is conventionally written additively using the notation "+" [2]. The group forms an abelian group, [5], over which ECC is based and all operations are performed.

Suppose the point $P$ is in $E(F_p)$, and suppose $P$ has a prime order $n$, then, the cyclic additive subgroup of $E(F_p)$ generated by $P$ is

$$\langle P \rangle = \{O_\infty, P, 2P, 3P, ..., (n-1)P\}. \qquad (4)$$

The prime p, the equation of the elliptic curve E, and the point P and its order n, are the public domain parameters. Furthermore, a randomly selected integer d from the interval [1, n-1] forms a private key. Multiplying P by the private key d, which is called scalar multiplication, will generate the corresponding public key Q, i.e. Q = dP. The pair (Q, d) forms the ECC public-private key pair with Q is the public key and d is the private key.

**The Elliptic Curve Digital Signature Algorithm (ECDSA)**

ECDSA is the elliptic curve analogue of DSA [4, 16, 19]. It was accepted by many standard organizations around 2000. Below, the ECDSA signature generation and the ECDSA signature verification algorithms are given. The algorithms are available in [4].

In ECC, there are a set of domain parameters denoted by $D = (q, a, b, P, n, h)$. $q$ represents the field order of the prime field $F_q$. The parameters $a, b \in F_q$ are coefficients of the elliptic curve equation $E$. The parameter $P \in E(F_q)$ is the base point. The parameter $n$ is the order of the point $P$. $P$ is the generator of the cyclic sub group $\langle P \rangle$ (Eq. (4)). The parameter $h$ is known as cofactor. It is found as $h = order(E(F_q))/n$, Where o*rder* ($E(F_q)$) is the number of elements in $E(F_q)$.

Given the public-private key pair (Q, d), and domain parameters, the ECDSA signature generation and verification can be formulated as shown in Algorithms (1) and (2), respectively. A hash function, H shown in line 4 of Algorithm (1), accepts a variable size message M as input and produces a fixed-size output, referred to as a hash code H(M) or a message digest [2]. Hash functions are used for data integrity in conjunction with digital signature schemes, where a message is typically hashed first, and then the hash value as the representative of the message is signed in place of the original message. The receiver authenticates the message by applying the hash function on the message and re-computes the hash value.

Algorithm (1) ECDSA signature generation

Given parameters $q$, $a$ ,$b$, $P$, $n$ and private key $d$, to sign a message $m$, A does the following

1. Select $k \in [1, n-1]$.
2. Compute $kP = (x_1, y_1)$.
3. Compute $r = x_1 \bmod n$. If $r=0$ then go to step 1.
4. Compute $e = H(m)$.
5. Compute $s = k^{-1}(e + dr) \bmod n$. If $s=0$ go to step 1.
6. Return(r, s).

Algorithm (2) ECDSA signature verification

To verify A's signature ($r$, $s$) on $m$, B uses parameters $q$, $a$, $b$, $P$, $n$, $h$, public key $Q$, message $m$ and signature ($r$, $s$).

1. Verify that $r$ and $s$ are integers in the interval [1, $n$- 1]. If any verification fails then return ("Reject the signature").
2. Compute $e=H(m)$.
3. Compute $w= s^{-1} \bmod n$.
4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
5. Compute $X = u_1P + u_2Q$.

*a.* If $X = O_\infty$ then return ("Reject the signature");

6. Take the $x$-coordinate of $X$ as $x_1$ and compute $v = x_1 \bmod n$.
7. If $v = r$ then return ("Accept the signature"); Else return ("Reject the signature").

