Implementation of Elliptic Curve Digital Signature Algorithm

Aqeel Khalique Kuldip Singh Sandeep Sood

Department of Electronics & Computer Engineering,
Indian Institute of Technology Roorkee
Roorkee, India

ABSTRACT
The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the Digital Signature Algorithm (DSA). It 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, and is under consideration for inclusion in some other ISO standards. Unlike the ordinary discrete logarithm problem and the integer factorization problem, no sub exponential-time algorithm is known for the elliptic curve discrete logarithm problem. For this reason, the strength-per-key-bit is substantially greater in an algorithm that uses elliptic curves. This paper describes the implementation of ANSI X9.62 ECDSA over elliptic curve P-192, and discusses related security issues.

Categories and Subject Descriptors
D.4.6 [**Operating Systems**]: Security and Protection –*access controls*, a*uthentication cryptographic control*; E.3 [**Data**]: Data Encryption –*Public key cryptosystem, standards*.

General Terms
Algorithms, Security.

Keywords
integer factorization, discrete logarithm problem, elliptic curve cryptography, DSA, ECDSA.

1. INTRODUCTION

Cryptography is the branch of cryptology dealing with the design of algorithms for encryption and decryption, intended to ensure the secrecy and/or authenticity of message. The DSA was proposed in August 1991 by the U.S. National Institute of Standards and Technology (NIST) and was specified in a U.S. Government Federal Information Processing Standard (FIPS 186) called the Digital Signature Standard (DSS). Its security is based on the computational intractability of the discrete logarithm problem (DLP) in prime-order subgroups of Z_p^*. Digital signature schemes are designed to provide the digital counterpart to handwritten signatures (and more). Ideally, a digital signature scheme should be existentially non-forgeable under chosen-message attack. The ECDSA have a smaller key size, which leads to faster computation time and reduction in processing power, storage space and bandwidth. This makes the ECDSA ideal for constrained devices such as pagers, cellular phones and smart cards.

The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the DSA. ECDSA was first proposed in 1992 by Scott Vanstone [1] in response to NIST's (National Institute of Standards and Technology) request for public comments on their first proposal for DSS. It was accepted in 1998 as an ISO (International Standards Organization) standard (ISO 14888-3), accepted in 1999 as an ANSI (American National Standards Institute) standard (ANSI X9.62), and accepted in 2000 as an IEEE (Institute of Electrical and Electronics Engineers) standard (IEEE 1363-2000) and a FIPS standard (FIPS 186-2)

Digital signature schemes can be used to provide the following basic cryptographic services:

- data integrity (the assurance that data has not been altered by unauthorized or unknown means)
- data origin authentication (the assurance that the source of data is as claimed)
- non-repudiation (the assurance that an entity cannot deny previous actions or commitments)

In this paper, first we start with the cryptography schemes based on integer factorization (IF) and discrete logarithm (DL) in section 2. In section 3, we discuss ECC in detail. In section 4, we show the implementation and results. Further in section 5 and 6 we compare and conclude respectively.

2. CRYPTOGRAPHIC SCHEMES

2.1 Integer Factorization

In Integer factorization, given an integer n which is the product of two large primes p and q such that:

$$n = p * q \quad (1)$$

It is easy to calculate n for given p and q but it is computationally infeasible to determine p and q given n for large values of n. One of the famous algorithms is RSA. The RSA Algorithm is shown below:

1. Choose two large prime numbers, p and q (1024 bits)
2. Compute n = p * q and z = (p-1) * (q-1).
3. Choose a number, e, less than n, which has no common factors (other than 1) with z.

4. Find a number, d, such that e * d -1 is exactly divisible (i.e., with no remainder) by z.

The public key is the pair of numbers (n, e), private key is the pair of numbers (n, d).

The encryption is done as follows:

$$c = m^e \bmod n \quad (2)$$

To decrypt the received cipher text message, c

$$m = c^d \bmod n \quad (3)$$

which requires the use of the private key, (n, d).

