IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 14, Issue 6 Ver. I (Nov - Dec 2018), PP 73-84
www.iosrjournals.org

Efficient and Secure Pairing-free ID-based Directed Proxy Signature Scheme

R. R. V. Krishna Rao, N. B. Gayathri, P. Vasudeva Reddy.
(Department of Engineering Mathematics, Andhra University, India)
Corresponding Author: R. R. V. Krishna Rao

Abstract: *The proxy signature, a variant of the ordinary digital signature, has been an active research topic in recent years; it has many useful applications, including distributed systems and grid computing. In an ordinary proxy signature scheme any one can verify the validity of a proxy signature produced by the proxy signer on behalf of original signer. But public verifiability of proxy signature is not desirable in some applications where the signed message is sensitive to the signature receiver, for example signatures on medical records, tax information. To meet this requirement, the concept of directed proxy signature was introduced. A directed proxy signature scheme is a kind of signature scheme in which the verification ability is controlled by the proxy signer. Although many identity-based proxy signature schemes have been proposed in the literature, only a few identity-based directed proxy signature schemes are available. However, it has been found that all these schemes are using bilinear pairings over elliptic curves. But the computation of a bilinear pairing is very expensive. Hence the schemes which use pairings are less efficient and are not much applicable in practice. In order to improve the computational and communicational efficiency, in this paper, we propose a pairing-free Identity based directed proxy signature scheme. The proposed scheme is proven secure against different types of adversaries in the random oracle model under the assumption that the elliptic curve discrete logarithm problem is hard. We compare our scheme with well known existing schemes and efficiency analysis shows that the proposed scheme is more efficient.*

Date of Submission: 26-11-2018 Date of acceptance: 07-12-2018

I. Introduction

Digital signature is one of the fundamental and useful cryptographic primitive, which provides authentication and non-repudiation for electronic transactions in digital world. To implement the digital signature in the real world, it needs to consider different features and properties to make them adequate and proper for different usages. Many signature schemes have been proposed in literature with different cryptographic settings such as traditional public key infrastructure (PKI) [1] and Identity based cryptosystem [2]. The security of the traditional PKI is based on the certificate, signed by a certification authority (CA), containing the relationship between the key pairs, i.e., a public key and a private key, and the user's identity and legitimacy. But certificate management leads to extra storage, large computation and communication costs. Contrast to traditional PKI, Identity Based cryptosystem (IBC) [2] does not need any certificate to ensure the authenticity of public/private key pair. In this system, public key of a user is derived from the user's identity and the secret key is generated by a trusted third party called Private Key Generator (PKG).

To deal with different scenarios, digital signature schemes have evolved into many variants such as Blind signature, Multi signature, Group signature, Ring signature etc. One of such variants is Proxy signature. A proxy signature scheme is an important cryptographic technique and allows an entity to delegate the signing capability to one or more entities. Proxy signatures have found numerous practical applications, including distributed systems [3], grid computing [4], mobile agent systems [5], and mobile communications [6]. In 1996, Mambo et al. [7] introduced the proxy signature scheme, in which the original signer (Alice) delegated his signing privilege to a proxy signer such that the proxy signer (Bob) on behalf of the original signer can sign some specific messages. An entity (Cindy) who receives a message with a proxy signature can easily check the correctness of the signature and be convinced about the agreement of the original signer. Since then many new constructions have been proposed in literature [8, 9, 10]. Furthermore, various extensions of the basic proxy signature primitive have been considered. These include threshold proxy signatures [11], blind proxy signatures [12], proxy signatures with warrant recovery [13], nominative proxy signatures [14], one-time proxy signatures [5], and anonymous proxy signatures [15]. Also, according to the delegation ways, proxy signature can be classified into three types: full delegation [9], partial delegation [10], and delegation by warrant [8].

