Implementation of Certificateless Signcryption based on Elliptic Curve Using Javascript

Abdul Wahid¹, Masahiro Mambo²

¹School of Natural Science and Technology, Kanazawa University, Kanazawa - Ishikawa, Japan
²Institute of Science and Engineering, Kanazawa University, Kanazawa - Ishikawa, Japan
wahidabd@gmail.com

ARTICLE INFO
Article history:
June 9, 2016
July 19, 2016
August 2, 2016

Keywords:
Certificateless Cryptography
Elliptic Curve Cryptography
Signcryption
Protocol
Javascript

ABSTRACT
Certificateless Cryptography (CLC) provides a new approach to addressing problems in encryption techniques. In the CLC, any string including identity can be used as a public key so that there is no need to use a certificate to link the identity to the public key. In addition, the CLC concept makes it possible to revoke keys with a certain time limit. Another emerging concept is signcryption which is a cryptographic system that combines the digital signature system and the encryption system into one logical step thus requiring a smaller computational cost compared to the regular signature-then-encryption systems. Elliptic Curve Cryptography (ECC) is a cryptography using elliptic curve which offers the same security as the RSA cryptosystem but with a smaller key size thus requiring less memory and computation. In this paper, we combine these three concepts to obtain all the advantages possessed by each concept without compromising system security, namely we construct an efficient Certificateless Signcryption (CLSC) scheme based on elliptic curve. In the implementation, we use client-based programming, javascript, which can reduce the workload of the server in order to achieve system security. We compare the computational cost, performance and ciphertext size of our scheme with some previous schemes and analyze the security properties of our scheme.

I. Introduction
Security protocol is one of the most important mechanisms in providing security of public networks because crucial information is hidden by this mechanism. In the design of security protocols, security issues and efficiency are a major concern. Concerning the security issues, we should consider at least the following 4 major security issues of networks, namely (1) Confidentiality: Confidentiality ensures that the data or information cannot be accessed by unauthorized users; (2) Integrity: Integrity ensures that the data or information cannot be modified during delivery; (3) Authentication: Authentication has to guarantee both of the authenticity of user and the authenticity of data. User authentication ensures that the user who can access the system is a true user, while data authentication ensures that the received data actually comes from the true sender; and (4) Non-repudiation: Non-repudiation ensures that user cannot deny the data or information which she or he has sent.

Message encryption schemes and digital signature schemes are cryptographic tools used for confidentiality, integrity, authentication and non-repudiation [1]. Confidentiality can be achieved by the encryption schemes. Integrity, authentication, and non-repudiation can be achieved by the digital signature schemes.

Public-key cryptosystems are one of cryptographic protocols widely used nowadays. In the systems, each user chooses their own private key to calculate its corresponding public key. In the use of the public key, one should be able to know whose public key one is using. A certificate issued by a certification authority shows this connection between the public key and user’s identity. Thus, the systems require Public Key Infrastructure (PKI) whose bandwidth consumption and maintenance cost are usually high.

In order to reduce the burden caused by the PKI, Identity-Based Cryptography (IBC) was invented. This concept was first discovered by Adi Shamir [2] as well as Tatsuaki Okamoto [3] and its secure and efficient technique was recently discovered by Boneh and Franklin in 2001 [4]. In the IBC, the identity of a user such as name, ID number, email and telephone number serves as the public key, and there is no longer any doubt about the authenticity of the public key. Therefore, PKI can be eliminated. In addition, this technique easily

Website: http://ijcandi.org E : haviuiddin.ijcandi@gmail.com
allows one to set the validity period of the keys without requiring an additional key revocation mechanism. Despite these advantages, the private key of the IBC is generated from user’s identity by the Private Key Generator (PKG) and a key escrow problem inherently exists in the IBC.

Certificateless cryptosystem (CLC) [5] which is a variant of the IBC is intended to prevent the key escrow problem. In the ordinary IBC, keys are generated by key generation center (KGC) which is given a complete power and is fully trusted. In contrast, the CLC considers a compromised KGC. To prevent a complete breakdown of the system under the compromised KGC, the key generation process is split between the KGC and each user. First, the user generates a random value which is never revealed to anyone, including the KGC, as in the public-key cryptosystem. Then the KGC generates a private key based on the identity of the user, where the private key is now a partial private key of the system and sent to the user. Afterwards, all cryptographic operations by the user are performed by using a complete private key which involves both the partial private key and the user's random secret value. Therefore, the best features of the IBC and the public-key cryptosystem are combined.