Below a proof is given to show how the signature verification of ECDSA works. If a signature (*r, s*) on a message *m* was generated by A, then necessarily the following will be true as a result of Algorithm (1), step number 5:

$$s \equiv k^{-1}(e + dr) (\mathrm{mod}\, n) \qquad (5)$$

From Eq. (5), by the principles of modular arithmetic, we will obtain that

$$k \equiv s^{-1}(e + dr) \equiv s^{-1}e + s^{-1}rd \ (\mathrm{mod}\ n) \qquad (6)$$

However, in algorithm (2), step number 3, $s^{-1} (\mathrm{mod}\, n)$ is represented by the parameter *w* as $w \equiv s^{-1} (\mathrm{mod}\ n)$. Substituting $s^{-1} (\mathrm{mod}\ n)$ in Eq.(6) by $w$, we will get

$$k \equiv we + wrd \ (\mathrm{mod}\ n) \qquad (7)$$

But, in Algorithm (2), step number 4, $we \ (\mathrm{mod}\ n)$ is represented by $u_1$ and $wr \ (\mathrm{mod}\ n)$ is represented by $u_2$. Thus, based on equation (7),

$$k = u_1 + u_2 d \ (\mathrm{mod}\ n) \qquad (8)$$

From the verification algorithm, we can see that

$$X = u_1 P + u_2 Q \qquad (9)$$

However, the public key $Q = dP$, where d is a private key in the interval $[1, n-1\}$ and *P* is the generator of the cyclic sub group $\langle P \rangle$(Eq. (4)). Therefore, substituting *Q* in Eq. (9) by *dP* and using Eq. (8), we will obtain,

$$X = u_1 P + u_2 dP = (u_1 + u_2 d)P = kP \qquad (10)$$

This proves that *v* = *r*. Because, $X = kP$ indicates that the *x*-coordinate of *kP*, $x_1$, in Algorithm (1) step numbers 2 and 3, and the *x*-coordinate of *X*, $x_1$, in Algorithm (2) step number 6, are equal in essence.

The ECDSA algorithm is involving modular inversion and the elliptic curve point multiplication operations (scalar multiplication) in the process of signature generation and signature verification. Both the modular inversion operation and scalar multiplication operation can have impact on the performance of the algorithm. In fact, the most time consuming operation in ECDSA is the elliptic curve scalar multiplication operation. This work focuses on a possible way of minimizing the scalar multiplication operations.

## PROPOSED SCHEME

Scalar multiplication dominates the execution time of ECC based schemes [4, 10]. In ECDSA, there are scalar multiplications in the signature generation and signature verification processes. In step 2 of algorithm (1) (ECDSA signature generation), the base point *P* is multiplied by the scalar or integer value *k*. Furthermore, in step 5 of algorithm (2) (ECDSA signature verification), the base point *P* is multiplied by an integer value $u_1$ and the public key *Q* is multiplied by an integer value $u_2$. As there are two scalar multiplications in the ECDSA verification algorithm, execution of the signature verification process needs a longer time than the signature generation process. So, attention is given to the verification process to examine if a scheme can be developed to minimize the execution time needed for signature verification of ECDSA.

Observing algorithms (1), and (2), there is an important relationship between the signature generation and the signature verification. The elliptic curve point $kP = (x_1, y_1)$ computed in the signature generation algorithm must be equal to the elliptic curve point $X = (x_1, y_1)$ computed during signature verification. Thus, if these points are equal, one can declare that the signature is valid and the signature is indeed generated by the owner of the public key *Q*. Therefore, finding any mathematical relationship without impairing the underlying ECDLP problem so that the points $kP = (x_1, y_1)$ and $X = (x_1, y_1)$ can be equal, leads us to a new scheme. Based on this notion, an attempt is made to search for such mathematical relationships and, accordingly, the following scheme is proposed.

Let the signature *s* is generated as

$$s = e + k + d (\mathrm{mod}\, n) \qquad (11)$$

Where *e* is the hash value *H*(*m*) of a given message *m*, *k* the per message secret, and *d* the private key. Hence, *k* can be computed as

$$k = s - e - d (\mathrm{mod}\, n) \qquad (12)$$

As the elliptic curve point $X = (x_1, y_1)$ must be equal to the elliptic curve point $kP = (x_1, y_1)$ (see algorithms (1), and (2)), in the verification process,

the point $X = (x_1, y_1)$ can be calculated based on the following steps.

