A SURVEY ON ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM AND ITS VARIANTS

Greeshma Sarath[1] , Devesh C Jinwala[2] and Sankita Patel[3]

[1,2,3]Department of Computer Engineering, SVNIT, Surat
greeshmasarath88@gmail.com, dcjinwala@gmail.com,
sankitapatel@gmail.com

ABSTRACT

The Elliptic Curve Digital Signature Algorithm (ECDSA) is an elliptic curve variant of the Digital Signature Algorithm (DSA). It gives cryptographically strong digital signatures making use of Elliptic curve discrete logarithmic problem. It uses arithmetic with much smaller numbers 160/256 bits instead of 1024/2048 bits in RSA and DSA and provides the same level of security. The ECDSA was accepted in 1999 as an ANSI standard, and was accepted in 2000 as IEEE and NIST standards. It was also accepted in 1998 as an ISO standard. Many cryptologist have studied security aspects of ECDSA and proposed different variants. In this paper, we discuss a detailed analysis of the original ECDSA and all its available variants in terms of the security level and execution time of all the phases. To the best of our knowledge, this is a unique attempt to juxtapose and compare the ECDSA with all of its variants.

KEYWORDS

Network Protocols, Wireless Network, Mobile Network, Virus, Worms &Trojon

1. INTRODUCTION

ECC is one of the most advanced and promising techniques in the field of Public key cryptography. It offers many advantages over other cryptographic techniques which uses Integer factorization or discrete logarithmic approach. The hardest problem in which ECC is built upon is Elliptic Curve Discrete Logarithmic Problem (ECDLP). ECDLP is based on the infeasibility in computing discrete logarithms on elliptic curves over finite fields. It gives Elliptic curve cryptography a greater strength-per-key-bit. It uses arithmetic with much shorter numbers 160,256 bits instead of 1024,2048 bits and provides same level of security. Elliptic Curve Digital Signature Algorithm was first proposed in 1992 by Scott Vanstone in response to NIST's proposal of DSS [1][2]. It was later accepted in 1998 as an ISO standard (ISO 14888-3), as an ANSI standard (ANSI X9.62) in 1999, and as an IEEE standard (IEEE 1363-2000) and as a NIST standard (FIPS 186-2) in 2000.

However, it has disadvantages too. It is conceptually more difficult to understand and finding secure curves in set up phase is more difficult. ECDSA based on elliptic curve discrete logarithmic problem and is the most secure digital signatures scheme [4]. Many researches are developing different variants of ECDSA each having its own advantages and disadvantages and many cryptologist are trying to find weaknesses in ECDSA variants. This paper analyzes and describes different variants of ECDSA, their pros and cons and the attacks possible on each of the variants.

This section gives a brief introduction about the paper. Section 2, elaborates elliptic curve arithmetic operations and Elliptic curve discrete logarithmic problem. Section 3 gives a detailed description of original ECDSA scheme, its security proofs and an attack possible on original ECDSA scheme. Section 4 describes a variant of ECDSA suitable for signer with limited computation capability and Section 5 a variant suitable for a verifier with limited computation capability and its security proofs. Section 6 explains a two level digital signature scheme by using two different secrets. Section 7 describes Elliptic curve German digital signature scheme with inverse calculation in key generation phase. Section 8 describes a variant of ECDSA and a forging possible on it and section 9 details its improved version. Section 10 gives a brief description of two other variants to make ECDSA secure against adaptive chosen message attack and to avoid duplicate signatures. Section 11 elaborates Elliptic curve korean certificate based digital signature algorithm.. Section 12 discusses the implications, performance results and comparison of all ECDSA variants.

2. Elliptic Curve Arithmetic

Elliptic curve cryptography is based on the arithmetic of points on an elliptic curve[12][13]. Elliptic curves are represented by cubic equations similar to those used for calculating the circumference of an ellipse. An elliptic curve E over a field K is defined by a equation [3]:

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \ldots\ldots\ldots\ldots\ldots \quad (1)$$

Where $a_1,\ a_2,\ a_3,\ a_4,\ a_6 \in K\ and\ \Delta \neq 0$, where Δ is defined as follows:

$$\Delta = -d_2^2\, d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6;$$

$$where\ d_2 = a_1^2 + 4a_2\ , d_4 = 2a_4 + a_1a_3,$$
$$d_6 = a_3^2 + 4a_6\ and\ d_8 = a_1^2a_6 + 4a_2a_6a_1a_3a_4 + a_2a_3^2a_4^2$$

Set of all points (x, y) which satisfies the above equation along with ∞, a point at infinity, are the points on the elliptic curve.