On the other hand, there are several approaches such as signcryption and Elliptic Curve Cryptography which focus more on the efficiency. Signcryption introduced by Yuliang Zheng in 1997 [6] is a technique in which the functions of digital signature and encryption are achieved in just one logical step. It is effective in reducing computational cost and communication overhead compared to the signature-then-encryption technique. So far, there have been many studies on the signcryption. Elliptic Curve Cryptography (ECC) is based on the algebraic structure of elliptic curve over a finite field [7]. ECC has become a very important part in cryptography because of its high performance by a shorter key with the same level of security as other public key techniques. The Elliptic Curve-Discrete Logarithm Problem (EC-DLP) and Elliptic Curve-Computational Diffie Hellman Problem (EC-CDHP) can be defined in the ECC, and one can construct security protocols based on these problems [8], [9].

The efficiency of the CLC can be improved by applying these approaches. Actually, in 2008 Barbosa and Farshim [10] proposed a Certificateless Signcryption (CLSC) scheme which combines the CLC and the signcryption scheme. So far, several CLSC schemes have been proposed. However, most of the schemes are based on bilinear pairings. The time needed for running bilinear pairings is about 10 times slower than that needed for running the finite field exponentiation algorithm [11]. In order to overcome such an efficiency problem, more efficient CLSCs based on the finite field exponentiation have been offered by [11], [12] without using bilinear pairings. Even so, the computation of finite field exponentiations as well as bilinear pairings need large integer values for keeping the complexity of problems related to them. Under limited resource environments such as low memory and power consumption with constrained bandwidth, we need to find a more efficient construction.

In this paper, we further apply the ECC to the framework of the CLSC and construct a more efficient CLSC. To this end, we do not modify existing CLSCs but construct a new certificateless signcryption scheme based on elliptic curve cryptography from scratch. In the design of our construction, we pay attention to certificateless hybrid signcryption schemes [8], [9] explained in the next section. The proposed scheme provides confidentiality, authentication, integrity, non-repudiation as well as unforgeability and forward secrecy. Since it is one of CLSC schemes, it solves the certificate management problem and the key escrow problem. By the evaluation of our CLSC scheme via the implementation and other analysis, we shows that our CLSC scheme has a better efficiency than existing schemes in terms of the ciphertext size and the execution time of key generation, signcryption, and unsigncryption phases.

II. Related Work

In 1997, Zheng [6] offered a primitive cryptographic technique that carries out both digital signature and message encryption functions simultaneously which he called signcryption. The cost of signcryption is much smaller than the signature-then-encryption model. There are several signcryption schemes [13]–[16] that have been proposed since 1997. One of them is a signcryption scheme proposed by Zheng and Imai [16] that utilizes the hardness of EC-DLP. They proved that this signcryption scheme has an efficiency of approximately 58% of the computational cost and 40% of communication cost the signature-then-encryption scheme based on an elliptic curve.

In the signcryption scheme, the user's public key is a random element of some group. Therefore, this scheme does not provide user authentication itself because the random group element cannot define the identity of the user. This problem can be solved by the use of certificates, where there is a CA that provides a setting in which the public key is bound to the identity of each user. This system is known as the PKI. However, PKI has difficulties in the manufacture, storage, and distribution of its digital certificates.

To overcome these issues, Shamir [2] introduced the concept of Identity-based cryptography. The main idea is that the identity information such as name, e-mail, telephone number, or identity number of each user is used as its public key and not derived from certificates issued by the CA. In Identity-Based Cryptography,
users can perform secure communications without the need to distribute public key certificates, without the need to store a public key directory and without the participation of online Public Key Generator (PKG). In addition to that already offered by Chen and Malone-Lee in 2005 [14], there are already some identity-based signcryption schemes that have been offered [14], [17], [18]. Unfortunately, their work still has the disadvantages that key escrow problem such that PKG holding all secret keys of the system has to be fully trusted.

Certificateless cryptography [5] is proposed to solve this key escrow problem. As a variant of Identity-Based Cryptography, certificateless cryptography uses a user’s identity as a public key and the KGC generates a partial private key of the user from th identity. Another private key is created by the user. KGC is not fully trusted because it does not know the whole of user’s private key.

