Two Key Signature Scheme with Application to Digital Certificates

N. Anil Kumar
DST-CMS, C R Rao Advanced Institute of Mathematics Statistics and Computer Science (AIMSCS)
Prof C R Rao Road, Hyderabad-500 046
anil230@gmail.com

Chakravarthy Bhagvati
Department of Computer and Information Sciences
University of Hyderabad
Hyderabad-500 046 India.
chakcs@uohyd.ernet.in

Abstract: **Elliptic curves offer all the features of the conventional cryptography like digital signatures, data encryption / decryption, key exchange, identification, etc. at a reduced key size. Standardized elliptic curve based signature schemes are described in ANSI X9.62, IEEE 1363-2000, etc. We propose a two key digital signature scheme using elliptic curves and compare our scheme with the standard scheme and found that with respective to brute force method our scheme is more secure. We suggest an application of the proposed two key signature scheme in Digital Certificates.**

Keywords: Digital Certificates; Digital signatures; Elliptic curves.

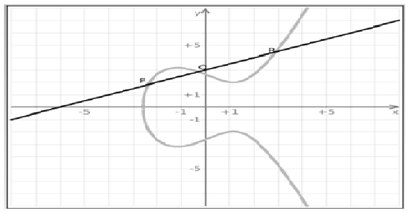
I. INTRODUCTION

A signature is a technique for non-repudiation based on the public key cryptography. A digital signature or digital signature scheme is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit. [1]

II. PRELIMINARIES

Simplified Weierstrass equation for elliptic curve is $y^2 = x^3 + ax + b$ where a, b, x and y belongs to some field with $4a^3 + 27b^2 \neq 0$ (in that field). Let P and Q be two points on the elliptic curve. The line joining the two points will intersect the curve at the third point say R. The addition of points is defined as

$$P + Q + R = identity$$



To make elliptic curve addition a group, identity is defined to be a point at infinity denoted as O or ∞. A line is said to pass through point at infinity when it is exactly vertical.

Let *P*, *Q* and *R* be the points on the elliptic curve then the following holds.

- *P*+*Q* will be point on the curve (Closure)
- *P*+*Q*=*Q*+*P* (Commutativity)
- (*P*+*Q*) +*R*=*P*+(*Q*+*R*) (Associativity)
- *P*+*O*=*O*+*P*=*P* (Existence of an identity element)
- There exist *(-P)* such that $(-P) + P = P + (-P)$ = (*O*) (Existence of inverse)

Scalar multiplication is defined a repeated addition. Let *n* be integer.

$$nP = P + P + \cdots n \text{ times}$$

The points on the elliptic curve form an abelian group.

Let $P(x_1, y_1) + Q(x_2, y_2) = R(x_3, y_3)$ then

$$x_3 = m^2 - x_1 - x_2 \text{ and } y_2 = m(x_1 - x_3) - y_1$$

Where $m = \dfrac{y_2 - y_1}{x_2 - x_1}$ for $P \neq Q$

$$= \frac{3x_1^2 + a}{2y_1} \quad \text{for } P = Q$$

The elliptic curve arithmetic which have been defined over real numbers can also be defined over finite fields. Most of the definitions over real numbers can be carried over to finite fields. [2, 3] gives detailed description on arithmetic of elliptic curves.

Let $q = p^k$ where *p* is prime. F_q is the Galois field of order *q*. Let E be the elliptic curve. The set of points on the elliptic curve $E(F_q)$ is

$$E(F_q) = \{O\} \cup \{(x,y) \in F_q \times F_q \mid y^2 = x^3 + ax + b$$

III. Elliptic Curve Digital Signature Algorithm

The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the Digital Signature Algorithm (DSA). We refer to this as one key ECDSA. It is the most widely standardized elliptic curve-based signature scheme, appearing in the ANSI X9.62, FIPS 186-2, IEEE 1363-2000 and ISO/IEC 15946-2 standards as well as several draft standards [4, 5].

The Domain parameters D = (q, FR, S, a, b, P, n, h) are comprised of

1. The field order q.
2. An indication FR (Field Representation) of the representation used for the elements of F_q.
3. A seed S if the elliptic curve was randomly generated.
4. Two coefficients $a, b \in F_q$ that define the equation of the elliptic curve E over F_q (i.e. $y^2 = x^3 + ax + b$ in the case of a prime field and $y^2 + xy = x^3 + ax^2 + b$ in case of binary field).
5. Two field elements x_p and y_p in F_q that define a finite point $P = (x_p, y_p) \in E(F_q)$ in affine coordinates. P is called the base point.
6. The order of P is n.
7. The cofactor $h = \# E(F_q) / n$.

