Seele: A Secure Multiple Elliptic Curves Digital Signature Algorithm for Blockchain

Wei Bi*, Xiaoyun Jia, Maolin Zheng
Seele Tech Corporation, San Francisco, USA
wei@seelenet.com

Abstract: Most cryptocurrency systems or systems based on blockchain technology are currently using the elliptic curves digital signature algorithm (ECDSA) on the secp256k1 curve, which is susceptible to backdoors implemented by the curve creator (secp256k1). Seele proposes a multiple elliptic curves digital signature algorithm (MECDSA), which allows not only for setting the number of elliptic curves according to practical security requirements, but also for editing the parameters of each elliptic curve. The performance analysis proves that the scheme is secure and efficient, and can avoid any backdoors implemented by curve creators. Seele suggests that the systems based on blockchain should operate in two elliptic curves considering the contradiction between security and efficiency.

Keywords: Digital Signature, Blockchain, Elliptic Curve Discrete Logarithm Problem (ECDLP).

1 Introduction

Digital signatures1-2 are used to detect unauthorized modifications to data and to authenticate the identity of the signer. The recipient of signed message can use a digital signature as evidence to demonstrate to a third party that the signature was generated by the claimed signer. Digital signatures uses a combination of hash functions and public key cryptography. Firstly, the message digest is created using a hash function. The signature is created by encrypting the message digest using the signer’s private key. Anyone can use the signer’s public key and the same hash function to verify the received signature.

Until now, most cryptocurrency systems3-5 or systems based on blockchain technology6-7 have used in elliptic curves digital signature algorithm (ECDSA)8-10, which is based on secp256k111. Scep256k1 was almost never used before Bitcoin became popular, but it is now gaining in popularity due to several beneficial properties. Most commonly-used curves have random structure, but secp256k1 was constructed in a unique, non-random way which allows for especially efficient computation. As a result, it is often more than 30% faster than other curves if the implementation is sufficiently optimized. However, it cannot prevent the curve’s creator from inserting any backdoors into the curve. So Seele proposes a multiple elliptic curves digital signature algorithm
(MECDSA), which is more secure and can avoid any backdoors. In MECDSA scheme, the user cannot only choose the number of elliptic curves according to practical security requirements, but also can customize the curve by editing its parameters.

The multiple elliptic curves digital signature algorithm serves four purposes in Seele. Firstly, the signature proves that the owner of the private key, who is by implication the owner of the funds, has authorized the spending of the funds. Secondly, the proof of authorization is undeniable. Thirdly, the signature proves that the transaction has not and cannot be modified by anyone after it has been signed. Fourthly, the signature scheme can avoid any backdoors in the ECDSA curve.

In section 2, we summarize existing elliptic curve digital signature algorithm (ECDSA). In section 3, we propose a multiple elliptic curves digital signature algorithm (MECDSA). In section 4, we analyze the performance of MECDSA from its validity, security and efficiency.

2 Elliptic Curve Digital Signature Algorithm (ECDSA)

An elliptic curve E over a finite prime field Fp is defined by the short Weierstrass equation,

\[ E : y^2 = x^3 + ax + b \]

where \( a, b \in Fp \), \( 4a^3 + 27b^2 \neq 0(modp) \), and \( p \) is a large prime. \( P \) is a randomly selected element on the elliptic curve \( E \), called the base point, whose order \( n \) is a large prime that makes \( nP = O \), where \( O \) is the zero element of the field \( E(Fp) \), \( n > 2^{160} \), \( n > 4\sqrt{p} \).

A signer chooses a random number \( d \in [1,n] \) as a private key, and computes public key \( Q = dP \).

2.1 ECDSA Signature Generation

A signer signs the message \( m \) according to the following steps:

Step 1: Compute \( e = H(m) \), where \( H \) is the secure hash algorithm.

Step 2: Select a random integer \( k \) from \([1, n - 1]\), and compute \( kP = (x, y) \).

Step 3: Compute \( r = x(modn) \). If \( r = 0 \), go back to step 2 and reselect \( k \).

Step 4: Compute \( s = k^{-1}(e + dr)(modn) \). If \( s = 0 \), go back to step 2 and reselect \( k \).

Then the signature on the message \( m \) is the pair \((r,s)\).

2.2 ECDSA Signature Verification

Anyone can check to see if the signature \((r,s)\) of the message \( m \) is valid by completing the following steps:

Step 1: Verify that \( r \) and \( s \) are integers in \([1, n - 1]\). If not, the signature is invalid.