Its security depends on the difficulty of factoring the large prime numbers. The best known method for solving Integer Factorization problem is Number Field Sieve which is a sub-exponential algorithm and having a running time of $\exp[1.923*(\log n)^{1/3}*(\log \log n)^{2/3}]$ [2].

2.2 Discrete Logarithm

Discrete logarithms are ordinary logarithms involving group theory. An ordinary logarithm $\log_a(b)$ is a solution of the equation $a^x = b$ over the real or complex numbers. Similarly, if g and h are elements of a finite cyclic group G then a solution x of the equation $g^x = h$ is called a discrete logarithm to the base g of h in the group G, i.e. $\log_g(h)$. A group with an operation * is defined on pairs of elements of G. The operations satisfy the following properties:

1. **Closure:** a * b ε G for all a, b ε G.
2. **Associativity:** a * (b * c) = (a * b) * c for all a, b ε G.
3. **Existence of Identity:** There exists an element e ε G, called the identity, such that e * a = a * e = a for all a ε G.
4. **Existence of inverse:** For each a ε G there is an element b ε G such that a * b = b * a = e. The element b is called the inverse of a.

Moreover, a group G is said to be abelian if a * b = b * a for all a, b ε G. The order of a group G is the number of elements in G.

The discrete logarithm problem is to find an integer x, $0 \le x \le n-1$, such that $g^x \equiv h \pmod p$, for given g ε Z*p of order n and given h ε Z*p. The integer x is called the discrete logarithm of h to the base g.

Digital Signature Algorithm (DSA), Diffie Hellman (DH) and El Gamal are based on discrete logarithms.

The best known method for solving Discrete Logarithm problem is Number Field Sieve which is a sub-exponential algorithm, having a running time of $\exp[1.923*(\log n)^{1/3}*(\log \log n)^{2/3}]$ [2].

2.2.1 Comparison with Integer Factorization

While the problem of computing discrete logarithms and the problem of integer factorization are distinct problems they share some properties:

- both problems are difficult (no efficient algorithms are known for non-quantum computers),
- for both problems efficient algorithms on quantum computers are known,
- algorithms from one problem are often adapted to the other, and
- difficulty of both problems has been exploited to construct various cryptographic systems.

2.2.2 Elliptic Curve Discrete Logarithm

An elliptic curve E_k, [3] defined over a field K of characteristic ≠ 2 or 3 is the set of solutions (x, y) ε K' to the equation

$$y^2 = x^3 + ax + b \quad (4)$$

a, b ε K (where the cubic on the right has no multiple roots).

Two nonnegative integers, a and b, less than p that satisfy:

$$4a^3 + 27b^2 \pmod p \neq 0 \quad (5)$$

Then E_p (a, b) denotes the elliptic group mod p whose elements (x, y) are pairs of nonnegative integers less than p satisfying:

$$y^2 \equiv x^3 + ax + b \pmod p \quad (6)$$

together with the point at infinity O.

The elliptic curve discrete logarithm problem can be stated as follows. Fix a prime p and an elliptic curve.

$$Q = xP \quad (7)$$

where xP represents the point P on elliptic curve added to itself x times. Then the elliptic curve discrete logarithm problem is to determine x given P and Q. It is relatively easy to calculate Q given x and P, but it is very hard to determine x given Q and P.

ECC is based on ECDLP. ECDH and ECDSA are cryptographic schemes based on ECC. The best known algorithm for solving ECDLP is Pollard-Rho algorithm which is fully exponential having a running time of $\sqrt{(\prod * n / 2)}$ [2].

3. ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic curve cryptosystems (ECC) were invented by Neal Koblitz [3] and Victor Miller [4] in 1985. They can be viewed as elliptic curve analogues of the older discrete logarithm (DL) cryptosystems in which the subgroup of Z_p^* is replaced by the group of points on an elliptic curve over a finite field. The mathematical basis for the security of elliptic curve cryptosystems is the computational intractability of the elliptic curve discrete logarithm problem (ECDLP) [5].