In the following, H denotes a cryptographic hash function whose outputs have bit length no more than that of *n* (if this condition is not satisfied, then the outputs of H can be truncated).

A. *ECDSA Signature generation*

Input: Domain parameters D = (q, FR, S, a, b, P, n, h), private key d, message m.

Output: Signature (r, s).

1. Select $k \in_R [1, n-1]$
2. Compute $kP = (x_1, y_1)$ and convert x_1 to an integer $\bar{x}_1$.
3. Compute $r = \bar{x}_1$ mod n. If $r = 0$ then goto step 1.
4. Compute e = H(m).
5. Compute $s = k^{-1}(e + dr)$ mod *n*. If $s = 0$ then goto step 1.
6. Return (r, s).

B. *ECDSA signature Verification*

Input: Domain parameters D = (q, FR, S, a, b, P, n, h), public key Q, message m, signature (r, s).

Output: Acceptance or rejection of the signature.

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 $X = u_1 P + u_2 Q$.
6. If $X = \infty$ then return ("Reject the signature").
7. Convert the x-coordinate x_1 of X to an integer $\bar{x}_1$; Compute $v = \bar{x}_1 \bmod n$.
8. If v = r then return ("Accept the signature"); Else return ("Reject the signature").

C. *Proof that signature verification works*

If a signature (r, s) on a message m was indeed generated by the legitimate signer, then $s \equiv k^{-1}(e + dr)$ mod *n* Rearranging gives

$$k \equiv s^{-1}(e + dr) \equiv s^{-1}e + s^{-1}rd \pmod n$$

$$\equiv we + wrd \equiv (u_1 + u_2)d \pmod n$$

Thus $X = u_1 P + u_2 Q = (u_1 + u_2)d = kp$ and so $v = r$ as required.

IV. Two Key ECDSA

The domain parameters

$$D = (q, FR, S, a, b, P, P_1, R, n, h)$$

are same as the original ECDSA except P_1 and R. P_1 is another base point.

Let *d* and d_1 be the private keys, $dP + d_1 P_1 = R$ and $dP = Q$

P, P_1 and *R* are the public parameters.

H defines the hash function as in ECDSA.

A. *Two key ECDSA Signature generation*

Input: Domain parameters

$D = (q, FR, S, a, b, P, P_1, R, n, h)$, private key *d* and d_1, message *m*.

Output: Signature (x_1, s_1, s_2).

1. Select k_1 and $k_2 \in_R [1, n-1]$
2. Compute $k_1P + k_2P_1 = S(x_1, y_1)$ If $S = \infty$ goto step 1.
3. Compute e = H(m). Convert the field element x_1 to an integer x (usually the first coordinate in the vector representation)
4. Let $s_1 = ek_1 + xd$ and $s_2 = ek_2 + xd_1$.
5. Return (x_1, s_1, s_2) .

B. Two key ECDSA signature Verification

Input: Domain parameters

$$D = (q, FR, S, a, b, P, P_1, R, n, h)$$

public key P, Q, P_1, R message m, signature (x_1, s_1, s_2)

Output: Acceptance or rejection of the signature.

1. Compute e = H(m).
2. Compute $u_1 = s_1P + s_2P_1$.
3. Compute y_1 and y_1^1 from x_1 using curve equation. Let $T = (x_1, y_1)$ and $T^1 = (x_1, y_1^1)$
4. Compute $u_2 = eT + x_1R$.
5. If $u_1 = u_2$ accept the signature, else if $u_2 = eT^1 + x_1R$ accept the signature.
6. Else reject the signature.

C. Proof that the signature verification works

If a signature (x_1, s_1, s_2) on a message m was indeed generated by the legitimate signer, then

$$\begin{aligned} s_1P + s_2P_1 &= (ek_1 + x_1d)P + (ek_2 + x_1d_1)P_1 \\ &= e(dP + d_1P_1) + x_1(k_1P + k_2P_1) \\ &= eT + x_1R \end{aligned}$$

D. Comparision of ECDSA and Two key ECDSA

The domain parameters of ECDSA are subset of two key ECDSA. If we want to sign a message with one key we can use standard ECDSA with the domain parameters of two key ECDSA. The person who is signing can decide whether to use two keys or one key.

The brute force method to attack the ECDSA is to find d (private key) from public key Q and domain parameter P which satisfy the relationship *Q=dP*. The order of *P* is *n* so to find Q we have to check *n* possibilities. So the time complexity is O (n). The brute force method to attack the Two key ECDSA is to find *d* and d_1 (private keys) from P, P_1 and *S* which satisfy the relationship $dP + d_1P_1 = S$. The order of *P* is n_1 and order of P_1 is n_2 . To find *S* we have to check $n_1 \times n_2$ possibilities. So the time complexity is O($n_1 \times n_2$).

