Implementation of Certificateless Signcryption based on Elliptic Curve Using Javascript

Abdul Wahid[a,1], Masahiro Mambo[b]

[a] *School of Natural Science and Technology, Kanazawa University, Kanazawa - Ishikawa, Japan*
[b]*Institute of Science and Engineering, Kanazawa University, Kanazawa - Ishikawa, Japan*
[1] *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) Authenticaton: 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 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 certificateles 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 consruct 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 algoritms are the following steps:

1. Setup(1^{κ}). This is a global set-up algorithm, which takes as input the security parameter 1^{κ} 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 Ram(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 $\in \{0, 1\} *$ 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 $\in$ 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 $\in$ Ram(pars), this algorithm outputs a ciphertext c $\in$ C(pars) or an error symbol $\perp$.
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 $\perp$.

B. Proposed CLSC based on Elliptic Curve

This scheme modifies the Elliptic Curve Cryptography based Certificaless 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 phase 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: $y^2=x^3+ax+b \bmod q$
   - G_q = a generator of EC
   - O = infinity point of EC, n is the order of F satisfy to n.G = O
   - Hash function $h_0 = \{0, 1\}^* \times Gq^2 \rightarrow Z_q^*$
   - Hash function $h_1 = \{0, 1\}^{*2} \times Gq^2 \rightarrow Z_q^*$
   - Hash function $h_2 = Gq^2 \times \{0, 1\}^* \times Gq^2 \rightarrow Z_q^*$
   - 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 every 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 $d_i = r_i + msk.h_0(ID_i,R_i,P_i) \bmod q$
   - 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 $Pk_i = <R_i, P_i>$.
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 $l_A \in_R [1,2,\ldots(q-1)]$, then Alice computes :
   - $U = l_A.P$
   - $Y_B = R_B + h_0(ID_B,R_B,P_B).P_{pub}$
   - $SK = h_2(l_A.(Y_B+P_B),U,ID_B,R_B,P_B)$
   - $C = E_{SK}(m,ID_A)$
   - $s = (d_A + l_A.h_1(m,ID_A) + x_A.h_1(m,ID_A)).\bmod q$
   - Alice sends to Bob chipertext = (C,U,s)
7. Unsigncryption
   Bob is the receiver. He receives chipertext (C',U',s') from Alice. Bob computes:
   - $SK = h_2((d_B+x_B).U,U,ID_B,R_B,P_B)$
   - $(m,ID_A) = D_{SK}(C')$
   - $Y_A = R_A + h_0(ID_A,R_A,P_A).P_{pub}$
   - Verify: Accept if $s'.P = Y_A + U.h_1(m,ID_A) + P_A.\ h1(m,ID_A)$ 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 brainpoolP512t1, 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.