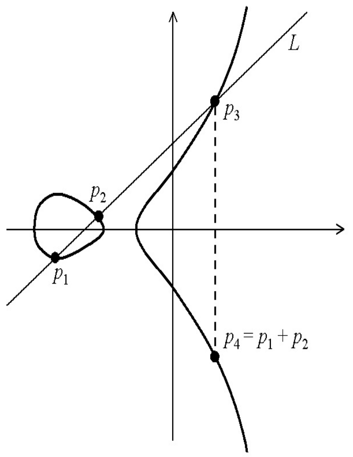

Figure 1. Point Addition

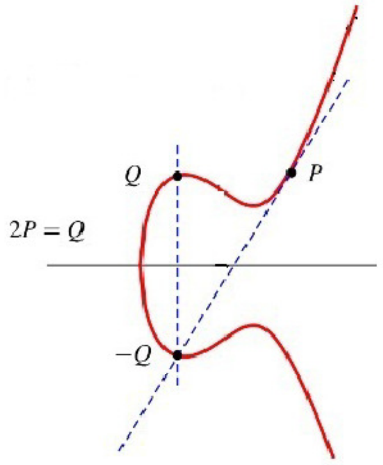

Figure 2. Point Doubling

The number of points on an elliptic curve, n, is the order of elliptic curve, (#(E(F_p)). The set of points of E (F_p) together with addition operation forms an abelian group with point at infinity, ∞ serving as the identity element. The Equation 1 is called weierstrass equation. The condition $\Delta \neq 0$ ensures that the elliptic curve is smooth, that is, there are no points at which the curve has two or more distinct tangent lines. If the field characteristic P is not equal to 2 or 3, that is prime field, and then the admissible change of Variables

$(x, y) \rightarrow \left(\frac{x-3a_1^2-12a_2}{36}, \frac{y-3a_1 x}{216} - \frac{a_1^3+4a_1a_2-12a_3}{36}\right)$ transform E to the curve,

$$y^2 = x^3 + \mathrm{ax} + \mathrm{b}; \text{where } \mathrm{a, b} \in \mathrm{K} \ldots\ldots\ldots\ldots\ldots\ldots. (2)$$

The $\Delta\ is\ 16(4a^3 + 27b^2)$.

2.1. Point Addition

Addition of points on an elliptic curve is defined by Chord and Tangent rule. Let P = (x_1, y_1) and Q = (x_2, y_2) be two distinct points on an elliptic curve E. Then the sum R, of P and Q, is defined as follows: Draw a line connecting P and Q extend it to intersect the elliptic curve at a third point. Then the sum, R is the negative of the third point. Negative of a point is defined by reflection of the point about the x-axis.

The double R, of P, is defined as follows: Draw the tangent line to the elliptic curve at P. Let it intersects the elliptic curve at a second point. Then the double R is the reflection of this point about the x-axis.

2.2. Point Multiplication

Point Multiplication (Scalar multiplication) is the arithmetic operation which computes kp where k is an integer and p is a point on elliptic curve. It is done by repeated addition. For example Q = kp means Q is obtained by adding p k times to itself (p + p + p....k times). Cryptanalysis involves determining k given P and Q. This operation dominates the execution time of elliptic curve cryptographic schemes.

2.3. Operations defined for E(F_p): $y^2 = x^3$+ax +b

(1) Identity: P + ∞ = ∞ + P = P for all $P \in$ E(F_p)

(2) Negatives: If P = (x,y) ∈ E(F_P), then (x,y) + (x, -y) = ∞. The point (x, -y) is denoted by –P and is called negative of P. Note that P indeed is a point in E(F_P).

(3) Point Addition: Let P=(x_1,y_1) ∈ E(K) and Q=(x_2,y_2) ∈ E(K) ; where P≠ ±Q, then P+Q = (x_3,y_3)

where,

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \quad \text{and} \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 (x_1 - x_3) - y_1$$

(3) Point Doubling: Let P=(x_1,y_1) ∈ E(K), then 2P = (x_3,y_3) where,

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \quad \text{and} \quad y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 (x_1 - x_3) - y_1$$

2.4. Elliptic Curve Discrete logarithm problem:

The elliptic curve discrete logarithm problem (ECDLP) is defined as follows: Given the elliptic curve domain parameters and a point $P \in E(F_p)$, find the unique integer k, $0 \leq k \leq n1$, such that P=kG, where n1 is order of E.

ECDLP is similar to the Discrete Logarithm Problem and is the elliptic curve analogue of DLP. In the ECDLP, the subgroup Z_p* is replaced by the group of points on an elliptic curve over a finite field. In addition, unlike the DLP and the integer factorization problem, no sub exponential-time algorithm is known for the ECDLP. ECDLP is considered to be significantly harder than DLP, thus giving elliptic curve cryptosystems a greater strength-per-key-bit than their discrete logarithmic counterparts.