Carefully chosen parameter results in more time complexity for MECDSA than ECDSA with respective to the brute force method

V. Digital Certificates

Deffi-Hellman Key exchange designed specifically to solve the problem of key exchange has the problem of man-in-the-middle attack. The problem can be resolved by digital certificates. Digital Certificate is a computer file with encoding as in described in [6]. The certificate is described in ASN.1 (Abstract Syntax Notation One) as

```
Certificate ::= SEQUENCE {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      BIT STRING
    }


 TBSCertificate ::= SEQUENCE {
    version         [0] EXPLICIT Version DEFAULT v1,
    serialNumber        CertificateSerialNumber,
    signature           AlgorithmIdentifier,
    issuer              Name,
    validity            Validity,
    subject             Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID  [1]  IMPLICIT UniqueIdentifier
                         OPTIONAL,
                  -- If present, version MUST be v2 or v3
    subjectUniqueID [2]  IMPLICIT UniqueIdentifier
                         OPTIONAL,
                  -- If present, version MUST be v2 or v3
    extensions      [3]  EXPLICIT Extensions
OPTIONAL
                  -- If present, version MUST be v3
    }


  Version ::= INTEGER { v1(0), v2(1), v3(2) }
  CertificateSerialNumber  ::=  INTEGER
   Validity ::= SEQUENCE {
      notBefore      Time,
      notAfter       Time
      }
  Time ::= CHOICE {
      utcTime        UTCTime,
      generalTime    GeneralizedTime
      }
  UniqueIdentifier  ::=  BIT STRING
  SubjectPublicKeyInfo  ::=  SEQUENCE  {
      algorithm            AlgorithmIdentifier,
      subjectPublicKey     BIT STRING
      }
Extensions  ::=  SEQUENCE SIZE (1...MAX) OF Extension
 Extension  ::=  SEQUENCE  {
      extnID      OBJECT IDENTIFIER,
      critical    BOOLEAN DEFAULT FALSE,
      extnValue   OCTET STRING
      }
```

The format is encoded using Distinguished Encoding Rules (DER) to the binary (zeros and ones). This is saved as a computer file with extension .cer which is called digital certificate.

To use the proposed two key signature algorithm in digital certificates we have to make modifications to subjectPublicKey and AlgorithmIdentifier. The subjectPublicKey gives the information of the public keys and all the domain parameters of the two key signature scheme. A new value for AlgorithmIdentifier is defined.

A. *Usage*

A subset of domain parameters of Two key ECDSA are required to use One key ECDSA. The user can use either one key or two key ECDSA algorithms depending on the security requirements for signing the documents, SSL encryption, etc. If the user wants less security he can use one key ECDSA and if he wants more security he can use two key ECDSA.

VI. CONCLUSIONS

We propose a two key digital signature scheme using elliptic curves and compare our scheme with the standard scheme and found that with respective to brute force method our scheme is more secure and demonstrated its application in digital certificates. The advantage with this scheme is that the user has the choice to choose either one key scheme or two key scheme depending on the requirement.

ACKNOWLEDGEMENTS

This work is supported by DST-CMS project Lr.No.SR/S4/MS:516/07, Dt.21-04-2008.

REFERENCES

[1] http://en.wikipedia.org/wiki/Digital_signature

[2] V. Miller, "Use of elliptic curves in cryptography," Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85, pp. 417–426, 1986.

[3] L. Washington, Elliptic Curves: Number Theory and Cryptography. CRC Press, 2003.

[4] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for Key Management–Part 1: General," NIST Special Publication, pp. 800 – 857, 2005.

[5] Standards for Efficient Cryptography Group (SECG), "Sec 1: Elliptic curve cryptography," September 2000, http://www.secg.org/download/aid-385/sec1_final.pdf

[6] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, May, Tech. Rep., 2008.

[7] S. Tuecke, D. Engert, I. Foster, M. Thompson, L. Pearlman, and C. Kesselman, "Internet X. 509 public key infrastructure proxy certificate profile," IETF, Draft draft-ietfpkix-proxy-01. txt, 2001.

[8] J. Cannon *et al.*, "The MAGMA computational algebra system," Software available on line http://magma.maths.usyd.edu.au, 2005.

[9] Darrel Hankerson, Affred Menezes, and Scott Vanstone, Guide to Elliptic Curve Cryptography, Springer Verlag, 2004.

[10] R. Housley, W. Polk, W. Ford and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 3280 Tech. Rep., 2002.

[11] N.Anil Kumar, R.Tandon and Chakravarthy Bhagvati, "Modified elliptic curve digital signature algorithm," In Proceedings of the International Conference on Mathematics and Computer Science 2009, Vol: 1, pp: 304-306, 2009.