Step 2: Compute \( e = H(m) \), where \( H \) is a same hash function used in signature generation.

Step 3: Compute \( w = s^{-1}(modn) \).
Step 4: Compute $u = ew(\text{mod} n), v = rw(\text{mod} n)$.
Step 5: Compute $R = uP + vQ = (x, y)$.
The signature is valid if $r = x(\text{mod} n)$, or otherwise it is invalid.
The signer can obviously operate the ECDSA $t$ times ($t$-ECDSA), and get the signature $(r_{t1}, s_{t1}, r_{t2}, s_{t2}, ..., r_{tn}, s_{tn})$ in $t$ elliptic curves, but this will make the length of the signature long. To fix this, Seele presents a secure multiple elliptic curves digital signature algorithm (MECDSA) in the next section, which can reduce the length of signature.

3 Multiple Elliptic Curves Digital Signature Algorithm

In this section, we propose a secure multiple elliptic curves digital signature algorithm (MECDSA), which can avoid any backdoors in the curve used by secp256k1. In this signature scheme, the user can not only choose the number of elliptic curves according to practical security requirements, but also can choose the curve by changing its parameters.

The parameters of MECDSA are same as ECDSA. There are $t$ elliptic curves $E_i: y^2 = x^3 + a_i x + b_i$ over finite prime field $\mathbb{F}_{p_i}$, $P_i$ is the base point whose order is $n_i$, where $i = 1, 2, ..., t$.

In MECDSA, the signer chooses the number of curve and the parameters of the curves. After this, the signer chooses $t$ random numbers $d_i \in [1, n_i]$ as a private key, and computes the public key $Q_i = d_iP_i$, where $i = 1, 2, ..., t$.

3.1 MECDSA Signature Generation

A signer signs the message $m$ according to the following steps (fig. 1):

Step 1: Compute $e = H(m)$, where $H$ is the secure hash algorithm.
Step 2: Select random integer $k_i \in [1, n_i - 1]$, computer $k_i P_i = (x_{i1}, y_{i1})$, where $i = 1, 2, ..., t$.
Step 3: Compute $r_i = x_i(\text{mod} n_i)$. If $r_i = 0$, go back to step 2 and reselect $k_i$, where $i = 1, 2, ..., t$.
Step 4: Compute $r = r_1 + r_2 + ... + r_t$. If $r = 0(\text{mod} n)$, return step 2, where $i = 1, 2, ..., t$.
Step 5: Compute $s_i = k_i^{-1}[e + d_i r](\text{mod} n_i)$. If $s_i = 0$, go back to step 2 and reselect $k_i$, where $i = 1, 2, ..., t$.
Then $(r, s_1, s_2, ..., s_t)$ is taken as the signature of the message $m$.

3.2 MECDSA Signature Verification

Anyone can check whether or not the signature $(r, s_1, s_2, ..., s_t)$ of $m$ is valid by completing the following steps:

Step 1: Verify that $r$ and $s_i$ are integers in $[1, n_i + n_2 + ... + n_t - t]$ and $[1, n_i - 1]$ respectively, where $i = 1, 2, ..., t$. If not, refuse the signature.
Step 2: Compute $e = H(m)$, where $H$ is a same hash function used in signature generation.
Step 3: Compute \( w_i = s_i^{-1}(mod n_i) \), where \( i = 1, 2, ..., t \).
Step 4: Compute \( u_i = e w_i (mod n_i) \), \( v_i = r w_i (mod n_i) \), where \( i = 1, 2, ..., t \).
Step 5: Compute \( R_i = u_i P_i + v_i Q_i = (x_i, y_i) \), where \( i = 1, 2, ..., t \).
Step 6: Compute \( r_i' = x_i (mod n_i) \), where \( i = 1, 2, ..., t \).
If \( r = r_1' + r_2' + \cdots + r_t' \), accept the signature, otherwise, refuse the signature.

\[ u_i = e w_i (mod n_i) = es_i^{-1}(mod n_i) = ek_i[e + d_ir]^{-1}(mod n_i) \]

4 Performance Analysis

In the section, we analyze the performance of MECDSA from different perspectives, including validity, security and efficiency.