However, these proxy signature schemes allow public verification, which might not be suitable for applications in which the verification of the proxy signature is sensitive to the receiver, for example signatures

on medical records, tax information. To meet this requirement, Lim and Lee [16] proposed a new type of signature called Directed signature. In a directed signature scheme, a signer sends a signature on a message to a designated verifier; only the designated verifier can directly verify the signature, while the others know nothing about validity of the message without the help of the signer or the designated verifier. In case of trouble or if necessary, both signer and the designated verifier can prove the validity of the signature to any third party. Directed signature schemes are very useful in practical applications where signed message is sensitive to the signature receiver. After the introduction of directed signatures by Lim and Lee in [13], many directed signature schemes are proposed in PKI based setting [17, 18, 19, 20, 21, 22]. The first efficient ID-based directed signature scheme was presented by Sun et al. [23] in the random oracle model. In 2009, Zhang et al. [24] proposed an ID-based directed signature scheme without random oracles. In the same year, Uma Prasada Rao et al. [25] proposed an efficient ID-based directed signature scheme using bilinear pairings over elliptic curves. In 2012, J. Ku et al. [26] proposed an efficient ID-based directed signature scheme on hyper elliptic curves.

The combination of directed signature technique with proxy signature integrates the advantages of both and is more applicable for signing on sensitive messages. Consider the following situation: Doctor Alice has issued a hospital record to the patient Bob, in the form of doctor's digital signature. Alice can delegate his signing right to the proxy signer (Doctor Charlie). Bob then can exclusively verify these signatures with others knowing nothing about his state of illness. Otherwise, his state of illness is exposed. After a period time, Bob also needs to prove validity of his hospital record to other doctor for cure. At the same time, Doctor Charlie also shares the ability and responsibility to acknowledge this hospital records when Bob may not be convenient to do so. Motivated by the above scenario, some directed proxy signature schemes have been proposed in the literature [27, 28, 29]. In these schemes, the proxy signer generates a proxy signature on behalf of the original signer to a designated verifier. The designated verifier can directly verify the proxy signature and he can convince any other party about the validity of the signature.

Moreover, all the above directed proxy signature schemes are designed using bilinear pairings over elliptic curves. The time consuming cryptographic operation is pairing operation and is more expensive than the evaluation of a scalar multiplication in elliptic curve. For example, ECC with 224 bit keys provides the same level of security as RSA with 2048 bit keys. Thus ECC has become popular since it provides higher security with smaller keys in size. This smaller key size improves the computational and communicational efficiency, storage capacity, bandwidth efficiency. The evaluation of one pairing operation is 20 times with that of scalar multiplication. In this regard, to further improve the computational efficiency in directed proxy signature schemes, it is required to design the signature scheme without using bilinear pairing operations. This motivated us to design a pairing free directed proxy signature scheme in ID-based setting.

Our Contribution: In this paper, we proposed an Identity based directed proxy signature scheme, in which only the designated verifier can directly verify the proxy signature generated by a proxy signer on behalf of the original signer and any other party can verify the validity of the proxy signature with the help of the *Aid* provided by the proxy signer or the designated verifier. To the best of our knowledge, this is the first pairing free directed proxy signature scheme. The proposed directed proxy signature scheme is secure against forgeability and visibility. We proved the security in random oracle model (ROM) under the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP). We compare our proposed scheme with the existing directed proxy signature schemes in terms of computational and communication point of view. Efficiency analysis shows that the proposed scheme is more efficient than all other schemes in the literature.

Organization: The rest of the paper is organized as follows. Section 2 briefly presents some preliminaries. The syntax and security model of our Pairing Free ID-based Directed Proxy Signature (PF-IDBDPS) scheme is given in Section 3. Section 4 presents our PF-IDBDPS Scheme. The correctness and security analysis of the proposed scheme is given in Section 5. Efficiency analysis is provided in Section 6. Section 7 concludes this paper.

II. Preliminaries

In this section we briefly describe the fundamental concepts on elliptic curve and the complexity assumption, on which the proposed scheme is designed and achieves the desired security.

Elliptic Curve Group: Let the symbol E/F_P denote an elliptic curve E over a prime finite field F_P, defined by an equation $y^2 = (x^3 + ax + b),\ a, b \in F_P$ and with the discriminant $\Delta = 4a^3 + 27b^2 \neq 0$. The points on E/F_P together with an extra point 'O' called the point at infinity form a group $G = \{(x, y) : x, y \in F_P, E(x, y) = 0\} \cup \{O\}$. Now G forms an additive group with point addition. For further details please refer [30].

Elliptic Curve Discrete Logarithm Problem (ECDLP): Given a random instance P the generator of G and $Q = xP$ where $x \in Z_q^*$, compute x from P and Q. Computation of x from P and Q is computationally hard by any polynomial-time bounded algorithm.
Notations and their meanings which we used throughout this paper are presented in the following Table no 1.
Table no 1: Notations and their meanings.