ECC is a relative of discrete logarithm cryptography. An elliptic curve E over Z_p as in Figure 1 is defined in the Cartesian coordinate system by an equation of the form:

$$y^2 = x^3 + ax + b \quad (8)$$

where a, b ε Z_p, and $4a^3 + 27b^2 \neq 0 \pmod p$, together with a special point O, called the point at infinity. The set $E(Z_p)$ consists of all points (x, y), x ε Z_p, y ε Z_p, which satisfy the defining equation, together with O.

Each value of a and b gives a different elliptic curve. The public key is a point on the curve and the private key is a random number. The public key is obtained by multiplying the private key with a generator point G in the curve.

The definition of groups and finite fields, which are fundamental for the construction of elliptic curve cryptosystem are discussed in next subsections.

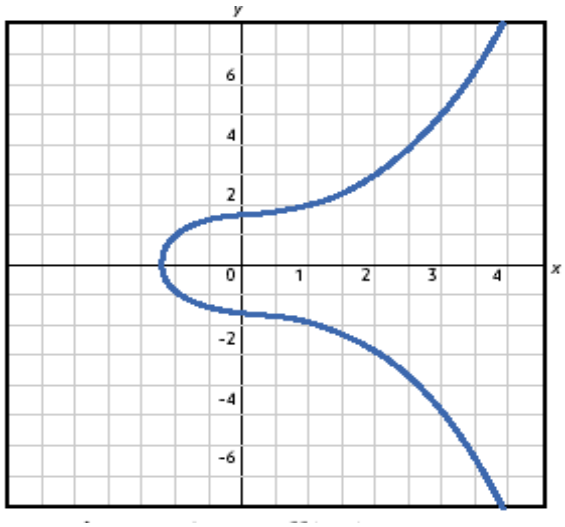

Figure 1. An Elliptic Curve

3.1 Groups

A group with an operation * is defined on pairs of elements of G. The operations satisfy the following properties:

- **Closure:** a * b ε G for all a, b ε G.
- **Associativity:** a * (b * c) = (a * b) * c for all a, b ε G.
- **Existence of Identity:** There exists an element e ε G, called the identity, such that e * a = a * e = a for all a ε G.
- **Existence of inverse:** For each a ε G there is an element b ε G such that a * b = b * a = e. The element b is called the inverse of a.

Moreover, a group G is said to be abelian if a * b = b * a for all a, b ε G. The order of a group G is the number of elements in G.

3.2 Finite Field

A finite field consists of a finite set of elements together with two binary operations called addition and multiplication, which satisfy certain arithmetic properties. The order of a finite field is the number of elements in the field. There exists a finite field of order q if and only if q is a prime power. If q is a prime power, then there is essentially only one finite field of order q; this field is denoted by F_q. There are, however, many ways of representing the elements of F_q. Some representations may lead to more efficient implementations of the field arithmetic in hardware or in software. If $q=p^m$ where p is a prime and m is a positive integer, then p is called the characteristic of F_q and m is called the extension degree of F_q.

3.2.1 Prime Field F_p

Let p be a prime number. The finite field F_p called a prime field, is comprised of the set of integers {0,1,2,….,p-1} with the following arithmetic operations:

- **Addition**: If a, b ε F_p then a+b=r, where r is the remainder when a+b is divided by p and $0 \le r \le p-1$ known as addition modulo p.
- **Multiplication**: If a, b ε F_p then a.b=s, where s is the remainder when a.b is divided by p and $0 \le s \le p-1$ known as multiplication modulo p
- **Inversion**: If is a non-zero element in F_p, the inverse of modulo a modulo p_, denoted by a^{-1}, is the unique integer c ε F_p for which a.c=1