The certificateless schemes that uses the elliptic curve approach has been proposed in papers [8], [9]. These papers propose a certificateless hybrid signcryption scheme, called CLSC-TKEM (CLSC-tag Key Encapsulation Mechanism), to encapsulate keys that are shared by the sender and the recipient. The concept is that the sender will create a session key using a random value and the recipient's public key. The sender then sends out a public value that has a relation with the random value along with the digital signature to the recipient. The receiver then calculates the session key by using the public key along with the receiver’s private key. We adopt this concept with slight modifications and also incorporate the concept of signcryption to construct a Certificateless Signcryption protocol scheme based on Elliptic Curve.

III. Certificateless Signcryption

In this section, we offer the use of Certificateless Signcryption (CLSC) based on elliptic curve cryptography without pairing function. A scheme based on the concept of Barbosa-Farshim scheme [10] which can accept ID input and message of any length as well as use a secure one-time symmetric key encryption scheme and collision resistance hash function.

A. Formal model CLSC

According Barbosa-Farshim scheme, certificateless signcryption is separated to six-tuple of probabilistic polynomial-time algorithms. Four of these algorithms are corresponding to key management operations, while two algorithms are identical to signcryption and unsigncryption algorithms. The detail algorithms are the following steps:

1. Setup(1^k). This is a global set-up algorithm, which takes as input the security parameter 1^k and returns the KGC’s secret key msk and global parameters pars including a master public key P_{pub} and descriptions of message space M(pars), ciphertext space C(pars) and randomness space R_{am}(pars). This algorithm is executed by the KGC, which publishes pars.

2. Extract-PPK(ID, msk, pars). An algorithm which takes as input msk, pars and an identifier string ID ∈ {0, 1}^* representing a user’s identity, and returns a partial private key d. This algorithm is run by the KGC, after verifying the user’s identity.

3. Gen-SV(ID, pars) An algorithm which takes an identity and the public parameters and outputs a secret value x and a public key P. This algorithm is run by a user to obtain a public value and a secret value which can be used to construct a full private key. The public key is published without certification.

4. Set-SK(d, x, pars). A deterministic algorithm which takes as input a partial secret key d and a secret value x and returns the full private key SK. Again, this algorithm is run by a user to construct the full private key.

The signcryption and Unsigncryption algorithms are as follows:

5. SC(m, SK_A, ID_A, PK_A, ID_B, PK_B, pars, l). This is the signcryption algorithm. On input of a message m ∈ M(pars), sender’s full private key SK_A, identity ID_A and public key PK_A, the receiver’s identity ID_B and public key PK_B, the global parameters pars and possibly some randomness l ∈ R_{am}(pars), this algorithm outputs a ciphertext c ∈ C(pars) or an error symbol ⊥.

6. USC(c, SK_B, ID_B, PK_B, ID_A, PK_A, pars). The deterministic Unsigncryption algorithm. On input of a ciphertext c, receiver’s full private key SK_B, identity ID_B and public key PK_B, the sender’s identity ID_A and public key PK_A and the global parameters params, this algorithm outputs a plaintext m or a failure symbol ⊥.

B. Proposed CLSC based on Elliptic Curve

This scheme modifies the Elliptic Curve Cryptography based Certificateless Hybrid Encapsulation Key scheme without Pairing [8] and the eCLSC-TKEM [9] to obtain all the advantages of both techniques. The scheme consists of three parties, namely Key Generator Center (KGC), Sender and Receiver. KGC’s function is to calculate the partial private key and public key pairs for all users when they first join the system. This
process is performed only once at the beginning and can be done offline. For the process, we divided this scheme into eight phases, setup-parameter which is run by KGC, Set Secret value which is run by each user, Set User Pseudonym which is run by each user or can be run by special server, Partial private key extract which is run by KGC, Set Private Key, Set Public Key which are run by each user, Signcrypt which is run by sender and Unsigncrypt which is run by receiver. In the initial phase, the system will select and publish all elliptic curve security parameters for all users that exist in the system. The following are the details of the process of the system.