Notation	Meaning
n, s	Security parameter and Master secret key of the system generated by Private Key Generator (PKG).
$params$	System Parameters.
z_q^*	The group with elements 1,2…q-1 under addition modulo q.
G	Additive cyclic group of prime order q.
H_1, H_2, H_3, H_4 ID_{os}, ID_{ps}, ID_v	Cryptographic one way hash functions. Original Signer's identity, Proxy Signer's identity and designated verifiers identity respectively.
D_i, PK_i	Secret key and Public key of the identity respectively.
$\mathrm{ADV}_1, \mathrm{ADV}_2, \mathrm{ADV}_3$	Type 1,Type 2 and Type 3 adversaries respectively.
ξ	An algorithm to solve ECDL problem by using adversaries
DSTG	A distinguisher to distinguish a valid signature on an adaptively chosen message by the attacker from one randomly drawn from the signature space.
Ω	Directed Proxy Signature on a message.
W	Delegation
m_w	Warrant message
σ	Part of the delegation
S	Proxy signing key

III. Syntax and Security Model

This section presents the syntax and security model for our PF-IDBDPS scheme.
Syntax of PF-IDBDPS Scheme: A formal model of the proposed PF-IDBDPS scheme consists of eight components whose functionalities are described as follows.

1. **Setup:** PKG runs this algorithm by taking $n \in Z^+$ as input and generates master secret key and master public key and publishes the list of public parameters as *params*.
2. **Extract:** PKG runs this algorithm by taking *params*, master secret key, *ID* as input and generates private keys D_i, for all entities participating in the scheme and distributes the private keys to their respective owners through a secure channel.
3. **Delegation Gen:** Taking *params*, master public key, an original signers identity ID_{os} with its private key D_{os}, a warrent m_w as input, this algorithm is run by the original signer and generates the delegation $W_{os \to ps}$ on the warrent m_w.
4. **Delegation Verification***:* Given a delegation $W_{os \to ps}$ on the warrent m_w, the proxy signer verifies and accepts the delegation of original signer if the delegation is valid; rejects otherwise.
5. **Proxy Key Gen:** Taking the proxy signer's private key and the delegation of the original signer $W_{os \to ps}$, the proxy signer run this algorithm to generate the proxy signing key S.
6. **Directed Proxy Signature Generation:** To sign a message $m \in \{0,1\}^*$ for a user with identity ID_v, this algorithm takes *params*, delegation, S, PK_{ps}, ID_v, PK_v and message $m \in \{0,1\}^*$ as input and outputs a directed proxy signature Ω.
7. **Directed Proxy Signature Verification (D.Verify):** The Designated Verifier ID_v runs this algorithm. To verify a directed proxy signature Ω on a message *m*, this algorithm takes *params*, Ω, PK_{os}, PK_{ps}, PK_v and ID_{os}, ID_{ps}, ID_v as input, and outputs '*accept*' Ω or '*reject*', otherwise.
8. **Public Proxy Signature Verification (P.Verify):** To verify a signature Ω on a message *m*, this algorithm takes *params*, Ω, PK_{os}, PK_{ps}, PK_v, ID_{os}, ID_{ps}, ID_v and an ***Aid*** provided by the proxy signer ID_{ps} or the designated verifier ID_v as input, and outputs '*accept*' Ω or '*reject*', otherwise.

Security Model of PF-IDBDPS Scheme:

1. **Unforgeability:** According to the definition of Huang *et al.* [31], there are three types of adversaries:
   Type 1: Adversary ADV_1 only has the public keys of the original signer and proxy signer.
   Type 2: Adversary ADV_2 has the public keys of the original signer and proxy signer, and it also has the private key of the proxy signer.
   Type 3*:* Adversary ADV_3 has the public keys of the original signer and proxy signer, and it also has the private key of the original signer.
   Clearly, if the directed proxy signature scheme is secure against types 2 and 3 adversary, it is also secure against type 1. In the following analysis, we only consider types 2 and 3 adversary.
2. **Existential unforgeability against the adversary** ADV_2**:** Existential unforgeability of the directed proxy signature scheme can be defined by considering the following game played between a challenger ξ and an adversary ADV_2.
   i. **Initialization Phase:** The challenger ξ runs the Setup algorithm to generate the systems parameters params, master secret key and master public key and sends them to the ADV_2 by keeping s secret.
   ii. **Queries Phase:** In this phase, ADV_2 makes queries on the following oracles.
   - **Extraction Oracle:** On receiving a query from ADV_2, the challenger ξ computes D_i by taking ID_i as input and gives this to ADV_2.
   - **Delegation Generation queries:** On receiving a query from ADV_2, the challenger ξ computes delegation W by taking designator's identity ID_i and a warrant m_w.
   - **Delegation Verification Oracle:** On receiving a query from adversary ADV_2 with (ID_i, W, m_w), ξ checks the validity of the delegation. It outputs 1 if the delegation is valid. Otherwise returns 0.