3. ECDSA

Elliptic Curve Digital Signature Algorithm [6] was first proposed in 1992 by Scott Vanstone in response to NISTs proposal of DSS. It was later accepted in 1998 as an ISO standard (ISO 14888-3), as an ANSI standard (ANSI X9.62) in 1999, and as an IEEE standard (IEEE 1363-2000) and as a NIST standard (FIPS 186-2) in 2000.

Elliptic curve digital signature algorithm consists of 3 phases: 1. Key generation, 2. Signature generation, 3.Signature verification. A setup phase has to execute before the key generation phase to generate the domain parameters. Domain parameters for an elliptic curve describe an elliptic curve E defined over a finite field Fp, a base point $g \in E$ (Fp) (generator) with order n. The parameters should be chosen carefully so that ECDLP is resistant to all known attacks. The elliptic curve is chosen by choosing $(a,b) \in (1, P)$ and substituting in equation. So the domain parameters can be defined as p, E (a, b), g, n.

3.1. Key pair generation using ECDSA:

Let A be the signatory for a message M. Entity A performs the following steps to generate a public and private key:

(1) Select a unique and unpredictable integer, d, in the interval [1, n-1]
(2) Compute Q = dg
(3) Sender A's private key is d
(4) Sender A's public key is the combination (E, g, n, Q)

3.2. Signature Generation Using ECDSA

Using A's private key, A generates the signature for message M using the following steps:

(1) Select a unique and unpredictable integer k in the interval [1,n-1]
(2) Compute kg = (x_1,y_1), where x_1 is an integer
(3) Compute r = x_1 mod n; If r = 0, then go to step 1
(4) Compute h = H(M), where H is the SHA-512[10]
(5) Compute $s = k^{-1}(h + dr) \bmod n$; If s = 0, then go to step1
(6) The signature of A for message M is the integer pair (r, s)

3.3. Signature Verification Using ECDSA

The receiver B can verify the authenticity of A's signature (r, s) for message M by performing the following:

(1) Obtain signatory A's public key (E, q, n, Q)
(2) Verify that values r and s are in the interval [1,n-1]
(3) Compute w = s^{-1} mod n.
(4) Compute h = H(M), where H is the same secure hash algorithm used by A.
(5) Compute u1 = hw mod n
(6) Compute u2 = rw mod n
(7) Compute u1g + u2Q = (x_0,y_0)
(8) Compute v = x_0 mod n
(9) The signature for message M is verified only if v = r

3.4. Security of ECDSA

Public key is generated by computing the point Q, where Q = dg. In order to crack the elliptic curve key, adversary Eve would have to discover the secret key d when Q and g are provided. The order of the Elliptic curve, E is a prime number n, then computing d given dg and g would take roughly 2n=2 operations [7] . For example, if the key length n is 192 bits (the smallest key size that NIST recommends for curves defined over GF(p)), then Eve will be required to compute about 296 operations. If Eve had a super computer and could perform one billion operations per second, it would take her around two and a half trillion years to find the secret key. This is the elliptic curve discrete logarithm problem behind ECDSA. The curve parameter should be chosen so carefully to secure Elliptic curve from well known attacks like Pollard's rho[1] and Pohlig-Hellman.

3.5. Proof of ECDSA signature Scheme

Signature send by A to B is (r, s) and s can be generated only by A because only A knows its private key d. s = k^{-1}(h + dr) mod n on rearranging

- K = s^{-1} (h + dr)
- Kg = s^{-1} (h + dr)g
- Kg = s^{-1}hg + s^{-1}drg
- r = hwg + rwdg
- r = u_1g + u_2Q

3.6. A Possible Attack on ECDSA

The secret k used for signing two or more messages should be generated independent of each other. In particular, a different secret k should be used for signing different messages otherwise the private key d can be recovered. However if a secure random or pseudo-random number generator is used, then the chance of generating a repeated k value is negligible. If same secret k is used to generate signature of two different messages m1 and m2 then it will result in two signatures (r,s1) and (r, s2).

- s1 = k^{-1}(h1 + dr)
- s2 = k^{-1}(h2 + dr) ; where h1 = SHA512 (m1) and h2 = SHA512 (m2).
- ks1- ks2 = h1+dr-h2-dr

- $k = (h1-h2)/(s1-s2)$
- $d = (ks-h)/r$

4. VARIANT 1

The scheme [5] is suitable for a signer who has limited computing capability like, a signer using his Smart Card which stores his secret key and signs a message on a terminal.

Key pair phase of this scheme is same as the ECDSA scheme.