3.2.2 Binary Field F_2^m

The field F_2^m, called a characteristic two finite field or a binary finite field, can be viewed as a vector space of dimension m over the field F_2 which consists of the two elements 0 and1. That is, there exist m elements $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ in F_2^m such that each element α can be uniquely written in the form:

$\alpha = a_0 \alpha_0 + a_1 \alpha_1 + \ldots\ldots\ldots + a_{m-1} \alpha_{m-1}$, where $a_i \varepsilon \{0,1\}$

Such a set $\{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$ is called a basis of F_2^m over F_2. Given such a basis, a field element α can be represented as the bit string (a_0 a_1 …….+a_m.1) Addition of field elements is performed by bitwise XOR-ing the vector representations. The multiplication rule depends on the basis selected. ANSI X9.62 permits two kinds of bases: polynomial bases and normal bases.

3.2.3 Domain Parameters

The domain parameters for ECDSA consist of a suitably chosen elliptic curve E defined over a finite field F_q of characteristic p, and a base point G ε $E(F_q)$. Domain parameters may either be shared by a group of entities, or specific to a single user. To summarize, domain parameters are comprised of:

1. a field size q, where either q=p, an odd prime, or $q=2^m$
2. an indication FR (field representation) of the representation used for the elements of F_q
3. (optional) a bit string seedE of length at least 160 bits
4. 4. two field elements a and b in F_q which define the equation of the elliptic curve E over F_q' (i.e., $y^2 = x^3 + ax + b$ in the case p>3, and $y^2 + xy = x^3 + ax + b$ in the case p=2)
5. two field elements x_G and y_G in F_q which define a finite point G=(x_G, y_G) of prime order in $E(F_q)$
6. the order εof the point G, with $n>2^{160}$ and $n>4\sqrt{q}$ and
7. the cofactor h= $\#E(F_q)/n$

3.3 Elliptic Curves Operations over Finite Fields [6]

The main operation is Point multiplication is achieved by two basic elliptic curve operations.

1. Point addition, adding two points J and K to obtain another point L i.e. L= J + K, require 1 inversion and 3 multiplication operation.

2. Point doubling, adding a point J to itself to obtain another point L i.e. L = 2J, requires 1 inversion and 4 multiplication operation.

3.3.1 Point Addition

Point addition is the addition of two points J and K on an elliptic curve to obtain another point L on the same elliptic curve.

Consider two points J and K on an elliptic curve as shown in Figure 2. If K≠ -J then a line drawn through the points J and K will intersect the elliptic curve at exactly one more point –L. The reflection of the point –L with respect to x-axis gives the

point L, which is the result of addition of points J and K. Thus on an elliptic curve L = J + K. If K = -J the line through this point intersect at a point at infinity O. Hence J + (-J) = O. A negative of a point is the reflection of that point with respect to x-axis [7].

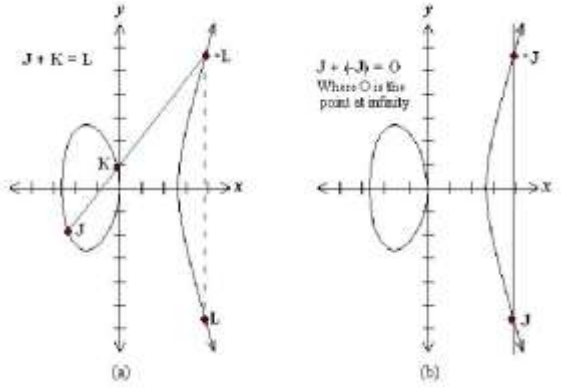

Figure 2. Point Addition

3.3.2 Point doubling

Point doubling is the addition of a point J on the elliptic curve to itself to obtain another point L on the same elliptic curve To double a point J to get L, i.e. to find L = 2J, consider a point J on an elliptic curve as shown in Figure 3. If y coordinate of the point J is not zero then the tangent line at J will intersect the elliptic curve at exactly one more point –L. The reflection of the point –L with respect to x-axis gives the point L, which is the result of doubling the point J, i.e., L = 2J. If y coordinate of the point J is zero then the tangent at this point intersects at a point at infinity O. Hence 2J = O when y_j=0. Figure 3 shows point doubling [7].