   iii. **Forgery Phase:** Finally, ADV_2 outputs (ID_i^*, W, m_w), as delegation and wins the game if
   - (ID_i^*, W, m_w) is a valid delegation.
   - (ID_i^*) has never been submitted to the Extraction Oracle and has never been queried to the Delegation Generation Oracle.

Definition 1: A directed proxy signature scheme is existential unforgeability against ADV_2, if the advantage ε is negligible after making at most q_{DG} Delegation Generation queries with in running time t.

3. **Existential unforgeability against the adversary** ADV_3**:** Existential unforgeability of the directed proxy signature scheme can be defined by considering the following game played between a challenger ξ and an adversary ADV_3.

i. **Initialization Phase:** The challenger ξ runs Setup algorithm to generate the systems parameters *params*, master secret key and master public key and sends them to the ADV_3 by keeping s secret.

ii. **Queries Phase:** In this phase, ADV_3 makes queries on the following oracles.
- **Extraction Oracle:** On receiving a query from ADV_3, the challenger ξ computes D_i by taking ID_i as input and gives this to ADV_3.
- **Delegation Generation queries:** On receiving a query from ADV_3, the challenger ξ computes delegation W by taking designator's identity ID_i and a warrant m_w.
- **Proxy Key Generation queries:** When ADV_3 queries a Proxy Key Gen of the proxy signer for W, m_w, ξ computes the proxy signing key S_{ps} and ξ responds to ADV_3 with S_{ps}.
- **Directed Proxy Signature Generation queries:** On receiving a query from adversary ADV_3 with (ID_{ps}, ID_v, m), proxy signing oracle returns a valid signature σ signed by public/private key of the proxy user ID_{ps}, by taking delegation W with message $m \in \{0,1\}^*$ as input.

- **Direct Verify Oracle:** On receiving a query from adversary ADV_3 with $(ID_{ps}, ID_v, m, \sigma)$, ξ checks the validity of the signature by extracting ID_v's private key D_v. It outputs 1 if the proxy signature is valid. Otherwise returns 0.
- **Public Verify Oracle:** On receiving a query from adversary ADV_3 with $(ID_{ps}, ID_v, m, \sigma)$, ξ checks the validity of the proxy signature and returns $\perp$ to ADV_3 if σ is invalid. Otherwise, ξ produces an **Aid** in the name of the signer ID_s or the designated verifier ID_v, then forwards **Aid** to ADV_3.

iii. **Forgery Phase:** Finally, ADV_3 outputs $(ID_{ps}^*, ID_v^*, m^*, \sigma^*)$ as forgery and wins the game if

- σ^* is a valid signature.
- (ID_{ps}^*) has never been submitted to the Extraction Oracle and Proxy Key Generation, (ID_{ps}^*, ID_v^*, m^*) has never been queried to the Proxy Signature Oracle.

Definition 2: A directed proxy signature scheme is existential unforgeability against ADV_3, if the advantage ε is negligible after making at most q_{DPS} Directed Proxy Signature queries, q_{DV} Directed Verification queries, and q_{PV} Public Verification queries with in running time t.

Definition 3: (Unforgeability): An IDBDPS scheme is said to be EUF-CMA, if there exists no polynomial time adversary (Type 2 and Type 3) with non-negligible advantage in the above two games.

Invisibility: The invisibility property requires that it should be (computationally) infeasible for any fourth party to decide whether a signature was indeed produced by a proxy signer ID_{ps}, designated to ID_v, on message m. To precisely define this property, we consider the following game between a probabilistic polynomial time distinguisher DSTG and a challenger ξ as described in [23].