4.1. Signature Generation

(1) Select a unique and unpredictable integer k in the interval [1,n-1]
(2) $kg \leftarrow (x_1,y_1)$, where x_1 is an integer.
(3) $r \leftarrow x_1 \bmod n$; If r = 0, then go to step 1
(4) $h \leftarrow H(M)$, where H is the SHA-512
(5) $s \leftarrow d^{-1}(rk - h) \bmod n$; If s = 0, then go to step1
(6) The signature of A for message M is the integer pair (r,s)

Here the advantage is that there is no need of calculating inverse of d in each individual signing operation. d is the private key of the signer which will remain stable for a period of time, it can be precomputed and stored in the key generation phase itself.

4.2. Signature Verification

The receiver B can verify the authenticity of A's signature (r, s) for message M by performing the following:

(1) Obtain signatory A's public key (E, q, n, Q)
(2) Verify that values r and s are in the interval [1,n-1]
(3) $w \leftarrow r^{-1} \bmod n$
(4) $h \leftarrow H(M)$, where H is the same secure hash algorithm used by A
(5) $u1 \leftarrow hw \bmod n$
(6) $u2 \leftarrow sw \bmod n$
(7) $(x0,y0) \leftarrow u1g + u2Q$
(8) $v \leftarrow x0 \bmod n$
(9) The signature for message M is verified only if v = r

4.3. Proof of the Scheme

Signature send by A to B is (r, s) and s can be generated only by A because only A knows its private key d. $s = d^{-1}(rk - e) \bmod n$ on rearranging,

- $sd = (rk - e)$
- $dsr^{-1} = r^{-1}(rk - e)$
- $dsgr^{-1} = gk - egr^{-1}$
- $kg = egw + Qws$
- $r = u1g + u2Q$.

Attack on same k can be implemented successfully on this method as well.

5. VARIANT 2

The scheme [5] is suitable for the verifier who has limited compute apparatus. That is in this scheme the complexity of verification operation is less compared to that of the above schemes. Key pair generation phase of this scheme is same as the ECDSA scheme.

5.1. Signature Generation

(1) Select a unique and unpredictable integer k in the interval [1,n-1]
(2) kg $\leftarrow (x_1,y_1)$, where x_1 is an integer
(3) r $\leftarrow$ x_1 mod n; If r = 0, then go to step 1
(4) h $\leftarrow$ H(M), where H is the SHA-512
(5) s $\leftarrow k(h + rd)^{-1}$ mod n; If s = 0, then go to step1
(6) The signature of A for message M is the integer pair (r,s).

5.2. Signature Verification

The receiver B can verify the authenticity of A's signature (r, s) for message M by performing the following:

(1) Obtain signatory A's public key (E, q, n, Q)
(2) Verify that values r and s are in the interval [1,n-1]
(3) h $\leftarrow$ H(M), where H is the same secure hash algorithm used by A
(4) u1 $\leftarrow$ hs mod n
(5) u2 $\leftarrow$ rs mod n
(6) $(x_0,y_0) \leftarrow$ u1g + u2Q
(7) v $\leftarrow x_0$ mod n
(8) The signature for message M is verified only if v = r

In this scheme, k^{-1} is no longer be calculated, but we must calculate $(h + rd)^{-1}$ in the signing phase. But there is no need of calculating inverse in verification phase which is one of the most costlier operation in modular arithmetic. So the complexity of the verification operation is less in this scheme. Since r and k are functions, pre-calculating couldn't be used to reduce the operation amount. If an attacker want to forge a signature, he must decide a pair of (r, s) too, which must fit for the equation R = (xR ,yR) = u1g+ u2Q = hsg+ rsQ. So he encounters the same difficulty as attacking original algorithm.

5.3. Proof of the Scheme

Signature send by A to B is (r, s) and s can be generated only by A because only A knows its private key d. s $\leftarrow k(h + rd)^{-1}$ mod n on rearranging,

- s(h + rd) = k.
- s(h + rd)g = kg
- shg + rdsg = kg
- u1g + u2dg = kg
- u1g + u2Q = kg.

Attack on same k can be implemented successfully on this method as well.

6. VARIANT 3

In this scheme [6], [7] two levels of digital signature are implemented by using two secrets k1 and k2. Here d cannot be determined even if the same secret (k1, k2) is repeated. The processes are more complex than original ECDSA scheme and it increases the security level. Key pair generation phase of this scheme is same as the ECDSA scheme.