1. Compute $e = H(m)$.
2. Compute $u = s - e \pmod{n}$. (i.e. $u = k + d \pmod{n}$).
3. Compute $X = uP - Q$.

To make it further clarified and to show that verification process holds, the following proof is given.

*Proof:*

If the signature $(r, s)$ is indeed generated by the private key $d$ holder using Eq. (11), then Eq. (12) holds true. In the verification process $X$ must be computed as

$$\begin{aligned} X &= uP - Q \\ &= uP - dP \\ &= (u - d)P \\ &= (k + d - d)P = kP \end{aligned}$$

And this proves that $X = kP$, from which it can be concluded that $v = r$ is as intended. The signature generation and signature verification algorithms are formulated as shown in algorithms (3) and (4), respectively.

Algorithm (3) Proposed scheme signature generation

Given domain parameters $q$, $a$, $b$, $P$, $n$ and private key $d$, to sign a message $m$, A does the following

1. Select $k \in [1, n-1]$.
2. Compute $kP = (x_1, y_1)$.
3. Compute $r = x_1 \bmod n$. If $r = 0$ then go to step 1.
4. Compute $e = H(m)$.
5. Compute $s = e + k + d \pmod{n}$. If $s = 0$ or $s = e$ then go to step 1.
6. Return$(r, s)$.

Algorithm (4) Proposed scheme signature verification

1. To verify A's signature $(r, s)$ on $m$, B uses domain parameters $q$, $a$, $b$, $P$, $n$, $h$, public key $Q$, message $m$ and signature $(r, s)$.
2. Verify that $r$ and $s$ are integers in the interval $[1, n-1]$. If any verification fails then return ("Reject the signature").
3. Compute $e = H(m)$.
4. Compute $u = (s - e) \pmod{n}$. If $u = 0$ return ("Reject the signature").
5. Compute $X = uP - Q$.
6. If $X = O_\infty$ then return ("Reject the signature");
7. Take the $x$-coordinate of $X$ as $x_1$ and compute $v = x_1 \bmod n$.
8. If $v = r$ then return ("Accept the signature"); Else return ("Reject the signature").

In this proposed scheme, the execution time required to verify a signature is reduced almost by half when it is compared with the execution time required to verify a signature in ECDSA. Both of the algorithms are compared for the underlying field size of 32-bit and 64–bit. Signature verification process of the proposed scheme was 48-57% faster than that of the ECDSA (see section 6). The reason is, in ECDSA's signature verification process, there are two elliptic curve point multiplications i.e. $u_1P$ and $u_2Q$ (algorithm (2)). The results of the point multiplications are to be added. So, there is one point addition operation. Whereas in the proposed one, there is only one scalar multiplication i.e. $uP$. Furthermore, there is one point addition operation (algorithm (4)). Thus, it would be reasonable and expected that the execution time for signature verification to be reduced almost by half.

The prominent issue here is security considerations. Basically, cryptographic schemes are designed to secure our on-line communication as well as stored information asset. So, is this proposed scheme as secure as ECDSA?

The security of ECDSA relies on the mathematical problem ECDLP. Currently known efficient algorithms to solve the ECDLP are fully exponential time algorithms, and hence, the problem is intractable. Similarly, this proposed scheme is relied on the ECDLP. However, the way signature is generated and verified in this scheme is different from that of the ECDSA. In ECDSA, the adversary is required to recover $d$ by brute search or by understanding the per message secret key $k$ or by using currently known efficient algorithms. If the adversary can get an opportunity to know the value of a single message secret key $k$, it is possible to recover $d$ from $k$. For this proposed scheme, if the adversary learns the per message secret $k$, $d$ can be recovered as

$$d = (s - k - e) \pmod{n} \qquad (13)$$

Without the knowledge of $k$, guessing $d$ and $k$ from the relationship $s = e + k + d \pmod{n}$ is difficult as there are different values of $d$ and $k$ in $[1, n-1]$ that can satisfy such relationship. In fact, there are