Invisibility against the distinguisher DSTG: The game between a challenger ξ and a distinguisher DSTG is defined as follows:

1. **Initialization Phase:** This phase is same as in the above Game.
2. **Phase 1:** DSTG adaptively makes a number of different queries to the challenger as mentioned in the above Game. The Challenger responds to these queries in the same way as in the unforgeability game.
3. **Challenge:**
   After Phase 1 is over, DSTG submits ID_{ps}, ID_v, and a message m to the challenger under the following conditions: ID_v^*, has not been submitted to Extraction queries, Delegation Generation queries and proxy signature generation queries. The Challenger then generates a random bit $b \in \{0,1\}$ and produces a signature σ^* as in the Sign Oracle if $b = 1$. Otherwise, it picks a random σ^* from the signature space. In both cases σ^* is forwarded to DSTG.
4. **Phase 2:** DSTG again adaptively performs several oracle queries as it did in Phase 1, subjected to the following conditions:
   DSTG cannot run Delegation Generation queries ID_v^*; and Direct Verifiy or Public Verifiy Oracle on $(ID_{ps}^*, ID_v^*, m^*, \sigma^*)$.
5. **Guess:** Finally DSTG outputs a bit $b' \in \{0,1\}$. DSTG succeeds if $b = b'$.

Definition 4 (Invisibility): An IDBDPS scheme is said to be invisible, if there exists no polynomial time distinguisher with non-negligible advantage in the above game.

IV. Proposed Pairing Free Identity-Based Directed Proxy Signature (PF-IDBDPS) Scheme.

In this section we propose our efficient PF-IDBDPS scheme and we prove its security.

PF-IDBDPS Scheme: The proposed PF-IDBDPS scheme consists of the following algorithms.

1. Setup: Given a security parameter $n \in Z^+$, PKG does the following.

i. PKG chooses (q, P, G) according to n, where q is a prime, G is additive cyclic elliptic curve group, P is the generator of G.

ii. PKG selects a random $s \in Z_q^*$ as the master secret key and sets master public key as $P_{Pub} = sP$.

iii. Choose four cryptographic hash functions $H_1, H_2, H_3, H_4, : \{0,1\}^* \to Z_q^*$. PKG publishes the system parameters as $\tau = \{q, G, P, P_{Pub}, H_1, H_2, H_3, H_4\}$ and keeps s secret.

2. Extract: PKG runs this algorithm by taking ID_i and system parameters τ as input. PKG chooses a random number $r_i \in Z_q^*$, and computes $R_i = r_i P, h_{1i} = H_1(ID_i, R_i, P_{Pub})$ and $d_i = r_i + sh_{1i} \bmod q$. PKG sends $D_i = (d_i, R_i)$ to the user securely. User keeps d_i as his private key and publishes R_i. The user can validate D_i by checking whether the equation $d_i P = R_i + h_{1i} P_{Pub}$ holds or not.

3. Delegation Generation: The original signer creates a warrant m_w which keeps the record of proxy information such as the identities of the original signer, proxy signer, proxy validity period etc. Original signer chooses a number $x \in Z_q^*$ and computes $X = xP, h_{2os} = H_2(m_w, X, ID_{ps})$ and $\sigma = h_{2os} d_{os} + x \bmod q$. Original signer outputs $(ID_{os}, R_{os}, ID_{ps}, m_w, X, \sigma)$ as delegation on the warrant m_w and send it to the proxy signer.

4. Delegation Verification: Given a delegation $(ID_{os}, R_{os}, ID_{ps}, m_w, X, \sigma)$, the proxy signer computes $h_{2os} = H_2(m_w, X, ID_{ps})$ and checks whether the equation $\sigma P = h_{2os}(R_o + h_{1os} P_{Pub}) + X$ holds or not. If it holds, proxy signer accepts the delegation (X, σ) corresponding to ID_{os}, R_{os}, ID_{ps}, on m_w. Otherwise rejects.

5. Proxy Key Generation: After validating the delegation (X, σ), proxy signer generates the proxy signing key by using original signer's delegation key and proxy signer's private key d_{ps}.

i. Compute $h_{2ps} = H_2(m_w, X, ID_{os})$.

ii. Compute $S = \sigma + h_{2ps} d_{ps} \bmod q$.