Table 1. Elliptic curve values of our implementation

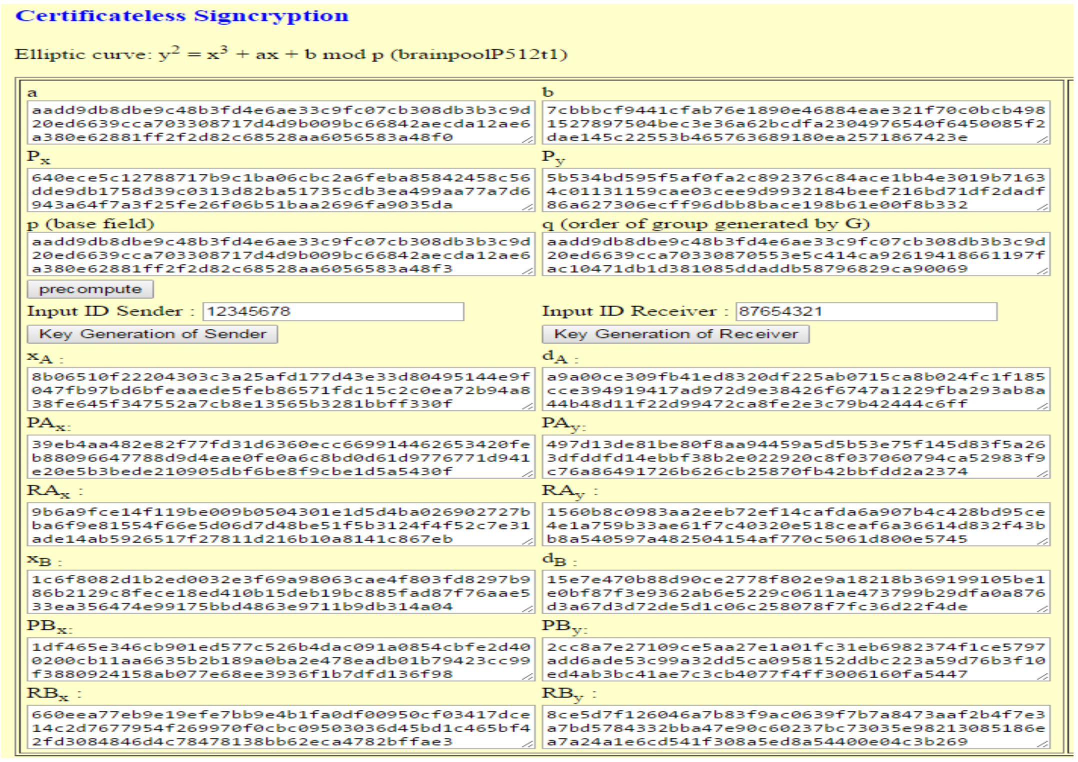
Elliptic Curve types	Var	Value (Hexadecimal)
brainpoolP256r1()	p	a9fb57dba1eea9bc3e660a909d838d726e3bf623d52620282013481d1f6e5377
	a	7d5a0975fc2c3057eef67530417affe7fb8055c126dc5c6ce94a4b44f330b5d9
	b	26dc5c6ce94a4b44f330b5d9bbd77cbf958416295cf7e1ce6bccdc18ff8c07b6
	n	a9fb57dba1eea9bc3e660a909d838d718c397aa3b561a6f7901e0e82974856a7
	P	8bd2aeb9cb7e57cb2c4b482ffc81b7afb9de27e1e3bd23c23a4453bd9ace3262, 547ef835c3dac4fd97f8461a14611dc9c27745132ded8e545c1d54c72f046997
	msk	3aea7fa0202e5d35038356102a6a9a19eb114d94f56498da40849f4105a9016
	P_{pub}	9a78b67f611ad4eb7d19d460cd4cc0e180d358c85a8212391bb266b1ab0ddb38, 67fb43aff1bf1a893520df1ef63145a9507856acce15061d6325c85c3ab0c7c3
brainpoolP512r1()	p	add9db8dbe9c48b3fd4e6ae33c9fc07cb308db3b3c9d20ed6639cca703308717d 4d9b009bc66842aecda12ae6a380e62881ff2f2d82c68528aa6056583a48f3
	a	7830a3318b603b89e2327145ac234cc594cbdd8d3df91610a83441caea9863bc 2ded5d5aa8253aa10a2ef1c98b9ac8b57f1117a72bf2c7b9e7c1ac4d77fc94ca
	b	3df91610a83441caea9863bc2ded5d5aa8253aa10a2ef1c98b9ac8b57f1117a72 bf2c7b9e7c1ac4d77fc94cadc083e67984050b75ebae5dd2809bd638016f723
	n	aadd9db8dbe9c48b3fd4e6ae33c9fc07cb308db3b3c9d20ed6639cca703308705 53e5c414ca92619418661197fac10471db1d381085ddaddb58796829ca90069
	P	81aee4bdd82ed9645a21322e9c4c6a9385ed9f70b5d916c1b43b62eef4d0098ef f3b1f78e2d0d48d50d1687b93b97d5f7c6d5047406a5e688b352209bcb9f822, 7dde385d566332ecc0eabfa9cf7822fdf209f70024a57b1aa000c55b881f8111b2 dcde494a5f485e5bca4bd88a2763aed1ca2b2fa8f0540678cd1e0f3ad80892