6.1. Signature Generation

(1) SELECT Select two integers k1 and k2 such that $1 \leq k1,k2 \leq n-1$
(2) $k1g \leftarrow (x_1, y_1)$, $k2g \leftarrow (x_2,y_2)$ where x_1,x_2,y_1,y_2 are integers
(3) $r1 \leftarrow x1 \bmod n$; $r2 \leftarrow x2 \bmod n$ If $r1,r2 = 0$, then go to step 1
(4) $h \leftarrow H(M)$, where H is the SHA-512
(5) $s \leftarrow k1^{-1}(hk2 + d(r1 + r2)) \bmod n$; If $s = 0$, then go to step1
(6) The signature of A for message M is the integer pair (r1, r2, s)

6.2. Signature Verification

(1) Obtain signatory's public key (E, q, n, Q)
(2) Verify that values r1,r2 and s are in the interval [1,n-1]
(3) $w \leftarrow s^{-1} \bmod n$
(4) $h \leftarrow H(M)$, where H is the same secure hash algorithm used by A
(5) $u1 \leftarrow hwk2 \bmod n$
(6) $u2 \leftarrow (r1+r2)\, w \bmod n$
(7) $(x0,y0) \leftarrow u1g + u2Q$
(8) $v \leftarrow x0 \bmod n$
(9) The signature for message M is verified only if $v = r1$.

6.3. Proof of the Scheme

Signature send by A to B is (r, s) and s can be generated only by A because only A knows its private key d. $s = k1^{-1}(hk2 + d(r1 + r2)) \bmod n$ on rearranging,

- $k1 = s^{-1}(hk2 + d(r1 + r2))$.
- $k1g = s^{-1}(hk2 + d(r1 + r2))g$.
- $k1g = s^{-1}hk2g + s^{-1}d(r1 + r2)g$.
- $r = hwk2g + (r1 + r2)wdg$.
- $r = u1g + u2Q$

If same secret (k1,k2) is used for signing two different messages, It will generate two different signatures (r1,s1) and (r1,s2)

- $s1 = k1^{-1}(h1k2 + d(r1 + r2))$
- $s2 = k1^{-1}(h2k2 + d(r1 + r2))$ Where $h1 = SHA512(m1)$ and $h2 = SHA512(m2)$
- $k1s1 - k1s2 = h1k2 + dr - h2k2 - dr$
- $k1(s1 - s2) = k2(h1 - h2)$

We cannot obtain k1, k2 from this equation and so this scheme is more secure than original ECDSA scheme.

7. VARIANT 4

This scheme is also called Elliptic Curve German Digital Signature Algorithm[8]. One of the disadvantages of ECDSA scheme is the calculation of inverse in signing phase. Calculation of inverse is one of the expensive operations in Modular Arithmetic, so avoiding it will reduce the cost and time. In ECGDSA inverse calculation is done in the key pair generation phase and not in Signing phase. A key will remain constant for a stable amount of time so signing is done more frequently than key generation. ECGDSA will save time and cost than ECDSA.

7.1. Key pair generation using ECGDSA

Let A be the signatory for a message M. Entity A performs the following steps to generate a public and private key

(1) Select a unique and unpredictable integer, d, in the interval [1,n-1]
(2) $Q \leftarrow (d^{-1} \bmod n)g$
(3) Sender A's private key is d
(4) Sender A's public key is the combination (E, g, n, Q)

7.2. Signature Generation using ECGDSA

Using A's private key, A generates the signature for message M using the following steps:

(1) Select a unique and unpredictable integer k in the interval [1,n-1]
(2) $kg \leftarrow (x_1,y_1)$, where x_1 is an integer
(3) $r \leftarrow x_1 \bmod n$; If $r = 0$, then go to step 1
(4) $h \leftarrow H(M)$, where H is the SHA-512
(5) $s \leftarrow (kr-h)\, d \bmod n$; If $s = 0$, then go to step1
(6) The signature of A for message M is the integer pair (r,s)

7.3. Signature Verification using ECGDSA

The receiver B can verify the authenticity of A's signature (r, s) for message M by performing the following:

(1) Obtain signatory A's public key (E, q, n, Q)
(2) Verify that values r and s are in the interval [1,n-1]
(3) $w \leftarrow r^{-1} \bmod n$
(4) $h \leftarrow H(M)$, where H is the same secure hash algorithm used by A
(5) $u1 \leftarrow hw \bmod n$
(6) $u2 \leftarrow sw \bmod n$
(7) $(x0,y0) \leftarrow u1g + u2Q$
(8) $v \leftarrow x0 \bmod n$
(9) The signature for message M is verified only if $v = r$

7.4. Proof of the Scheme

Signature send by A to B is (r, s) and s can be generated only by A because only A knows its private key d. s = (kr-h) d mod n.

- s = (kr-h) d
- $s * r^{-1}d^{-1} = (kh * r^{-1})$
- $sw*d^{-1}g = kg - hw* g$
- kg = hw*g + sw*Q
- r = u1g+u2Q

If same secret (k) is used for signing two different messages, It will generate two different signatures (r,s1) and (r,s2).

- s1 = (kr-h1) d
- s2 = (kr-h2) d, Where h1 = SHA512 (m1) and h2 = SHA512 (m2)
- s1-s2= h2-h1

K cannot be determined even though same secret is used to sign two different messages. So this scheme is not vulnerable to attack on same secret.