6. Proxy Signature Generation: To generate a valid directed proxy signature on a given message $m \in \{0,1\}^*$, proxy signer takes designated verifier identity ID_v, R_v and proxy signing key S as input along with τ, ID_{ps}, R_{ps}, m and does as follows.

i. The proxy signer chooses $t_1, t_2 \in Z_q^*$ and computes $U_{ps} = t_1 P, V_{ps} = t_2 P, W_{ps} = U_{ps} + R_v + h_{1v} P_{Pub}$.

ii. Compute $h_3 = H_3(m, ID_{ps}, ID_v, U_{ps}, R_{ps})$ and $h_4 = H_4(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, h_3)$.

iii. Compute $k_{ps} = h_3 S + h_4 t_2 \bmod q$.

Hence the proxy signature is $\Omega = (R_{ps}, W_{ps}, V_{ps}, k_{ps}, ID_{os}, R_{os}, X, \sigma)$. The proxy signer sends the proxy signature as (m_w, m, Ω_1) to the designated verifier.

7. Designated Proxy Signature Verification:

To accept or reject the proxy signature (m_w, m, Ω_1), designated verifier takes $(\tau, (m_w, m, \Omega_1), ID_v, R_v, ID_{ps}, R_{ps})$ as input and does the following.

i. Check whether the message m is the same as defined in m_w, continue if the message and warrant are valid and correspond to each other. Reject otherwise.

ii. $Y_v = W_{ps} - d_v P = (U_{ps} + R_v + h_{1v} P_{Pub}) - (r_v + h_{1v} s) P = U_{ps} + R_v + h_{1v} P_{Pub} - R_v - h_{1v} P_{Pub} = U_{ps}$.

iii. Compute $h_3 = H_3(m, ID_{ps}, ID_v, U_{ps}, R_{ps})$ and $h_4 = H_4(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, h_3)$.

iv. Check whether the following equation holds or not $(k_{ps} P - (R_{ps} + h_{1ps} P_{Pub}) h_{2ps} h_3) h_4^{-1} = h_4^{-1}(h_3 h_{2os}(R_{os} + h_{1os} P_{Pub}) + X) + V_{ps}$.

If the equation holds, verifier accepts the signature (m_w, m, Ω_1) and outputs 1; rejects and outputs 0 otherwise.

8. Proxy signature Public Verification:

Any Fourth party other than designated verifier can check the validity of proxy signature with the help of the aid provided by designated verifier or proxy signature generator. Public verifier takes $(\tau, (m_w, m, \Omega_1), ID_v, R_v, ID_{ps}, R_{ps})$ as input and does as follows.

i. Either ID_{ps} or ID_v computes $Aid = U_{ps} = Y_v$, and then sends to the fourth party.

ii. Compute $h_3 = H_3(m, ID_{ps}, ID_v, U_{ps}, R_{ps})$ and $h_4 = H_4(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, h_3)$.

iii. Check whether the equation holds or not.

$(k_{ps}P - (R_{ps} + h_{1ps}P_{Pub})h_{2ps}h_3)h_4^{-1} = h_4^{-1}(h_3h_{2os}(R_{os} + h_{1os}P_{Pub}) + X) + V_{ps}$.

If it does, return 1, else 0.

Proof of correctness of the proposed scheme: The correctness of the scheme can be verified as follows.

$$(k_{ps}P - (R_{ps} + h_{1ps}P_{Pub})h_{2ps}h_3)h_4^{-1}$$
$$= ((h_3S + h_4t_2)P - (R_{ps} + h_{1ps}P_{Pub})h_{2ps}h_3)h_4^{-1}$$
$$= (h_3(\sigma P + h_{2ps}d_{ps}P) + h_4t_2P - (R_{ps} + h_{1ps}P_{Pub})h_{2ps}h_3)h_4^{-1}$$
$$= (h_3(h_{2os}(R_{os} + h_{1os}P_{Pub}) + X) + h_3(h_{2ps}(R_{ps} + h_{1ps}P_{Pub})) + h_4t_2P - (R_{ps} + h_{1ps}P_{Pub})h_{2ps}h_3)h_4^{-1}$$
$$= (h_3(h_{2os}(R_{os} + h_{1os}P_{Pub}) + X) + h_4t_2P)h_4^{-1}$$
$$= h_4^{-1}(h_3h_{2os}(R_{os} + h_{1os}P_{Pub}) + X) + V_{ps}.$$

V. Security Analysis

In this section, we study the security of our PF-IDBDPS scheme against Type II and Type III adversaries. The security results are described in the following theorems.