	msk	233276ac0ac1417aad31bab918f8b4676f0eca401343e2adfb154126a7df47ea90 7083357104431c1d0b12a61a85ac9561955783aa2ad71247c5a8ce3b3005e0
	P_{pub}	46fa83aee0bdb5e1197ef8b571a05b0f47ef44efd3ec6a8b739b0a72fd13c945a7 8d82a8ff1fc00949aecf47db237efa63f3edcc2e130d74ae2c1d80f31c577cl, 72a279fc2d13b3b54ff3434ad0ad85a8ff830fca4bc24b3b7260cc2f659711825e 61b3598717cc33fec53e0b970af07aab2d2623b5de98a7a89df234520be0d5
brainpoolP512t1()	p	aadd9db8dbe9c48b3fd4e6ae33c9fc07cb308db3b3c9d20ed6639cca703308717 d4d9b009bc66842aecda12ae6a380e62881ff2f2d82c68528aa6056583a48f3
	a	aadd9db8dbe9c48b3fd4e6ae33c9fc07cb308db3b3c9d20ed6639cca703308717 d4d9b009bc66842aecda12ae6a380e62881ff2f2d82c68528aa6056583a48f0
	b	7cbbbcf9441cfab76e1890e46884eae321f70c0bcb4981527897504bec3e36a62 bcdfa2304976540f6450085f2dae145c22553b465763689180ea2571867423e
	n	aadd9db8dbe9c48b3fd4e6ae33c9fc07cb308db3b3c9d20ed6639cca703308705 53e5c414ca92619418661197fac10471db1d381085ddaddb58796829ca90069
	P	640ece5c12788717b9c1ba06cbc2a6feba85842458c56dde9db1758d39c0313d 82ba51735cdb3ea499aa77a7d6943a64f7a3f25fe26f06b51baa2696fa9035da, 5b534bd595f5af0fa2c892376c84ace1bb4e3019b71634c01131159cae03cee9d 9932184beef216bd71df2dadf86a627306ecff96dbb8bace198b61e00f8b332
	msk	8e20b5a49ee25eec72bf9371da99b07156c7f11128b8607807ff5f347d6f80176c 576fcea2bb29540920fc8a2a71c925910d98def772276e706fa4b5c4d4fe32
	P_{pub}	8a27e89f37d19598bf4a4069b5cfdf25d54e5c3e931a1f26dcc862ed110f91a557 8ffcc04417b3af4fe6909c26d7abfba6291d1415533ddc2ebaea41ff6189aa, 2951add5a59615f9a5c9012498e43bcc4893e3cbf0c7c778be00199d172e1fde6 c46676727e7fc164ef36d36b4536462c9b69ca9274dfd2722cef00ac76e76c4