8. VARIANT 5

In ECGDSA (Variant 4) there is no need of finding inverse in signing phase but there is a need in key generation phase. In this variant[9] there is no need in finding inverse in both key generation and signing phase. This scheme embeds the information of signature into a point on the ellipse.

8.1. Key pair generation

Let A be the signatory for a message M. Entity A performs the following steps to generate a public and private key:

(1) Select a unique and unpredictable integer, d, in the interval [1,n-1]
(2) Q $\leftarrow$ (dg mod n)
(3) Sender A's private key is d
(4) Sender A's public key is the combination (E, g, n, Q)

8.2. Signature Generation

Using As private key, A generates the signature for message M using the following steps:

(1) Select a unique and unpredictable integer k in the interval [1,n-1]
(2) kg $\leftarrow$ (x_1,y_1), where x_1 is an integer
(3) r $\leftarrow$ x_1 mod n; If r = 0, then go to step 1
(4) h $\leftarrow$ H(M), where H is the SHA-512
(5) s $\leftarrow$ (kh + (r xor h)d)g mod n
(6) If s = 0, then go to step1
(7) The signature of A for message M is the pair (r, s)

8.3. Signature Verification

The receiver B can verify the authenticity of A’s signature (r, s) for message M by performing the following:

(1) Obtain signatory A’s public key (E, q, n, Q)
(2) Verify that values r and s are in the interval [1,n-1]
(3) h $\leftarrow$ H(M), where H is the same secure hash algorithm used by A
(4) w $\leftarrow$ h^{-1} mod n
(5) u $\leftarrow$ (r xor h) mod n
(6) (x0, y0) $\leftarrow$ w(s - uQ)
(7) v $\leftarrow$ x0 mod n
(8) The signature for message M is verified only if v = r

8.4. Proof of the scheme

Signature send by A to B is (r, s) and s can be generated only by because only A knows its private key d. s = (kh + (r xor h)d) g mod n

- s = (kh + (r xor h)d)g modn
- s = (kh +ud) g
- sw = kg +uwQ
- kg = sw - uwQ
- r = w (s-uQ)

8.5. A Forging possible on variant 5

An attacker T can forge the signature with the knowledge of public parameters (E, g, n, Q)

(1) Select an integer k in the interval [1,n-1]
(2) kg = (x1,y1), where x1 is an integer
(3) r = x1 mod n; If r = 0, then go to step 1
(4) h = H(M), where H is the SHA-512
(5) s = (khg + (r xor h)Q) mod n
(6) The Forged signature of A for message M is the pair (r, s)

On receipt on signature B will verify the signature as normal verification procedure of variant 4

V = w(s - uQ)
= w(s(r xor h)Q)
= w((khg + (r xor h)Q)(r xor h)Q)
= w(khg)
= kg

Then the forged signature validated, the attacker can successfully attack. Therefore this digital signature scheme is not secure. Anyone can use legitimate user’s public-key to forge the signature of any information.

9. AN IMPROVED VARIANT 5

Variant 5 can be made secured by adding one more step in both the signing and verification phase[9].

9.1. Signature Generation

Using A's private key, A generates the signature for message M using the following steps

(1) Select a unique and unpredictable integer k in the interval [1,n-1]
(2) kg ← (x1,y1), where x1 is an integer
(3) r ← x1 mod n; If r = 0, then go to step 1
(4) h ← H(M), where H is the SHA-512
(5) s1 ← (kh + (r xor h)d) g mod n; If s1 = 0, then go to step1
(6) s2 ← s1d; If s2 = 0, then go to step1
(7) The signature of A for message M is the integer pair (r,s)

9.2. Signature Verification

The receiver B can verify the authenticity of A's signature (r, s) for message M by performing the following:

(1) Obtain signatory A's public key (E, q, n, Q)
(2) Verify that values r and s are in the interval [1,n-1]
(3) Verify the equation s2g = s1Q If the equation was established continue with the verification process else refuse the signature
(4) h ← H(M), where H is the same secure hash algorithm used by A
(5) w ← h^{-1}modn
(6) u ← (r xor h) mod n
(7) (x0, y0) ← w(s - uQ)
(8) v ← x0 mod n
(9) The signature for message M is verified only if v = r

In this scheme after generating the signature s1, which is a point on the ellipse, it is encrypted with the private key of the signer. Firstly verifier verifies the encrypted result s2 before verifying the signature. As d, private key is known only to the signer and the encryption scheme s2 = s1d is guaranteed by elliptic curve discrete logarithm problem the improved scheme is secure. Attacker attempting to forge the signature by replacing the message cannot encrypt the signature so the improved scheme can prevent forgery. The signature generation and validation phase involves more elliptic curve point multiplication operation and hence is more complex and will take more time and cost. But as the complexity is increasing security level provided by the algorithm also increases.