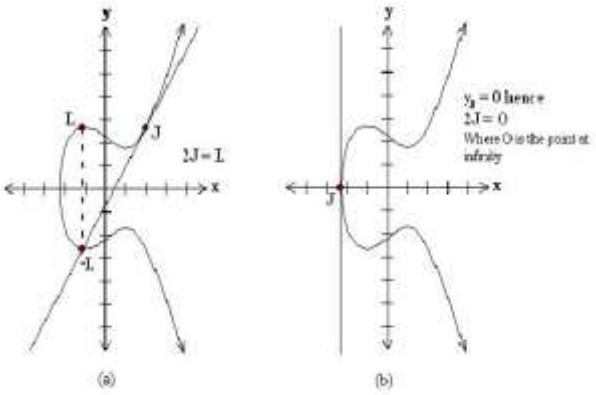

Figure 3. Point Doubling

3.3.3 Algebraic Formulae Over F_p

- P+O=O+P=P for all P ε $E(F_p)$
- If P=(x, y) ε $E(F_p)$ then (x, y)+(x,-y)=O. (The point (x,-y) is denoted by –P, and is called the negative of P, observe that –P is indeed a point on the curve.
- Point addition Let P=(x_1, y_1) ε $E(F_p)$ and Q=(x_2, y_2) ε $E(F_p)$, where P≠ ± Q. Then P+Q=(x_3, y_3) where
  $x_3= \{(y_2-y_1)/(x_2-x_1)\}^2 - x_1-x_2$ and $y_3= \{(y_2-y_1)/(x_2-x_1)\}(x_1-x_3) -y_1$
- Point doubling Let P=(x_1, y_1) ε $E(F_p)$ where P≠ -P. Then 2P=(x_3, y_3) where
  $x_3=\{(3x_1^2+a)/2y_1\}^2-2x_1$ and $y_3=\{(3x_1^2+a)/2y_1\}^2 (x_1-x_3) -y_1$

3.3.4 Algebraic Formulae Over F_2^m

- P+O=O+P=P for all P ε $E(F_2^m)$
- If P=(x, y) ε $E(F_p)$ then (x, y)+(x, -y)=O. (The point (x, -y) is denoted by –P, and is called the negative of P, observe that –P is indeed a point on the curve.
- (Point addition) Let P=(x_1, y_1) ε $E(F_2^m)$ and Q=(x_2, y_2) ε $E(F_2^m)$, where P≠ ± Q. Then P+Q=(x_3,y_3) where
  $x_3= \{(y_2+y_1)/(x_2+x_1)\}^2 + \{(y_2+y_1)/(x_2+x_1)\}+ x_1 + x_2 +a$
  and $y_3= \{(y_2+y_1)/(x_2+x_1)\}(x_1+x_3) +x_3 + y_1$
- (Point doubling) Let P=(x_1,y_1) ε $E(F_2^m)$ where P≠ -P. Then 2P=(x_3,y_3)

where $x_3=x_1^2+(b/ x_1^2)$ and $y_3=x_1^2+\{x_1+(y_1/x_1)\}x_3 + x_3$

4. IMPLEMENTAION AND RESULTS

Elliptic Curve Digital Signature Algorithm is implemented over elliptic curve P-192 as mandated by ANSI X9.62 in C language. The Project contains necessary modules for domain parameters generation, key generation, signature generation, and signature verification over the elliptic curve.

ECDSA has three phases, key generation, signature generation, and signature verification.