1. Set-Up Parameter: It is run by the KGC. KGC selects and publishes system security parameters which are given below.
   - \( F_q \): Finite field of large prime number \( q \)
   - \((a,b)\): 2 elliptic curve value \( < q, \) satisfy to \( 4a^3+27b^2 \neq 0 \) and \( q \neq 0 \)
   - \( E/F_q \): elliptic curve over finite field, satisfy to \( q: y^2=x^3+ax+b mod q \)
   - \( G_q \): a generator of EC
   - \( O \): infinity point of EC, \( n \) is the order of \( F \) satisfy to \( n.G=O \)
   - \( Hash\) function \( h_0 = \{ 0,1 \}^r \times Gq^2 \rightarrow Z^*_q \)
   - \( Hash\) function \( h_1 = \{ 0,1 \}^r \times Gq^2 \rightarrow Z^*_q \)
   - \( Hash\) function \( h_2 = Gq^2 \times \{ 0,1 \}^r \times Gq^2 \rightarrow Z^*_q \)
   - After that, PKG chooses integer \( msk \in Z^*_q \) as the master secret key and calculate \( P_{pub} = msk.G_q \) as master public key.
   - PKG then publishes the public parameters \((F_q, E/F_q, G_q, h_0, h_1, h_2, P_{pub})\) but keeps secret the msk.

2. Set secret value: It is run by each user. User \( i \) with \( ID_i \) performs the following steps.
   - Chooses randomly \( x_i \in Z^*_q \).
   - Computes public key \( P_i = x_i.P \)

3. Partial private key extract: It is run by KGC. Here, KGC produce the partial private key of each user based on their identity. The KGC processes the user \( i \) with \( ID_i \) in the following step:
   - Chooses randomly \( r_i \in Z_q^* \) and computes \( R_i = r_i.P \)
   - Computes \( d_i = r_i + msk.h_0(ID_i, R_i, P_i) mod q \)
   - Sends to user \( = <R_i, d_i> \) in a secure channel
   - User validates \( d_i.P = R_i + h_0(ID_i, R_i, P_i).P_{pub} \)

4. Set Private Key: It is run by each user. User \( i \) with identity \( ID_i \) perform to set a private key \( SK_i = <d_i, x_i> \).

5. Set Public Key: It is run by each user. User \( i \) with identity \( ID_i \) perform to set a public key \( P_{K_i} = <R_i, P_i> \).

6. Signcryption
   Alice is the sender. She wants to send message ‘m’ to Bob as the receiver with identity \( ID_B \), and a pair public key \((R_B, P_B)\). Alice chooses \( l_A \in \mathbb{R} [1,2,...(q-1)]\), then Alice computes:
   - \( U = I_A.P \)
   - \( Y_U = R_B + h_0(ID_B, R_B, P_B).P_{pub} \)
   - \( SK = h_2(I_A, (Y_U + P_B), U, ID_B, R_B, P_B) \)
   - \( C = E_{SK}(m, ID_A) \)
   - \( s = (d_A + l_A.h_1(m, ID_A) + x_A.h_1(m, ID_A)).mod q \)
   - Alice sends to Bob chipertext = \((C, U, s)\)

7. Unsigncryption
   Bob is the receiver. He receives chipertext \((C’, U’, s’)\) from Alice. Bob computes:
   - \( SK = h_2(d_A + x_A).U, U, ID_B, R_B, P_B) \)
   - \( (m, ID_A) = D_{SK}(C’) \)
   - \( Y_A = R_A + h_0(ID_A, R_A, P_A).P_{pub} \)
   - Verify: Accept if \( s’.P = Y_A + U.h_1(m, ID_A) + P_A. h_1(m, ID_A) \) is hold.

IV. Implementation in Javascript
   In this section, we have implemented the simulation of our proposed scheme using JavaScript to test its truth. All of the security parameters use large numbers to protect the system from various types of attacks.
   Here, we applied three types of elliptic curve function that are 512 bits brainpoolP512r1, 512 bits brainpoolP512r1, and 256 bits brainpoolP256. These values are taken from https://github.com/spruegel/Fast-ECDSA-in-JavaScript/blob/master/jsbn/sec_mod.js by spruegel. Table 1 shows the value of each type of elliptic curve function.
As for the operating point on its elliptic curve, we used a modification of the ec.js file belonging to Tom Wu. To increase the speed, we replaced the multiplication point simultaneously with a faster-windowed method approach. In addition, for the speed in the processing of the extract key generation in this simulation, we made a key generation() function in the different js file. This file stores the master secret key owned by KGC that must remain confidential. The following figure is the functions that have been mentioned.