10. OTHER VARIANTS

In order to make ECDSA secure against existential forgery by adaptive chosen message attack authors of [11] proposed a new variant of ECDSA named as ECDSAII. In ECDSAII instead of calculating the hash of the message they are calculating hash of, the message appended with r wher er = X-Coordinate of kg(mod n). Even if same message is signing hash generated will differ with high probability since k is randomly generated number.

Authors of the same paper improved ECDSAII so that it avoids the notion of duplicate signatures. They name this algorithm as ECDSAIII The alteration is to replace r=X-Coordinate of kg(mod n) with r = X-Coordinate of kg(mod n) + Y-Coordinate of kg(mod n). Since it is hard to find two elliptic curve points Q1 = (x1, y1) and Q2 = (x2, y2) such that one knows the respective discrete logarithms Qi =kiP and such that x1 + y1 = x2 + y2. It avoids duplicate signatures to an extent. This can only happen when the line L(t) : X + Y = t for some constant t is geometrically related to the group law linking Q1 and Q2. If L(t) is a tangent at Q1 and we know the discrete logarithm k1 then we know that L(t) intersects the curve in one other point, say Q2 = k2P, of the required form and that k2 = (2k1) (modn). Hence we need to avoid points where L(t) is a tangent. But for all possible values of t the line L(t) is only a tangent for atmost four points on any given elliptic curve.

11. ECKCDSA

A group of Korean cryptographers, in association with government-supported agencies, had developed a candidate algorithm for Korean digital signature standard, which is named KCDSA, Korean Certificate Based Digital Signature Algorithm, is a signature algorithm in which the public key is validated by means of a certificate issued by some trusted authority. The X.509-based certificate may be used for this purpose. In this case, the Cert Data can be simply the formatted certification data defined by X.509.An elliptic curve variant of KCDSA is EC-KCDSA Elliptic curve Korean Certificate Based Digital Signature Algorithm. The algorithm uses the public key PA := [d1 mod n]g and z is a hash-value of Cert Data. Cert Data denotes the signer's certification data, which should contain at least signer's distinguished identifier, public key Q and the domain parameters.

11.1. Signature Generation

(1) Select a unique and unpredictable integer k in the interval [1,n-1]
(2) kg $\leftarrow$ (x1,y1), where x1 is an integer
(3) r $\leftarrow$ x1 mod n; If r = 0, then go to step 1
(4) h $\leftarrow$ H(z || M), where H is the SHA-512
(5) s $\leftarrow$ d(k - r xor h) mod n; If s = 0, then go to step1
(6) The signature of A for message M is the integer pair (r,s)

Here the advantage is that there is no need of calculating inverse of d in each individual signing operation. d is the private key of the signer which will remain stable for a period of time, it can be precomputed and stored in the key generation phase itself.

11.2. Signature Verification

The receiver B can verify the authenticity of A's signature (r, s) for message M by performing the following:

(1) Obtain signatory A's public key (E, q, n, Q)
(2) check the validity of the signer's certificate, extracts the signer's certification data Cert Data from the certificate and computes the hash value z = h(Cert Data)
(3) Verify that values r and s are in the interval [1,n-1]
(4) h $\leftarrow$ H(z || M), where H is the same secure hash algorithm used by A
(5) w $\leftarrow$ r^{-1}modn
(6) u1 $\leftarrow$ r xor h mod n
(7) u2 $\leftarrow$ s

(8) $(x0,y0) \leftarrow u1g + u2Q$
(9) $v \leftarrow x0 \bmod n$
(10) The signature for message M is verified only if v = r

11.3. Proof of the Scheme

Signature send by A to B is (r, s) and s can be generated only by A because only A knows its private key d. u1g + u2Q on rearranging,

- r xor hg + sQ
- r xor hg +d(k – r xor h)d^{-1}g
- r xor hg +(k – r xor h)g
- kg

12. Comparison Among The Variants

Table 1. Comparison of ECDSA Variants

Algorithm	Signature Generation	Verification	No. of Secrets	Attack on same secret	Inverse in Signing	Inverse in Key Generation	Inverse in Verification
ECDSA	$k^{-1}(e+dr)$	$u1 = es^{-1}$ $u2 = rs^{-1}1$ $u1g + u2Q$	1	Vulnerable	Yes	No	Yes
Variant 1	$d^{-1}(rk-h)$	$u1 = r^{-1}e$ $u2 = r^{-1}s$ $u1g + u2Q$	1	Vulnerable	No	Yes	Yes
Variant 2	$k(h + rd)^{-1}$	u1 = hs u2 = rs u1g+u2Q	1	Vulnerable	Yes	No	No
Variant 3	$k1^{-1}(ek2+d(r1 +r2))$	$u1 = r^{-1}e$ $u2 = r^{-1}s$ u1g+u2Q	2	Not Vulnerable	Yes	No	Yes
Variant 4	s=(kr-e)d	$u1 = r^{-1}e$ $u2 = r^{-1}s$ u1g+u2Q	1	Not Vulnerable	No	Yes	Yes
Variant 5	s1 = (ke + (r xor e)d)g s2 = s1d	s2g=s1Q u = r xor e; $e^{-1}(s -uQ)$	1	Not Vulnerable	No	No	Yes
ECKCDSA	d(k – r xor h	u1 = r xor h u2=su1g+ u2Q	1	Not Vulnerable	No	Yes	No