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 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 msk owned by KGC that must remain confidential. The following figure is the functions that have been mentioned.

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.

```
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("4b94caecbc45dda458c029536dacf6da1d06dea3
                        1d7f2d6ab358d56787cf63bb96dc58d19479b0c6bc69f09105c29
                        1f8ad6a18abeb968d1d7b7a25455eeb9cc4", 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



Certificateless Signcryption

Elliptic curve: $y^2 = x^3 + ax + b \bmod p$ (brainpoolP512t1)

a
```
aadd9db8dbe9c48b3fd4e6ae33c9fc07cb308db3b3c9d
20ed6639cca703308717d4d9b009bc66842aecda12ae6
a380e62881ff2f2d82c68528aa6056583a48f0
```
b
```
7cbbbcf9441cfab76e1890e46884eae321f70c0bcb498
1527897504bec3e36a62bcdfa2304976540f6450085f2
dae145c22553b465763689180ea2571867423e
```
P_x
```
640ece5c12788717b9c1ba06cbc2a6feba85842458c56
dde9db1758d39c0313d82ba51735cdb3ea499aa77a7d6
943a64f7a3f25fe26f06b51baa2696fa9035da
```
P_y
```
5b534bd595f5af0fa2c892376c84ace1bb4e3019b7163
4c01131159cae03cee9d9932184beef216bd71df2dadf
86a627306ecff96dbb8bace198b61e00f8b332
```
p (base field)
```
aadd9db8dbe9c48b3fd4e6ae33c9fc07cb308db3b3c9d
20ed6639cca703308717d4d9b009bc66842aecda12ae6
a380e62881ff2f2d82c68528aa6056583a48f3
```
q (order of group generated by G)
```
aadd9db8dbe9c48b3fd4e6ae33c9fc07cb308db3b3c9d
20ed6639cca70330870553e5c414ca92619418661197f
ac10471db1d381085ddaddb58796829ca90069
```
precompute

Input ID Sender : 12345678

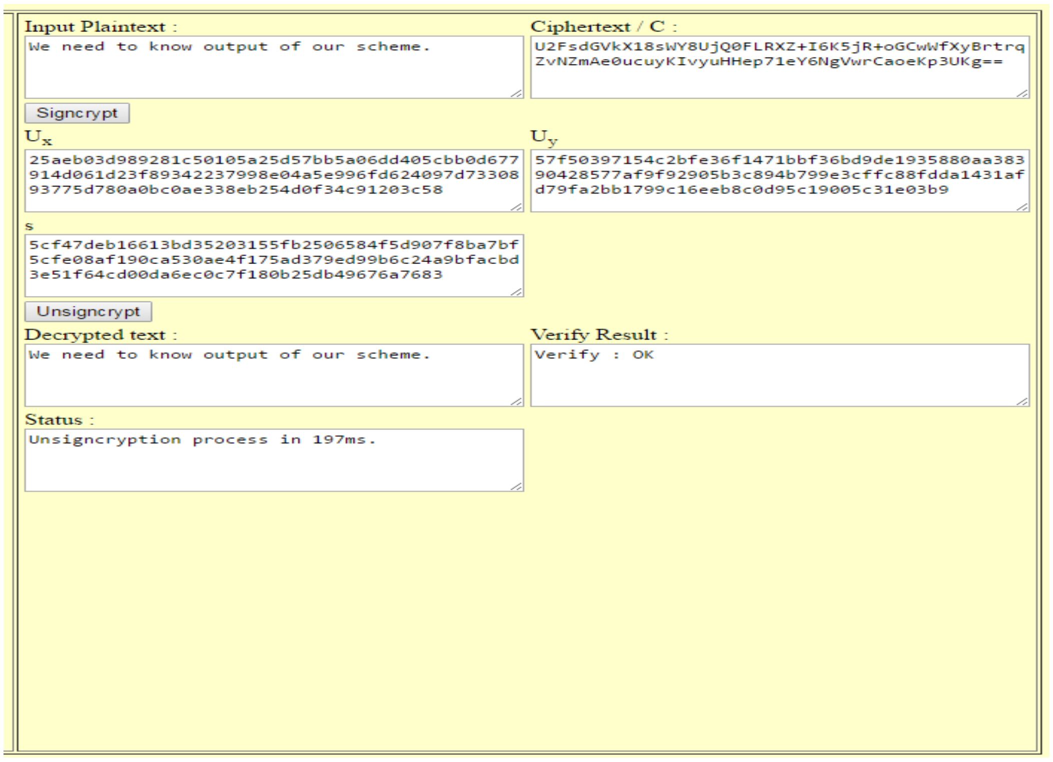
Key Generation of Sender

Input ID Receiver : 87654321

Key Generation of Receiver

x_A :
```