$(n-1)\times(n-1) = n^2 - 2n + 1$ possible solutions in the interval [1, *n* - 1] for the equation $s = e + k + d \pmod n$ for a given values of *s* and *e*. Such a result is very huge number. The complexity of this approach will be $O(n^2)$. So, by using this approach, it is not possible to guess the value of *k* or *d*. Rather than using this method, the straight forward attack – exhaustive search - is easier. That is, computing all the points $P, 2P, 3P....(n-1)P$ until the point *Q* is encountered.

If such a guess would have been possible, then adversaries could have been also successful in recovering *d* from the relationship $u_1 + u_2 d = k$ (see algorithm (2)) in ECDSA. Thus, from these arguments, it can be seen that this proposed scheme can be as secure as ECDSA. However, cryptographic schemes should normally pass through a lot of evaluations by different mathematicians and computer scientists before they get employed in real world applications in order that their security can be assured.

Currently, the most known efficient algorithm to attack ECDLP is the Pollard's rho algorithm. The main idea of Pollard's rho algorithm is to find distinct pairs (*a, b*) and (*a', b'*) of integers modulo *n* such that [4]

$$\begin{aligned} &aP + bQ = a'P + b'Q. \\ &(a-a')P = (b'-b)Q = (b'-b)dP. \end{aligned} \tag{14}$$

Here, the goal is to recover the private key *d*. From the Eq. (14), the value of *d* can be recovered as

$$\begin{aligned} &(a-a')P = (b'-b)dP \\ &d = (a-a')(b'-b)^{-1} \pmod n \end{aligned} \tag{15}$$

The method for finding the pairs (*a, b*) and (*a', b'*) is to select random integers $a, b \in [1, n-1]$ and store the triples $(a, b, aP + bQ)$ in a table until another point equal to $aP + bQ$ is obtained for the second time [4]. This occurrence is called *collision* [4]. By the birth day paradox, the expected number of iterations before a collision is obtained is approximately $\sqrt{\pi n/2} \approx 1.2533\sqrt{n}$.

In the proposed variant of ECDSA, there is one loophole so far discovered while designing the algorithm. If the adversary prepares his/her own message *m* and calculates the hash value *e* as $e = H(m)$, then by assigning *s* = *e* and $r = x_Q \pmod n$- where $x_Q$ is the *x* -coordinate of the public key *Q* and *n* is the order of the base point- the signature pair (*r, s*) will be a valid signature. In signature verification, the verifier will verify the signature as

$$u = s - e \pmod n = e - e \pmod n = 0$$

Then,

$$X = uP - Q = 0P - Q = -Q$$

However, -*Q* contains the coordinate pair $(x_Q, -y_Q)$, and hence, $v = x_Q \pmod n = r$ is as required. This is the reason for the check $s \neq e$ in algorithm (3) and the check $u \neq 0$ in algorithm (4).

## ALTERNATE FORM OF THE PROPOSED SCHEME

An alternate form of the proposed scheme can be achieved by including the parameter *r* while computing *s* as shown below:

$$s = er + k + d \pmod n \tag{16}$$

And *k* can be computed based on the following equation

$$k = s - er - d \pmod n \tag{17}$$

Therefore, in the verification process *u* (algorithm (4)) can be computed as shown below

$$u = s - er \pmod n. \tag{18}$$

In the verification process *X* can be calculated as

$$\begin{aligned} X = uP - Q &= (s-er) \pmod n P - dP \\ &= (er + k + d - er - d)P = kP \end{aligned} \tag{19}$$

## PERFORMANCE COMPARISON OF ECDSA AND THE PROPOSED SCHEME UPON PRACTICAL IMPLEMENTATION

To test the time taken to verify a signature or to generate a signature in ECDSA, and in the proposed scheme, three sample inputs are used for *k* and *d*. The impact of the message size on the execution time is negligible.