ECDSA Key Generation:

An entity A's key pair is associated with a particular set of EC domain parameters D= (q, FR, a, b, G, n, h). E is an elliptic curve defined over F_q , and P is a point of prime order n in $E(F_q)$, q is a prime. Each entity A does the following:

1. Select a random integer d in the interval [1, n- 1].

2. Compute Q = dP.

3. A's public key is Q, A's private key is d.

ECDSA Signature Generation:.

To sign a message m, an entity A with domain parameters D= (q, FR, a, b, G, n, h) does the following:

1. Select a random or pseudorandom integer k in the interval [1, n-1].

2. Compute kP =x_1, y_1 and r= x_1 mod n (where x_1 is regarded as an integer between 0 and q-1). If r= 0 then go back to step 1.

3. Compute k^{-1}mod n.

4. Compute s= k^{-1} {h (m)+ dr} mod n, where h is the Secure Hash Algorithm (SHA-1). If s = 0, then go back to step 1.

5. The signature for the message m is the pair of integers (r, s).

ECDSA Signature Verification:

To verify A's signature (r, s) on m, B obtains an authenticated copy of A's domain parameters D = (q, FR, a, b, G, n, h) and public key Q and do the following

1. Verify that r and s are integers in the interval [1, n-1].
2. Compute $w = s^{-1} \bmod n$ and h (m)
3. Compute $u_1 = h(m)w \bmod n$ and $u_2 = rw \bmod n$.
4. Compute $u_1P + u_2Q = (x_0, y_0)$ and $v = x_0 \bmod n$.
5. Accept the signature if and only if v = r