8b06510f22204303c3a25afd177d43e33d80495144e9f
047fb97bd6bfeaaede5feb86571fdc15c2c0ea72b94a8
38fe645f347552a7cb8e13565b3281bbff330f
```
d_A :
```
a9a00ce309fb41ed8320df225ab0715ca8b024fc1f185
cce394919417ad972d9e38426f6747a1229fba293ab8a
44b48d11f22d99472ca8fe2e3c79b42444c6ff
```
PA_x:
```
39eb4aa482e82f77fd31d6360ecc669914462653420fe
b88096647788d9d4eae0fe0a6c8bd0d61d9776771d941
e20e5b3bede210905dbf6be8f9cbe1d5a5430f
```
PA_y:
```
497d13de81be80f8aa94459a5d5b53e75f145d83f5a26
3dfddfd14ebbf38b2e022920c8f037060794ca52983f9
c76a86491726b626cb25870fb42bbfdd2a2374
```
RA_x :
```
9b6a9fce14f119be009b0504301e1d5d4ba026902727b
ba6f9e81554f66e5d06d7d48be51f5b3124f4f52c7e31
ade14ab5926517f27811d216b10a8141c867eb
```
RA_y :
```
1560b8c0983aa2eeb72ef14cafda6a907b4c428bd95ce
4e1a759b33ae61f7c40320e518ceaf6a36614d832f43b
b8a540597a482504154af770c5061d800e5745
```
x_B :
```
1c6f8082d1b2ed0032e3f69a98063cae4f803fd8297b9
86b2129c8fece18ed410b15deb19bc885fad87f76aae5
33ea356474e99175bbd4863e9711b9db314a04
```
d_B :
```
15e7e470b88d90ce2778f802e9a18218b369199105be1
e0bf87f3e9362ab6e5229c0611ae473799b29dfa0a876
d3a67d3d72de5d1c06c258078f7fc36d22f4de
```
PB_x:
```
1df465e346cb901ed577c526b4dac091a0854cbfe2d40
0200cb11aa6635b2b189a0ba2e478eadb01b79423cc99
f3880924158ab077e68ee3936f1b7dfd136f98
```
PB_y:
```
2cc8a7e27109ce5aa27e1a01fc31eb6982374f1ce5797
add6ade53c99a32dd5ca0958152ddbc223a59d76b3f10
ed4ab3bc41ae7c3cb4077f4ff3006160fa5447
```
RB_x :
```
660eea77eb9e19efe7bb9e4b1fa0df00950cf03417dce
14c2d7677954f269970f0cbc09503036d45bd1c465bf4
2fd3084846d4c78478138bb62eca4782bffae3
```
RB_y :
```
8ce5d7f126046a7b83f9ac0639f7b7a8473aaf2b4f7e3
a7bd5784332bba47e90c60237bc73035e98213085186e
a7a24a1e6cd541f308a5ed8a54400e04c3b269
```

Fig. 2. Snapshot of Key Generation Result