4.1 Validity Analysis

According MECDSA signature generation and signature verification algorithm in MECDSA, we can get:
\[ v_i = rw_i \pmod{n_i} = rs_i^{-1} \pmod{n_i} = rk_i[e + d_i r]^{-1} \pmod{n_i} \]

So

\[ R_i = u_i P_i + v_i Q_i \]

\[ = [ek_i(e + d_i r)^{-1} \pmod{n_i}]P_i + [rk_i(e + d_i r)^{-1} \pmod{n_i}]Q_i \]

\[ = [ek_i(e + d_i r)^{-1} \pmod{n_i}]P_i + d_i [rk_i(e + d_i r)^{-1} \pmod{n_i}]P_i \]

\[ = k_i(e + d_i r)^{-1} \pmod{n_i} \]

\[ = (x_i, y_i) \]

So \( r_i' = x_i \mod n_i = r_i \) and \( r = r_1' + r_2' + \cdots + r_t' = r_1 + r_2 + \cdots + r_t \), where \( i = 1, 2, \ldots, t \). Therefore, we prove the correctness of MECDSA.

### 4.2 Security Analysis

The security of cryptosystems using elliptic curves hinges on the intractability of discrete logarithm problem in the algebraic system. Historically, this problem has received considerable attention from leading mathematicians and cryptologists around the world\(^{12-14}\). Unlike the case of the discrete logarithm problem in finite fields, there is no subexponential-time algorithm known for the elliptic curve discrete logarithm problem (ECDLP). The best algorithm known to date takes exponential time.

The security of the proposed MECDSA is based on the difficulty of solving the ECDLP, and is mainly faced with the following attacks. MECDSA security is presented by the following analysis of attacks:

1. The attacker wishes to obtain private key \( d_1, d_2, \ldots, d_t \) using all information that is available from the scheme. In this case, the attacker can compute \( r_1, r_2, \ldots, r_t \) according to the known public key \( Q_1, Q_2, \ldots, Q_t \), signature \( (r, s_1, s_2, \ldots, s_t) \) and message \( m \), and can further compute \( K_i \) where \( K_i = k_i P_i \). If an attacker wanted to obtain the private key, he would need to solved \( Q_i = d_i P_i \) or \( K_i = k_i P_i \) respectively for \( d_i \) and \( k_i \), which is clearly infeasible because the difficulty of solving the ECDLP, where \( i = 1, 2, \ldots, t \).

2. The attacker wishes to obtain the private key \( d_{i_1}, d_{i_2}, \ldots, d_{i_t} \) from a partial private key \( d_{j_1}, d_{j_2}, \ldots, d_{j_t} \), where \( t_1 + t_2 = t \). When the attacker can obtain \( d_{j_1}, d_{j_2}, \ldots, d_{j_t} \), because of the backdoors in \( t_2(t_2 < t) \) elliptic curves, it needs to solve \( Q_i = d_i P_i \) or \( K_i = k_i P_i \) respectively for \( d_i \) and \( k_i \), which is clearly infeasible because of the difficulty of solving the ECDLP, where \( i = i_1, i_2, \ldots, i_{t_i} \).

### 4.3 Efficiency Analysis

The computational cost of the proposed MECDSA is mainly determined by addition and multiplication operation in the elliptic curve and inverse operation in the finite prime field. Assume that \( t \) is the number of elliptic curves and \( l(n) \) is the length of the order \( n_i \) of the base point \( P_i \). Table 1 gives the efficiency analysis of MECDSA and t-ECDSA, including the computational cost and signature length.
Seele Yellow Paper Series No. 1

Table 1: Efficiency comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>Process</th>
<th>$P$</th>
<th>$Fq$</th>
<th>$EC$</th>
<th>Length of Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>t-ECDSA</td>
<td>Sig. Gen.</td>
<td>$t$</td>
<td>$2t$</td>
<td>$t$</td>
<td>$t$</td>
</tr>
<tr>
<td></td>
<td>Sig. Ver.</td>
<td>$0$</td>
<td>$2t$</td>
<td>$t$</td>
<td>$t$</td>
</tr>
<tr>
<td>MECDSA</td>
<td>Sig. Gen.</td>
<td>$2t-1$</td>
<td>$2t$</td>
<td>$t$</td>
<td>$0$</td>
</tr>
<tr>
<td></td>
<td>Sig. Ver.</td>
<td>$t-1$</td>
<td>$2t$</td>
<td>$t$</td>
<td>$t$</td>
</tr>
</tbody>
</table>

5 Conclusion

In order to avoid backdoor in the ECDSA curve of cryptocurrency systems, Seele proposes a multiple elliptic curves digital signature algorithm (MECDSA), which allows the signer to not only choose the number of elliptic curves, but also specify the parameters of each elliptical curve. Seele’s elliptic curves are not limited to the curve P-256 of national institute of standards and technology (NISt)\(^{15}\), SM2 of Chinese state cryptography administration\(^{16}\), secp256r111 and secp256k1 in standards for efficient cryptography, and others. Seele suggests that the systems based on blockchain should choose two elliptic curves considering the contradiction between security and efficiency.