Comparing all the variants Original ECDSA is vulnerable to attack on same secret. Variant 1 is suitable for signer with limited compute apparatus and variant 2 for a verifier with limited compute apparatus. Variant 3 requires the use of 2 variables and time taken for signature generation and verification is also more. In variant 4 it requires no inverse calculation in signing phase and time taken is less for signing phase but it takes more time in key generation phase as it needs to calculate inverse in key generation phase. In case of variant 5 there is no inverse calculation in both key generation and signing phase but it is more complex and it takes more time as it needs more point multiplication operation. As the complexity increases the security level also increases. Table I shows the comparison of operations and possible attacks on all the variants.

Time taken for key generation signature generation and verification of all the variants are measured and detailed in Table II. Here the key size used is 192 bits and the processor used for implementation is Pentium(R) Dual-Core CPU 2.30 GHz and platform used is java.

Table 2. Time taken for ECDSA Variants in Milliseconds

Algorithm	**Key Generation**	**Signature Generation**	**Verification**
ECDSA	78	93	125
Variant 1	82	78	125
Variant 2	78	98	120
Variant 3	78	153	131
Variant 4	83	78	125
Variant 5	78	141	218

13. CONCLUSION

Performance and Security of ECDSA and its variants is compared and listed. Algorithm to be used can be determined according to the application and compute apparatus available for the application. Improved variant 5 uses more elliptic curve operations and the time taken in each of the phases is large compared to the other schemes. For applications which need more security and is having enough resources improved variant 5 can be used as the signature scheme. Variant 1 and variant 4 can be used for applications with signer having limited resources. Among them variant 4 is more efficient as it does not need pre computing and storage of an extra value d^{-1}.Variant 3 is resistant to attack on same k by usage of two secrets k1 and k2 but variant 4 is also vulnerable to same attack even without using a second secret.

ACKNOWLEDGEMENTS

This work was supported in part by Space Applications Center, ISRO, Ahmedabad and in part by the Research Grant from the Department of Electronics and Information Technology, Ministry of Communications and Information Technology, Govt of India, New Delhi. The First author would like to thank Mr Deval Mehta , Head SCTD and Ms Bhanu Panjwani Sci/Engr-SC ,SAC, ISRO for their valid suggestions and support throughout the work.

REFERENCES

[1] Darrel Hankerson, Alfred Menezes, Scott VanstoneH, ”Guide to Elliptic Curve Cryptography”, Springer 2004
[2] T.Elgamal, ”A public key Cryptosystem and a signature scheme based on discrete logarithms”, IEEE transactions on information theory, 1985.
[3] Aqeel Khalique ,Kuldip Singh Sandeep Soodv,”Implementation of Elliptic Curve Digital Signature Algorithm”, International Journal of Computer Applications 2010.
[4] Don B. Johnson,Alfred J. Menezes,Elliptic Curve DSA (ECDSA): An Enhanced DSA
[5] Hu Junru , ”The Improved Elliptic Curve Digital Signature Algorithm”, 2011 IEEE.
[6] M.Prabu,R.Shanmugalakshmi, ”A comparative Analysis of signature schemes in a new approach to variant on ECDSA ”, 2009 IEEE.
[7] Hung-Zih Liao, Yuan-Yuan Shen, ”On the Elliptic Curve Digital Signature Algorithm”Tunghai Science Vol. 8: 109126 July, 2006

[8] “Technical Guideline TR-0311 Elliptic Curve Cryptography Version 2.0”, Bundesamt fur Sicherheit in der Informationstechnik 2012.

[9] Qiuxia Zhang , Zhan Li , Chao Song , ”The Improvement of digital signature algorithm Based on elliptic curve cryptography”, 2011 IEEE.

[10] William Stallings ”Cryptography and Network Security”, fourth edition

[11] John Malone-Lee , Nigel P. Smart, ”Modifications of ECDSA”, Springer- Verlag Berlin Heidelberg 2003.

[12] Victor S. Miller ”Use of Elliptic Curves in Cryptography”, Springer- Verlag Berlin Heidelberg 1986.

[13] N. Koblitz,” Elliptic curve cryptosystems”, Mathematics of Computation 48, 1987, pp. 203-209