```
We need to know output of our scheme.
```

```
U2FsdGVkX18sWY8UjQ0FLRXZ+I6K5jR+oGCwWfXyBrtrq
ZvNZmAe0ucuyKIvyuHHep71eY6NgVwrCaoeKp3UKg==
```

```
25aeb03d989281c50105a25d57bb5a06dd405cbb0d677
914d061d23f89342237998e04a5e996fd624097d73308
93775d780a0bc0ae338eb254d0f34c91203c58
```

```
57f50397154c2bfe36f1471bbf36bd9de1935880aa383
90428577af9f92905b3c894b799e3cffc88fdda1431af
d79fa2bb1799c16eeb8c0d95c19005c31e03b9
```

```
5cf47deb16613bd35203155fb2506584f5d907f8ba7bf
5cfe08af190ca530ae4f175ad379ed99b6c24a9bfacbd
3e51f64cd00da6ec0c7f180b25db49676a7683
```

```
We need to know output of our scheme.
```

```
Verify : OK
```

```
Unsigncryption process in 197ms.
```

Fig. 3. Snapshot of Signcryption and Unsigncryption 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 $SK = h_2(l_A\ (\ Y_B + P_B\ ),U,ID_B,R_B,P_B)$ 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 $l_A.(Y_B + P_B)$ should be same with equation $(d_B+x_B).U$.

- While $U = l_A.P$
  $d_B = r_B + msk.h_0(ID_B,R_B,P_B)$
  $Y_B = R_B + h_0(ID_B,R_B,P_B).P_{pub}$
- Then
  $$\begin{aligned}(d_B+x_B).U &= (r_B + msk.\ h_0(ID_B,R_B,P_B) + x_B\ ).l_A.P\\ &= l_A(r_B.P + msk.\ h_0(ID_B,R_B,P_B).P + x_B.P)\\ &= l_A(R_B + h_0(ID_B,R_B,P_B).P_{pub} + P_B)\\ &= l_A\ (\ Y_B + P_B\ ).\end{aligned}$$

Then, for the formula $s.P = Y_A + U.h_1(m,ID_A) + P_A.h_1(m,ID_A)$ should be hold. It is because,

- While $$\begin{aligned}d_A.P &= (r_A + msk.h_0(ID_A,R_A,P_A)).P\\ &= r_A.P + h_0(ID_A,R_A,P_A).msk.P\\ &= R_A + h_0(ID_A,R_A,P_A).P_{pub} = Y_A\end{aligned}$$
- Then
  $$\begin{aligned}s.P &= (d_A + l_A.h_1(m,ID_A) + x_A.h_1(m.ID_A)).P\\ &= d_A.P + l_A.P.h_1(m,ID_A) + x_A.P.h_1(m.ID_A)\\ &= Y_A + U.h_1(m,ID_A) + P_A.\ h_1(m,ID_A).\end{aligned}$$

B. Security Analysis

A pair public and private key security relies on Elliptic curve discrete logarithm problem (ECDLP). Partial public key $Y_A = R_A + H(ID_A,R_A,P_A).P_{pub} = d_a.P$, where P_A, R_A, P_{pub} and P is a point on the elliptic curve over finite field and d_A is a quite large integer value. If partial private key d_A and P are given, it will be "easy" to compute partial public key Y_A. However if the ones given are partial public key Y_A and P, it will be "hard" to find partial private key d_A. On the same way for the other public key P_A, it comes from a random secret value x_A ($P_A = x_A.P$). If secret value x_A and P are given, it will be "easy" to compute public key P_A. However if the ones given are public key P_A and P, it will be "hard" to find secret value x_A.

- **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:
  $SK = h_2(l_A.(Y_B+P_B),U,ID_B,R_B,P_B)$
  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.
  $SK = h_2((d_B+x_B).U,U,ID_B,R_B,P_B)$