References


Appendix

1 Curve P-256 of National Institute of Standards and Technology
For each prime $p$, a pseudo-random curve $E: y^2 = x^3 - 3x + b (mod p)$ of prime order $n$ is listed. The following parameters are given:
The prime modulus $p$:

\[
p = 115792089210356248762697446949407573530086143415290314195533631308867097853951
\]

The order $n$:

\[
n = 115792089210356248762697446949407573529996955224135760342422259061068512044369
\]

The coefficient $b$ (satisfying $b^2c = -27 mod p$):

\[
b = 5ac635d8a3a93e7b3ebd55769886bc651d06b0cc53b0f63bce3c3e27d2604b
\]

The base point $P$ in compressed form is:

\[
P = 02 \text{6b17d1f2e12c4247f8bec6e563a440f277037d812deb33a0f4a13945d898c296}
\]

and in uncompressed form is:

\[
P = 04 \text{6b17d1f2e12c4247f8bec6e563a440f277037d812deb33a0f4a13945d898c2964fe342e2fe1af7f9bf8ee7eb4a7c0f9e162bce33576b315ececb6406837bf51f5}
\]

The cofactor $h$:

\[
h = 01
\]

2 SM2 of Chinese State Cryptography Administration

The elliptic curve equation is $E: y^2 = x^3 + ax + b$, and the parameters of the equation are as following:
The prime modulus $p$:
3 secp256r1 of standards for efficient cryptography

The verifiably random elliptic curve domain parameters over \( F_p \) secp256r1 are specified by the extuple \( T = (p, a, b, G, n, h) \) where the finite field \( F_p \) is defined by:

\[
p = \text{ffffffff ffffffff ffffffff ffffffff 00000000 ffffffff 00000000 ffffffff} \]

The curve \( E: y^2 = x^3 + ax + b \) over \( F_p \) is defined by:

\[
a = \text{ffffffff ffffffff ffffffff ffffffff 00000000 ffffffff 00000000 ffffffff} \\
b = \text{28c9f9e9d9f5e344f5a9e4b f6509a7f39789f515abf92d8bcb4d14d940e93} \\
\]

The order \( n \):

\[
n = \text{ffffffff ffffffff ffffffff 7203df6b21605253b6f4093df54123} \\
\]

The base point \( P \) in compressed form is:

\[
P = \text{0232c4ae2c1f1981195f9904466a39c9948fe30bbf2f660be1715a4589334c74c7} \\
\]

and in uncompressed form is:

\[
P = \text{0432c4ae2c1f1981195f9904466a39c9948fe30bbf2f660be1715a4589334c74c7} \\
bc3736a2f4f6779c59bdcee36b692153d0a9877c62a474002df32e52139f0a0 \\
\]

The cofactor \( h \):

\[
h = 01 \\
\]

4 secp256k1 of standards for efficient cryptography

The elliptic curve domain parameters over \( F_p \) associated with a Koblitz curve secp256k1 are specified by the sextuple \( T = (p, a, b, G, n, h) \), where the finite field \( F_p \) is defined by:

\[
p = \text{ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff} \\
\]

The curve \( E: y^2 = x^3 + ax + b \) over \( F_p \) is defined by:

\[
a = 0 \\
b = 7 \\
\]

The base point \( P \) in compressed form is:
P = 02 79be667e f9dcbbac 55a06295 ce870b07 029bfcdb 2dce28d9 59f2815b 16f81798
and in uncompressed form is:
P = 04 79be667e f9dcbbac 55a06295 ce870b07 029bfcdb 2dce28d9 59f2815b 16f81798
483ada77 26a3c465 5da4fbfc 0e1108a8 fd17b448 a6855419 9c47d08f fb10d4b8
Finally the order \( n \) of \( G \) and the cofactor are:
\( n = ffffffff ffffffff ffffffff baaedce6 af48a03b bfd25e8c d0364141 \)
\( h = 01 \)