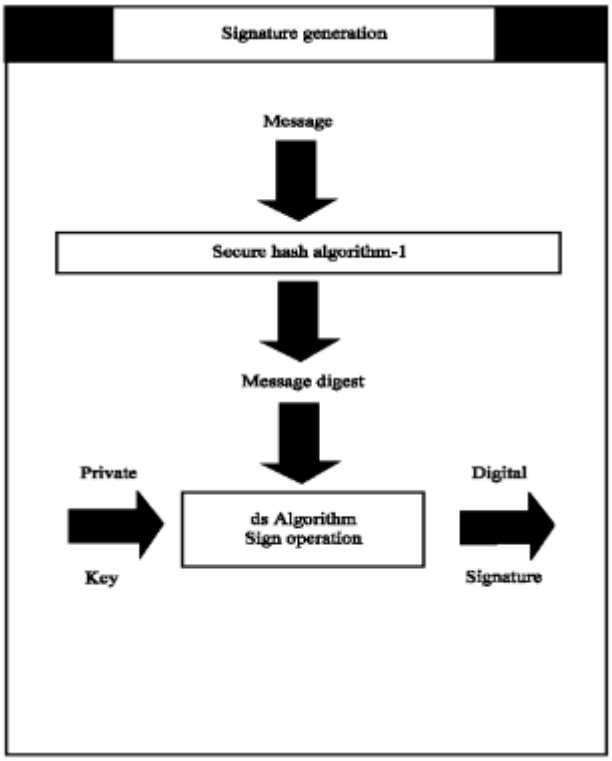

Figure 4. Signature Generation

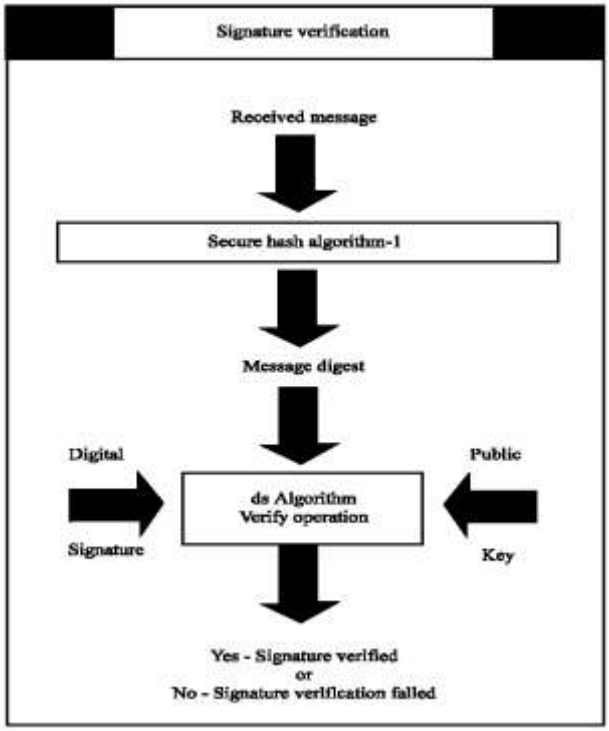

Figure 5. Signature Verification

Results

The following results are brought to highlight for given set of values.

The SHA-1 result are shown along with the private and public set of keys

SHA–1

Input: "a"

SHA Output: 86f7e437faa5a7fce15d1ddcb9eaeaea377667b8

Input: "ABC"

SHA Output: 3c01bdbb26f358bab27f267924aa2c9a03fcfdb8

Key Pair Generation:

198 bit random private key and corresponding public key:

Private A=

3410708343957475413710496549104959138812316708511486831983 98465

Public x of A=

3089182225850909019933101519334356466906901301271156815371

Public y of A=

2934312592567055080539106109257350191706192298057173813254

Private A=

9784754507269478441147399409938745992633565457803056150961 4891

Public x of A=

5794350039132556514670158969918976743409250716115312636030

Public y of A=

1009024622477364832125741509919741456473929964192222324391

Further for a given input file containing text had been taken and signature is generated and then verified by the values of r and s.

Signature Generation:

Input file="abcd"

Private:0xd43fb7ff56a7486859d87f785db45b043129f6468ccff4 2d0001

Signature:

r=0xb8d06fa44816c92b8b26f797e5f3cc07984d8b7f7e49a339

s=0xd74f17a1e19139d77558c6b2d16dcb1f4bb31da2ded25733

Proof of verification

If a signature (r, s) on a message m was indeed generated by A, then $s = k^{-1} (h (m)+dr) \bmod n$. Rearranging gives $k \equiv s^{-1} (e+dr) \equiv s^{-1}e + s^{-1} rd \equiv we + wrd \equiv u_1+u_2d \pmod n$.

Thus $u_1G + u_2Q = (u_1 + u_2d)\, G = kG$ and so v=r as required.

5. COMPARISON WITH RSA and DSA

In all cryptography systems discussed so far, there is a comparative difficulty of doing two types of operations-a forward operation which must be tractable [8], and an inverse operation which must be intractable. The degree of difference between the difficulties of these operations depends on the size of the key pairs. The inverse operation increases exponentially whereas the forward operation increases linearly as the key size increases as in Figure 6. Increase in key length give rise to complexity issues in both operations. Thus ECC is preferred as it provides same level security at 160 bit key length as of 1024 bit key length in RSA.

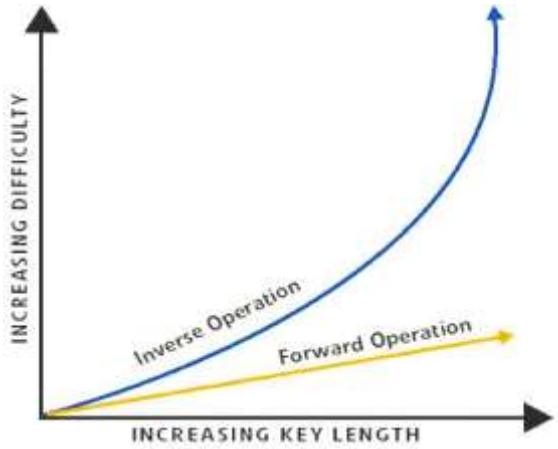

Figure 6. Difficulty of forward, inverse operation against key length

Table 5.1 shows the comparison of ECC with RSA, DSA, and DH in terms of key length and time to break on machine running 1 MIPS [9].