All the algorithms are executed in a Dell laptop. The laptop's processor is Intel Centrino with speed of 1.5GHz. It has 256MB RAM. Each of the algorithms has been run five times and then the time elapsed to execute the program at each run is registered.

The time taken to execute the ECDSA's and the proposed scheme's verification process is shown in Table 1. The corresponding average value is shown for each of the three sample inputs. Only the average values are given to save space.

As it can be observed from Table 1, the execution time difference between corresponding value for ECDSA and proposed scheme is very large. This is as a result of reduced elliptic curve point multiplications. Moreover, though the impact on its improvement is negligible as the underlying field size is increasing; two modular inversion operations available in ECDSA are eliminated in the proposed scheme.

Table 1: Average time taken to execute signature verification of ECDSA and the proposed scheme for each of the three sample inputs

| Algorithm | Average elapsed time to verify a signature (in seconds) – ECDSA vs. proposed scheme | | | | | |
|---|---|---|---|---|---|---|
| | For sample input 1 | | For sample input 2 | | For sample input 3 | |
| | 32-bit | 64-bit | 32-bit | 64-bit | 32-bit | 64-bit |
| ECDSA | 0.31 | 1.144 | 0.33 | 1.09 | 0.318 | 1.19 |
| Proposed Scheme | 0.165 | 0.55 | 0.16 | 0.53 | 0.17 | 0.39 |
| Difference in sec. | 0.145 | 0.594 | 0.17 | 0.56 | 0.148 | 0.80 |

For each sample input the improvement in percentage is calculated as

$$\frac{\text{Difference in Sec.}}{\text{ECDSA's Corresponding verif. time in sec}} \times 100\%. \quad (20)$$

The result is depicted in Table 2. For 32-bit field the overall average improvement is 48.28%. For the 64-bit field the overall average improvement is 56.93%. It can be seen that the proposed scheme's signature verification process can run faster than the ECDSA's signature verification by about 48-57%.

Table 2: Average improvement in percentage for signature verification process ECDSA

| Sample input number | Average improvement for verification process in percentage (%) | |
|---|---|---|
| | Signature verification | |
| | 32-bit | 64-bit |
| 1 | 46.77 | 51.92 |
| 2 | 51.52 | 51.64 |
| 3 | 46.54 | 67.23 |
| Sum | 144.83 | 170.79 |
| Average | 48.28 | 56.93 |

In general, it is observed that above 45% performance improvement can be achieved for signature verification process.

## CONCLUSION

A related new scheme is proposed and developed. This new scheme and ECDSA are implemented for comparison purposes. The signature verification algorithm of the newly proposed scheme is found to be above 45% faster than the verification process of ECDSA. The test was performed for randomly selected specific sizes of the private key $d$ and the one-time key $k$. The underlying field implementation was up to 192-bit size. Potentially, if further researches are conducted to examine its security strength, we believe that the result will play a great role in enhancing the speed of ECC based digital signature schemes.

## REFERENCES

[1] Menezes, A. Van Oorschot, P. Vanstone, S. "Handbook of Applied Cryptography", CRC Press, pp. 1-165, 223-481, 1997.

[2] Wenbo Mao, "Modern Cryptocraphy: Theory and Practice", Prentice Hall, pp. 139-199, 305-321, 2004.

[3] Stallings, W. "Cryptography and Network Security: Principles and Practice", Prentice Hall, pp. 24-27, 235-394, 2003.

[4] Hankerson, D. Menezes, A. Vanstone, S. "Guide to Elliptic Curve Cryptography", Springer, pp. 1-61, 75-147, 153-196, 2004.

[5] Shoup, V. "A Computational Introduction to Number Theory and Algebra (version 1)", Cambridge University Press, pp. 1-73, 180-281, 2005.