<table>
<thead>
<tr>
<th>Elliptic Curve types</th>
<th>Var</th>
<th>Value ( Hexadecimal )</th>
</tr>
</thead>
<tbody>
<tr>
<td>brainpoolP256r1()</td>
<td>p</td>
<td>a9b57ba1ee9bc3e660a99d983d87263e3b62d32620282013481d1f6e5377</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>7da50975f6c3057eff67530417affce7fb8055c126dc5e6e944b4430b5d9</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>26dc5e6ec94af4b43f30b5d9b7cb9f58416295cf7e16e6cdcf18ff0c7fe</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>a9b57ba1ee9bc3e660a99d983d8718c397a3a3b51a67901e082974856a7</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>8bd2a2eb9cb7e57cbe4b48f28e817a7a9bdc27e1e3bd23c23a4453bd9ace3262</td>
</tr>
<tr>
<td></td>
<td>msk</td>
<td>3eaaf7a0b20e5d53503853610a6a9a19b11b4d945f6498a4d4084f9f10a59016</td>
</tr>
<tr>
<td></td>
<td>P_pub</td>
<td>9a788b76f711a4a4eb7d19d9a0c4cefe80358c88a582123919bf266b1ab0dd38,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>67b43aaf7b1a893520df1e6f34f5907856acce1506d16325e853ca40be87c3</td>
</tr>
<tr>
<td>brainpoolP512r1()</td>
<td>p</td>
<td>add9d8b79e9c483fb4d4ea3e39cf0fcb703b63b39c20dcd6639caca730380717d</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>873803318603ba89e2327145ac234c594cbbd8d3df1610a83441ee98643c</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>3d9f1610a83441ee98643c2ed5d5a8253aa10a2ef1c9f89a8c885f71117af</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>aadd9d8bbde9c48bb3f4de6ae3339cf0c7cb083db3bc3b92d0ed6639c9ca730380717d</td>
</tr>
<tr>
<td></td>
<td>msk</td>
<td>2332760a01c91a7ad3b1ab9f8b476760cea401343edadb154267d4f3ea90</td>
</tr>
<tr>
<td></td>
<td>P_pub</td>
<td>46fa83ae0edbb3e1c197e8f5b7a03b0f4ef4f4df3ec6bba873f980721d3ea95a7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8d82a8af10fe949ae674df237ebf63f3edc2e130d74ae2c7d680f3157c7e,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>72a27a9fcd21d3b3bf4f3f4344ad0ad85aaf380fca4cb24b3b7260ce2f659711825e</td>
</tr>
<tr>
<td></td>
<td></td>
<td>61b35987173c3e53e56970a70af7aab2d2635ob5a78a94f3d524500edbd5</td>
</tr>
<tr>
<td>brainpoolP512t1()</td>
<td>p</td>
<td>aadd9d87bde9c48bb3f4de6ae3339cf0c7cb083db3bc3b92d0ed6639c9ca730380717d</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>72a27a9fcd21d3b3bf4f3f4344ad0ad85aaf380fca4cb24b3b7260ce2f659711825e</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>57cbb89c941fcb76e1890e4688e4aea3217f00ce4b9481527897504be3e46a8b</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>aadd9d87bde9c48bb3f4de6ae3339cf0c7cb083db3bc3b92d0ed6639c9ca730380717d</td>
</tr>
<tr>
<td></td>
<td>msk</td>
<td>80aa4d1b79b4a69bb72a8a03f4dbf86887498f720b221a0005b5881811112b</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>640952c57e12b788717b8c1ba06bcb2a6fe8a858424585c6dd9e1b758d9c3031d</td>
</tr>
<tr>
<td></td>
<td></td>
<td>82ba27153cdbe3aa4977a7ad943af64af3f25e606b51ba2966a9053da5,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5b535ad95f5a50fa2c82932763c84ace1eb483019b7163401315195a1933ce</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9932184e2be1f67d2d2a862d6730706cecf96dbb8ace198b61e008b332</td>
</tr>
<tr>
<td></td>
<td>msk</td>
<td>80a82789df3195986f4a6096b5cfd25d543e53e91af12d6e862d1109f557</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>8a2789df3195986f4a6096b5cfd25d543e53e91af12d6e862d1109f557</td>
</tr>
</tbody>
</table>
For the one-way hash function, we have used SHA256 which takes the input of any length and then generates a 256-bit output. The symmetric encryption function used in the signcryption process is the AES algorithm. We took both of these functions from cryptoJS, JavaScript implementations of standard and secure cryptographic algorithms from https://code.google.com/p/crypto-js/. Figure 2 and figure 3 show the snapshot of our certificateless signcryption output.

```javascript
var extract = {
    keygeneration: function (ID,x,r)
    {
        var ecparams = getSECCurveByName(usedCurve);
        var q = ecparams.getN();
        var w = 4;
        var curve = get_curve();
        var P = new ECPointFp(curve);
        P = ecpComb2Mult(x,PA,w,GLOBAL_precomp,q.bitLength());
        var R = new ECPointFp(curve);
        R = ecpComb2Mult(r,RA,w,GLOBAL_precomp,q.bitLength());
        var hash = Crypto.SHA256(ID.concat(RA.getX()).concat(R.getY()).concat(P.getY()).concat(P.getY()));
        var hash1 = Crypto.util.hexToBytes(hash);
        var e = BigInteger.fromByteArrayUnsigned(hash1).mod(q);
        var msk = new BigInteger("4b94caecbc45dda458c029536dacf6da1d06dea31d7f2d6ab358d56787c63b966dc58d19479b06b6c69f09105c291f8ad6a18abeb968d1d7b7a25455eebc4", 16);
        var x = e.multiply(msk).mod(q);
        var sk = r.add(x).mod(q).toString(16);
        return [sk,R,P];
    }
};
```