  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 (ID_A, 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), $s = (d_A + l_A.h_1(m,ID_A) + x_A.h_1(m,ID_A)).\mathrm{mod}\ q$ and send it to receiver. Then the receiver verifies the received data (m) using the received signature s. If $s.P = Y_A + U.h_1(m,ID_A) + P_A.\ h1(m,ID_A)$ 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.
  $$s.P = Y_A + U.h_1(m,ID_A) + P_A.\ h1(m,ID_A)$$
  If the attacker changed the ciphertext c to c1, then the received original message should also change from m to m1. As a result, during verification, the computed digital signature of m1 will not be the same as the digital signature of m (s) 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 $SK = h_2(l_A(Y_B + P_B),U,ID_B,R_B,P_B)$ equation and $s = (d_A + l_A.h_1(m,ID_A) + x_A.h_1(m,ID_A)).\mathrm{mod}\ q$. It is impossible to achieve without knowing l_A,d_A and x_A.
- **Non-repudiation**
  In this scheme, the receiver knows from the $SK = h_2((d_B+x_B).U,U,ID_B,R_B,P_B)$ 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 $s = (d_A + l_A.h_1(m,ID_A) + x_A.h_1(m,ID_A)).\mathrm{mod}\ q$ 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
- Ecmult = Elliptic Curve point multiplication 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

Schemes	Confidentiality	Authentication	Integrity	Unforgeability	Non-Repudiation	Forward Secrecy
Zheng[6]	Yes	Yes	Yes	Yes	No	No
ZI [16]	Yes	Yes	Yes	Yes	No	No
WNPZ [12]	Yes	Yes	Yes	Yes	Yes	Yes
XX [11]	Yes	Yes	Yes	Yes	Yes	Yes
SB [8]	Yes	Yes	Yes	Yes	Yes	unknown
WSB [9]	Yes	Yes	Yes	Yes	Yes	unknown
Ours	Yes	Yes	Yes	Yes	Yes	Yes

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

Schemes	Type	Participant	Exp	Div	Mul	Add	ECMult	ECAdd	Hash
Zheng[6]	SC[a]	Sender	1	1	-	1	-	-	2
		Receiver	2	-	2	-	-	-	2
ZI [16]	SC[a]	Sender	-	1	1	1	1	-	2
		Receiver	-	-	2	-	2	1	2
WNPZ [12]	CLSC[b]	Sender	4	1	3	2	-	-	4
		Receiver	5	-	3	2	-	-	4
XX [11]	CLSC[b]	Sender	5	-	4	2	-	-	3
		Receiver	5	-	4	2	-	-	3
SB [8]	CLSC-TKEM[c]	Sender	-	-	3	2	4	1	4
		Receiver	-	-	-	-	6	3	4
WSB [9]	CLSC-TKEM[c]	Sender	-	-	2	2	4	2	4
		Receiver	-	-	-	1	6	3	4
Our	CLSC[b]	Sender	-	-	2	2	3	2	3
		Receiver	-	-	-	1	5	3	3

[a]Signcryption, [b]Certificateless Signcryption, [c]Certificateless Signcryption-Tag Key Encapsulation Mechanism.

Table 4. Ciphertext size comparison

EC-CLSC Schemes	Ciphertext Size
SB [8]	$n_q + n_G + n_{ID} + m$
WSB [9]	$n_q + 2n_G + n_{ID} + m$
Ours	$n_q + n_G + n_{ID} + m$

n_q: The number of bits required to represent an element of F_q.
n_G: The number of bits required to represent an element of point EC.
n_{ID}: The number of bits required to represent an identity.
m: The number of bits in the message being signcrypted.

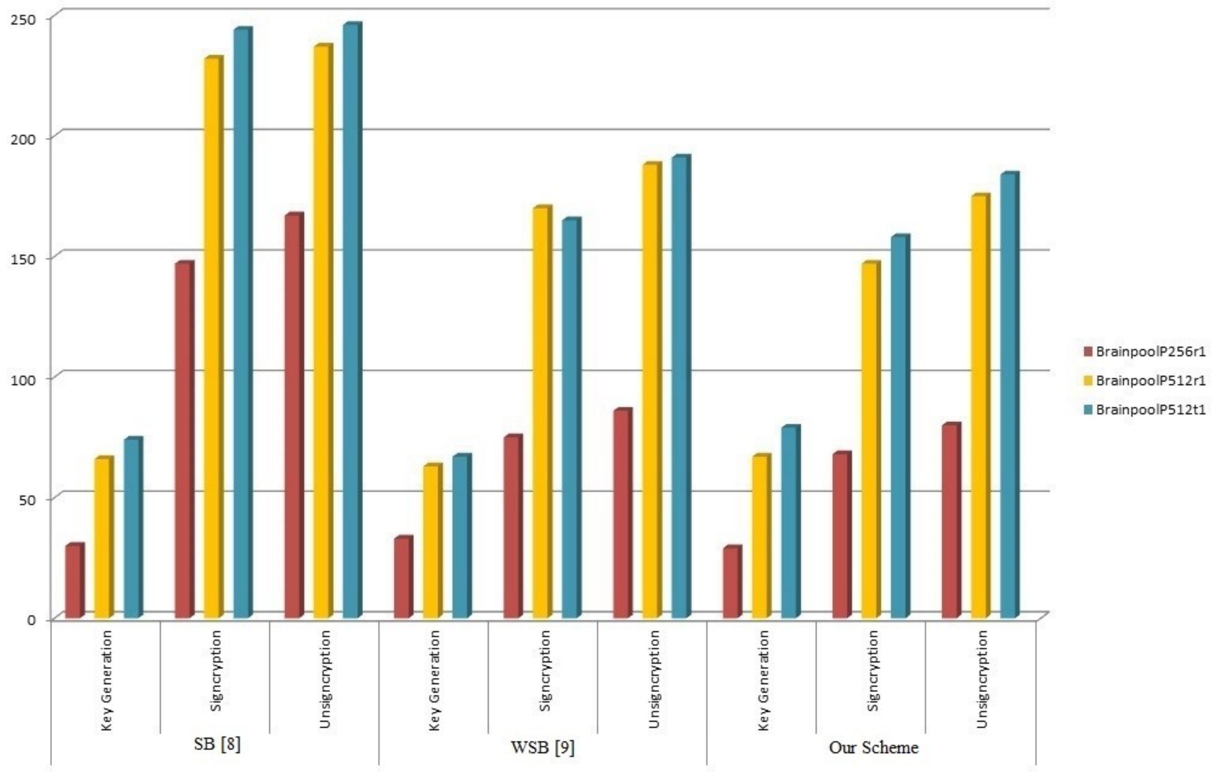