[6] Lewand, R. "Cryptological Mathematics", The Mathematical Association of America, pp. 1-26, 141-176, 2000.

[7] Diffie, W. Hellman, M. "New Directions in Cryptography", http:// crypto.csail.mit.edu/ classes/6.857/papers/diffie-hellman.pdf, 1976.

[8] Rivest, R. Shamir, A. Adleman, L. "A Method for Obtaining Digital Signatures and Public-key Cryptosystems", http://theory.lcs.mit.edu/~rivest/rsapaper.pdf, 1977.

[9] Bauer, F. "Decrypted Secrets: Methods and Maxims of Cryptology", Springer, pp. 8-43, 2000.

[10] Rosing, M. "Implementing Elliptic Curve Cryptography", Manning Publications, pp. 14-43, 104-120, 200-210, 1999.

[11] Rosner, M.C. "Elliptic Curve Cryptosystems on Reconfigurable Hardware", Master's Thesis, Worcester Polytechnic Institute, http://www.crypto.ruhr-uni-bochum.de/imperia/md/content/texte/theses/ms_mrosner.pdf, 1998.

[12] Johnson, D. B. "ECC, Future Resiliency and High Security Systems" http://www.comms.scitech.susx.ac.uk/fft/crypto ECCFut.pdf, 2000.

[13] "The Elliptic Curve Cryptosystem ", http://www.comms.scitech.susx.ac.uk/fft/crypto/EccWhite3.pdf, 2000.

[14] Pietilainen, H. "Elliptic Curve Cryptography on Smart Cards", Master's Thesis, Helsinki University of Technology, http://citeseer.ist.psu.edu/cache/papers/cs/25616/http:zSzzSzwww.iki.fizSznipsuzSzDippazSzdi.pdf/blake00elliptic.pdf, 2000.

[15] Schneier, B. "Applied Cryptography: Protocols, Algorithms, and Source Code in C", John Wiley & Sons, pp. 461-481, 1996.

[16] Turan, E. "ECDSA Optimization for ARM Processors for a NIST Curve Over GF($2^m$)", Master's Thesis, Oregon State University, http://security.ece.orst.edu/papers/01Turan.html, 2001.

[17] "Certicom ECC Challenge", http://www.certicom.com/download/aid-111/cert_ecc_challenge.pdf, 2006.

[18] Krasner, J. "Using Elliptic Curve Cryptography (ECC) for Enhanced Embedded Security " http://www.certicom.com/download/aid-355/WP-enhancedSecurity.pdf, 2004.

[19] Johnson, D. and et al, "The Elliptic Curve Digital Signature Algorithm (ECDSA)", http://www.certicom.com/download/aid-27/ECDSA.pdf, 2001.

[20] Vanstone, S. "Elliptic Curve Cryptography: The next generation of wireless security", http://www.certicom.com/download/aid-322/CIC_Markt& Technik_ECC.pdf, 2004.

[21] Tanik, H. K. "ECDSA Optimizations on an ARM Processor for a NIST Curve Over GF(p)", Master's Thesis , Oregon State University, http://islab.oregonstate.edu/papers/01Tanik.pdf, 2001.

[22] Balasubramaniam, P. et al ,"Implementation Issues in Elliptic Curve Based Cryptosystem", TRANSACTION ON CRYPTOLOGY, vol. 3, Issue 1, 2006.

[23] Eddy, W. et al, "An Interoperability Consideration in Selecting Domain Parameters for Elliptic Curve Cryptography", RS Information Systems, Inc., http://gltrs.grc.nasa.gov/reports/2005/CR-2005-213578.pdf, 2005.

[24] Joye, M. "Compact Encoding of Non-Adjacent Forms with Applications to Elliptic Curve Cryptography", Springer-Verlag, http://intro.gemplus.com/smart/rd/publications/pdf/JT01nafe.pdf, 2001.