Fig. 1. Key Generation function in javascript

Fig. 2. Snapshot of Key Generation Result
V. Analysis

In this section, we evaluate the correctness of the proposed Certificateless Signcryption scheme. Furthermore, we present a brief discussion about the security aspects of the proposed scheme. In addition, we offer the efficiency analysis in computational cost and speed performance after implementation.

A. Formula Correctness

The equation $SK = h_2(l_\lambda (Y_B + P_B), U, ID_B, R_B, P_B)$ in the Signcryption side and $SK = h_2((d_B + x_B).U, U, ID_B, R_B, P_B)$ in the Unsigncryption side should be same. So equation $l_\lambda (Y_B + P_B)$ should be same with equation $(d_B + x_B).U$.

- While $U = l_\lambda .P$
  
  $d_B = r_B + \text{msk}.h_0(ID_B, R_B, P_B)$
  
  $Y_B = R_B + h_0(ID_B, R_B, P_B).P_{pub}$

- Then
  
  $(d_B + x_B).U = (r_B + \text{msk}.h_0(ID_B, R_B, P_B) + x_B).l_\lambda .P$
  
  $= l_\lambda (r_B, P + \text{msk}.h_0(ID_B, R_B, P_B).P_{pub} + P_B)$
  
  $= l_\lambda (Y_B + P_B)$

Then, for the formula $s.P = Y_\lambda + U.h_1(m, ID_\lambda) + P.x.h_1(m, ID_\lambda)$ should be hold. It is because,

- While $d_\lambda .P = (r_\lambda + \text{msk}.h_0(ID_\lambda, R_\lambda, P_\lambda)).P$
  
  $= r_\lambda .P + h_0(ID_\lambda, R_\lambda, P_\lambda).\text{msk}.P$
  
  $= R_\lambda + h_0(ID_\lambda, R_\lambda, P_\lambda).P_{pub} = Y_\lambda$

- Then
  
  $s.P = (d_\lambda + l_\lambda.h_1(m, ID_\lambda) + x_\lambda.h_1(m, ID_\lambda)).P$
  
  $= d_\lambda .P + l_\lambda.P.h_1(m, ID_\lambda) + x_\lambda.P.h_1(m, ID_\lambda)$
  
  $= Y_\lambda + U.h_1(m, ID_\lambda) + P_\lambda.h_1(m, ID_\lambda)$

B. Security Analysis

A pair public and private key security relies on Elliptic curve discrete logarithm problem (ECDLP). Partial public key $Y_\lambda = R_\lambda + h(ID_\lambda, R_\lambda, P_\lambda).P_{pub} = d_\lambda .P$, where $P_\lambda, R_\lambda, P_{pub}$ and $P$ is a point on the elliptic curve over finite field and $d_\lambda$ is a quite large integer value. If partial private key $d_\lambda$ and $P$ are given, it will be “easy” to compute partial public key $Y_\lambda$. However if the ones given are partial public key $Y_\lambda$ and $P$, it will be “hard” to find partial private key $d_\lambda$. On the same way for the other public key $P_\lambda$, it comes from a random secret value $x_\lambda (P_\lambda = x_\lambda .P)$. If secret value $x_\lambda$ and $P$ are given, it will be “easy” to compute public key $P_\lambda$. However if the ones given are public key $P_\lambda$ and $P$, it will be “hard” to find secret value $x_\lambda$. 