Table 1 Key comparison of Symmetric, RSA/DSA/DH, ECC

Symmetric	RSA/DSA/DH	ECC	Time to break in MIPS years
80	1024	160	10^{12}
112	2048	224	10^{24}
128	3072	256	10^{28}
192	7680	384	10^{47}
256	15360	512	10^{66}

5.1 Comparison of ECC with RSA

1. RSA takes sub-exponential time and ECC takes full exponential time. For example, RSA with key size of 1024 bits takes $3x10^{11}$ MIP years with best known attack where as ECC with 160 bit key size takes 9.6x 10^11 MIP years[10].
2. ECC offers same level of security with smaller key sizes.
3. DATA size for RSA is smaller than ECC.
4. Encrypted message is a function of key size and data size for both RSA and ECC. ECC key size is relatively smaller than RSA key size, thus encrypted message in ECC is smaller.
5. Computational power is smaller for ECC.

5.2 Comparison of ECDSA with DSA

1. Both algorithms are based on the ElGamal signature scheme and use the same signing equation: $s = k^{-1}\{h (m) + dr\} \bmod n$.
2. In both algorithms, the values that are relatively difficult to generate are the system parameters(p, q and g for the DSA; E, P and n for the ECDSA).

3. In their current version, both DSA and ECDSA use the SHA-1 as the sole cryptographic hash function.
4. The private key d and the per-signature value k in ECDSA are defined to be statistically unique and unpredictable rather than merely random as in DSA [11].

5.3 Advantages of ECC

Thus, the ECC offered remarkable advantages over other cryptographic system.

1. It provides greater security for a given key size.
2. It provides effective and compact implementations for cryptographic operations requiring smaller chips.
3. Due to smaller chips less heat generation and less power consumption.
4. It is mostly suitable for machines having low bandwidth, low computing power, less memory.
5. It has easier hardware implementations.

So far no drawback of ECC had been reported.

6. CONCLUSION

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 algorithm known for solving the underlying integer factorization for RSA and discrete logarithm problem in DSA both take sub exponential time. The key generated by the implementation is highly secured and it consumes lesser bandwidth because of small key size used by the elliptic curves. 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 size include faster computation time and reduction in processing power, storage space and bandwidth. This makes ECDSA ideal for constrained environments such as pagers, 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.

7. REFERENCES

[1] Vanstone, S. A., 1992. Responses to NIST's Proposal Communications of the ACM, 35, 50-52.

[2] Vanstone, S. A., 2003. Next generation security for wireless: elliptic curve cryptography. Computers and Security, vol. 22, No. 5.

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

[4] Miller, V., 1985. Use of elliptic curves in cryptography. CRYPTO 85.

[5] Certicom ECC Challenge. 2009. Certicom Research

[6] Hankerson, D., Menezes, A., Vanstone, S., 2004. Guide to Elliptic Curve Cryptography. Springer.

[7] Botes, J.J., Penzhorn, W.T., 1994. An implementation of an elliptic curve cryptosystem. Communications and Signal Processing. COMSIG-94. In Proceedings of the 1994 IEEE South African Symposium, 85 -90.

[8] An intro to Elliptical Curve Cryptography[On-Line]. Available:http://www.deviceforge.com/articles/AT4234154468.html [2010].

[9] Gupta, V., Stebila, D., Fung, S., Shantz, S.C., Gura, N., Eberle, H., 2004. Speeding up Secure Web Transactions Using Elliptic Curve Cryptography. In Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS 2004). The Internet Society, 231-239.

[10] Raju, G.V.S., Akbani, R., 2003. Elliptic Curve Cryptosystem And Its Application. In Proceedings of the 2003 IEEE International Conference on Systems Man and Cybernetics (IEEE-SMC), 1540-1543.

[11] Johnson, D.B., Menezes, A.J., 2007. Elliptic Curve DSA (ECDSA): An Enhanced DSA. Scientific Commons.