Fig. 4. Comparison of performance of the CLSC schemes based on elliptic curve

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

[1] S. William, *Cryptography and Network Security*, vol. 139, no. 3. Boston, USA.: Prentice Hall, 2011.
[2] A. Shamir, “Identity-based cryptosystems and signature schemes,” in *Advances in Cryptology-CRYPTO’84*, 1984, pp. 47–53.
[3] T. Okamoto, “A Single Public-Key Authentication Scheme for Multiple Users,” *IEICE Trans.*, vol. J69-D, no. 10, pp. 1481–1489, 1986.
[4] D. Boneh and M. Franklin, “Identity-Based Encryption from the Weil Pairing,” *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.
[5] S. Al-Riyami and K. Paterson, “Certificateless public key cryptography,” *Adv. Cryptology-ASIACRYPT 2003*, pp. 1–40, 2003.

[6] Y. Zheng, “Digital signcryption or how to achieve cost (signature & encryption) cost (signature)+ cost (encryption),” *Adv. Cryptol. — Crypto ’97*, no. March, pp. 165–179, 1997.

[7] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York Inc, 2006.

[8] S. Seo and E. Bertino, “Elliptic Curve Cryptography based Certificateless Hybrid Signcryption Scheme without Pairing,” *CERIAS Tech Rep. 2013-10*, 2013.

[9] J. Won, S.-H. Seo, and E. Bertino, “A Secure Communication Protocol for Drones and Smart Objects,” in *ASIA CCS’15*, 2015, pp. 249–260.

[10] M. Barbosa and P. Farshim, “Certificateless Signcryption,” *ACM Symp. Information, Comput. Commun. Secur.*, pp. 369–372, 2008.

[11] X. Zheng and X. Yang, “Improvement of a Certificate less Signcryption Scheme without pairing,” *Int. J. Sci.*, vol. 2, no. 7, pp. 81–87, 2015.

[12] S. Wenbo, K. Neeraj, G. Peng, and Z. Zezhong, “Cryptanalysis and improvement of a certificateless signcryption scheme without bilinear pairing,” *Front. Comput. Sci.*, vol. 8, no. 4, pp. 656–666, 2014.

[13] J. Baek, R. Steinfeld, and Y. Zheng, “Formal proofs for the security of signcryption,” *J. Cryptol.*, vol. 20, no. 2, pp. 203–235, 2007.

[14] L. Chen and J. Malone-Lee, “Improved Identity-Based Signcryption,” *Public Key Cryptogr. 2005*, vol. 3386, pp. 362–379, 2005.

[15] R. Steinfeld and Y. Zheng, “A Signcryption Scheme Based on Integer Factorization,” *Inf. Secur. LNCS*, vol. 1975, pp. 308–322, 2000.

[16] Y. Zheng and H. Imai, “How to construct efficient signcryption schemes on elliptic curves,” *Inf. Process. Lett.*, vol. 68, no. 5, pp. 227–233, 1998.

[17] Z. Jin, Q. Wen, and H. Du, “An improved semantically-secure identity-based signcryption scheme in the standard model,” *Comput. Electr. Eng.*, vol. 36, no. 3, pp. 545–552, 2010.

[18] Y. Yu, B. Yang, Y. Sun, and S. lin Zhu, “Identity based signcryption scheme without random oracles,” *Comput. Stand. Interfaces*, vol. 31, no. 1, pp. 56–62, 2009.