Wahid & Mambo (Implementation of Certificateless Signcryption based on Elliptic Curve Using Javascript)
Confidentiality
If the attacker tries to obtain the original message from the ciphertext, he has to know the keys SK. There are two ways to obtain the key by the attacker:

\[\text{SK} = h_2((d_b + x_b).U, U, \text{ID}_b, \text{R}_b, P_b)\]

If the attacker tries to derive the key SK from the above equation, he has to find out the \(l_A\) random value. In this case, it is infeasible to solve the SK key value from the above equation because \(l_A\) is obtained randomly and only used once.

\[\text{SK} = h_2((d_b + x_b).U, U, \text{ID}_b, \text{R}_b, P_b)\]

It is also possible to derive the key from the above equation. However, the attacker has to obtain \(d_b\) and \(x_b\) because they are required to obtain SK. It is nearly impossible to obtain the SK from this second equation because \(d_b\) and \(x_b\) are the receiver’s private key which is known only by the receiver. In addition, because of the hardness of ECDLP, it is difficult to calculate \(x_A\) and \(d_A\) from the equation \(P_A = x_A.P\) and \(Y_A = d_A.P\) even if \(P_A, Y_A\), and \(P\) are known.

Authentication
User authentication
The receiver uses the sender’s identity and public key \((\text{ID}_A, \text{R}_A, P_A)\) and received digital signature \((s)\) to verify sender authentication. The sender signs in with their private key \((d_A, x_A)\). So in this scheme, the receiver can authenticate the identity of the sender.

Data authentication
The sender signs data \((m)\) with her private key \((d_A, x_A)\),
\[s = (d_A + l_A.h_1(m, \text{ID}_A) + x_A.h_1(m, \text{ID}_A)).\text{mod } q\]
and send it to receiver. Then the receiver verifies the received data \((m)\) using the received signature \((s)\). If \(s.P = Y_A + U.h_1(m, \text{ID}_A) + P_A.h_1(m, \text{ID}_A)\) is hold, it means that the data actually comes from the true sender.

Integrity
The receiver can verify whether the ciphertext was tampered or not at the time of transmission using the following equation.
\[s.P = Y_A + U.h_1(m, \text{ID}_A) + P_A.h_1(m, \text{ID}_A)\]
If the attacker changed the ciphertext \(c\) to \(c_1\), then the received original message should also change from \(m\) to \(m_1\). As a result, during verification, the computed digital signature of \(m_1\) will not be the same as the digital signature of \((m)\) sent by the sender to the receiver. Therefore, this scheme provides integrity.

Unforgeability
Unforgeability ensures that the attacker cannot create a valid ciphertext. In this scheme, the attacker cannot create a valid \((C, U, s)\) without the private key of the sender. If an attacker forged a valid \((C', U', s')\) from the previous \((C, U, s)\), then \((C', U', s')\) has to satisfy the equation
\[\text{SK} = h_2((Y_B + P_b).U, \text{ID}_b, \text{R}_b, P_b)\]
equation and \(s = (d_A + l_A.h_1(m, \text{ID}_A) + x_A.h_1(m, \text{ID}_A)).\text{mod } q\). It is impossible to achieve without knowing \(l_A, d_A\) and \(x_A\).

Non-repudiation
In this scheme, the receiver knows from the SK equation whether the original message was sent by the sender or not. The receiver can verify because the sender signs with his private key in the equation.
\[s = (d_A + l_A.h_1(m, \text{ID}_A) + x_A.h_1(m, \text{ID}_A)).\text{mod } q\]
equation. Thus it provides non-repudiation.

Forward Secrecy
This scheme ensures that even though the sender’s private key is obtained, the attacker cannot recover original message \(m\) from the ciphertext \((C, U, s)\). If the attacker tries to derive plaintext \(m\), he must decrypt its ciphertext using secret key SK using random value \(l_A\) or secret key of the receiver. Therefore, our scheme provides forward secrecy.

Table 2. gives a comparison of security attributes and features between our proposed protocol scheme and others scheme.

C. Computational Cost Analysis
In this section, the time complexity of the proposed scheme is evaluated. Table 3 gives a comparison between the computational costs of our proposed scheme and those of the others scheme, in which the computational costs of verification and symmetric encryption are neglected. We have used some notation to define a number of operation in table 3 are give below.
- Exp = Modul exponentiation operation
- Div = modular division operation
- Mul = modular multiplication operation
- Add = modular addition operation
- Ecadd = Elliptic Curve point addition operation
- Ecadd = Elliptic Curve point addition operation
- Hash = One way hash function
Table 2. Comparison of security properties of certificateless signcryption schemes and their variants

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Confidentiality</th>
<th>Authentication</th>
<th>Integrity</th>
<th>Unforgeability</th>
<th>Non-Repudiation</th>
<th>Forward Secrecy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zheng[6]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ZI [16]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>WNPZ [12]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>XX [11]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Unknown</td>
</tr>
<tr>
<td>SB [8]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WSB [9]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Ours</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Yes/No: Feature shown in the first row is/is not held.
unknown: It is unknown whether the security property shown in the top is achieved.

Besides the computational cost based on the mathematic operation, we evaluate the performance of several processes of our certificateless signcryption scheme by implementation. We make comparison of our scheme with only two existing CLSC schemes (SB [8] and WSB [9]) which are also based on elliptic curve. We do not compare our scheme with the other existing schemes because they use bilinear pairings or finite field exponentiation technique which have slower computation than elliptic curve computation. Figure 4 shows performances in execution time of each scheme in milisecond while table 4 shows the comparison of the size of ciphertexts transmitted from sender to receiver.

Based on the figure 4, we can see that our protocol performance is faster than SB [8] scheme and almost same speed with WSB [9] scheme. In table 4, we can see that our protocol has a shorter ciphertext size than WSB [9] scheme and same with SB [8] scheme. It means that the performance of our protocol is better than two other schemes.

Table 3. Computational costs of different schemes

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Type</th>
<th>Participant</th>
<th>Exp</th>
<th>Div</th>
<th>Mul</th>
<th>Add</th>
<th>ECMult</th>
<th>ECAdd</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Receiver</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>ZI [16]</td>
<td>SCa</td>
<td>Sender</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receiver</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>WNPZ [12]</td>
<td>CLSCb</td>
<td>Sender</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receiver</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>XX [11]</td>
<td>CLSCb</td>
<td>Sender</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receiver</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>SB [8]</td>
<td>CLSC-TKEMc</td>
<td>Sender</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receiver</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>WSB [9]</td>
<td>CLSC-TKEMc</td>
<td>Sender</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receiver</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Ours</td>
<td>CLSCb</td>
<td>Sender</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receiver</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

aSigncryption, bCertificateless Signcryption, cCertificateless Signcryption-Tag Key Encapsulation Mechanism.

Table 4. Ciphertext size comparison

<table>
<thead>
<tr>
<th>EC-CLSC Schemes</th>
<th>Ciphertext Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB [8]</td>
<td>n_q + n_id + n_id + m</td>
</tr>
<tr>
<td>WSB [9]</td>
<td>n_q + 2n_id + n_id + m</td>
</tr>
<tr>
<td>Ours</td>
<td>n_q + n_id + n_id + m</td>
</tr>
</tbody>
</table>

n_q: The number of bits required to represent an element of F_q.
n_id: The number of bits required to represent an element of point EC.

m: The number of bits required to represent an identity.

Wahid & Mambo (Implementation of Certificateless Signcryption based on Elliptic Curve Using Javascript)
We executed using windows 64-bit operating system, processor intel(R) Core(TM) i7-3770 CPU @3.40 GHz and memory (RAM) 16.0 GB. From that tables, we can see that bit length of the used elliptic curve influences of the speed of the system because has relation to the mathematic operation. While the Identity length does not effect to the speed. It is because one-way hash function will execute in the same way from different input length to produce same output length.

VI. Conclusion

In this paper, we have implemented Certificateless Signcryption based on the elliptic curve. In the sense of the type of discussion given in sect. V.B, our scheme meets all of the basic security needs such as message authentication, integrity, unforgeability, non-repudiation and forward secrecy. This scheme is implemented using javascript and utilizes several existing libraries such as cryptoJS, sec_mod.js from spruegel and ec.js owned by Tom Wu. In the scalar multiplication operation with the point on the elliptic curve, we have used the windowed method approach which is faster than the double-and-add approach because it uses less point summation (which in practice is slower than doubling). Our scheme is more efficient than the previous schemes because our CLSC based on elliptic curve which is more efficient than bilinear pairings and finite field exponentiations used in the previous CLSC schemes. Besides, our scheme offers shorter ciphertext size than previous CLSC schemes.

Acknowledgement

A part of this research is supported by JSPS A3 Foresight Program. The first author is supported by Directorate General of Resources for Science, Technology and Higher Education (DG-RSTHE) of Indonesia or DIKTI. We would like to appreciate their supports.

References