Theorem 1: If there exists a probabilistic polynomial time bounded Type II adversary ADV_2 who can break our proposed PF-IDBDPS scheme under the adaptively chosen message and identity attacks in the ROM, then there exists an algorithm ξ that can be used by ADV_2 to solve the ECDLP.

Proof: The Type II adversary ADV_2 knows the public keys of original signer and proxy signer and also knows the private key of proxy signer. The unforgeability of the proposed proxy signature scheme against Type II adversary ADV_2 requires that it is difficult to generate a valid delegation without original signer's private key. If ADV_2 generates a valid delegation, then he can compute the valid proxy signing key easily (as ADV_2 knows proxy's private key too) and a valid directed proxy signature as well. Now we show that, if there exists an ADV_2 who can forge a valid delegation of our scheme, then there exists an algorithm ξ to solve an instance of ECDLP. Thus, ξ can compute a for a given random instance $(P, aP) \in G$ where $a \in Z_q^*$. To solve ECDLP, ξ sets master secret key as a and master public key as $P_{Pub} = aP$, where a is unknown.

Simulation process is considered to be in ROM.

Initialization Phase: ξ generates the system parameters τ by running the Setup algorithm and sends it to ADV_2. Next, ξ answers ADV_2's queries in the following way.

Queries Phase: ADV_2 performs the oracle simulation and ξ responds to these oracles as follows.

1. **Queries on oracle** $H_1\left(H_1(ID_i, R_i, P_{Pub})\right)$: ξ maintains a list $\mathcal{L}_1$ which is initially empty. It contains tuples of the form $(ID_i, R_i, P_{Pub}, l_{1i})$. After receiving a query on $H_1(ID_i, R_i, P_{Pub})$, if there is tuple $(ID_i, R_i, P_{Pub}, l_{1i})$ on $\mathcal{L}_1$, ξ returns l_{1i}. Otherwise, ξ picks a random l_{1i} and adds to $\mathcal{L}_1$. Finally, ξ returns l_{1i}.
2. **Queries on oracle** $H_2\left(H_2(m_w, X_i, ID_i)\right)$: ξ maintains a list $\mathcal{L}_2$ which is initially empty. It contains tuples of the form (m_w, X_i, ID_i, l_{2i}). After receiving a query on $H_2(m_w, X_i, ID_i)$, if there is tuple (m_w, X_i, ID_i, l_{2i}) on $\mathcal{L}_2$, ξ returns l_{2i}. Otherwise, ξ picks a random l_{2i} and adds to $\mathcal{L}_2$. Finally, ξ returns l_{2i}.
3. **Extraction Queries:** When ADV_2 makes this query on ID_i, ξ first makes queries on H_1 and recovers l_{1i} from $\mathcal{L}_1$ list. Then ξ replies to ADV_2 as follows.
   i. If $i =$ original signer, aborts.
   ii. If $i \neq$ original signer, ξ chooses $r_i \in Z_q^*$ sets $R_i = r_iP - l_{1i}P_{Pub}$ and $d_i = r_i$.

4. **Delegation Generation Queries:** On receiving a Delegation Gen query on the warrant m_w with the original signer's identity ID_i, ξ first recovers the values l_{1i}, l_{2i} from $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively and then performs the following.
   i. If $ID_i = ID_{os}$ then ξ chooses $x_i \in Z_q^*$ and sets $X_i = x_i P$ and computes $\sigma_i = l_{2i} d_i + x_i \bmod q$.
   ii. If $ID_i \neq ID_{os}$, quit the protocol.

   Finally ξ returns $(ID_i, R_i, m_w, X_i, \sigma_i)$ as the delegation on m_w with the original signer's identity ID_i.
5. **Delegation Verification Queries:** On receiving Delegation verification query on (X_i, σ_i) on m_w with original signers identity ID_i, ξ recovers $(ID_i, R_i, P_{Pub}, l_{1i})$, (m_w, X_i, ID_i, l_{2i}) from $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively and performs the following
   i. If $ID_i = ID_{os}$ then ξ aborts.
   ii. Otherwise ξ verifies the correctness of the delegation.

   ξ verifies the correctness of the delegation (X_i, σ_i) with the equation $\sigma_i P = l_{2i}(R_i + h_{1i} P_{Pub}) + X_i$ and outputs the result. Note that the delegation (X_i, σ_i) is valid if ID_i, m_w have never been queried during the Extraction and Delegation Generation oracles respectively.

Finally, ADV_2 outputs (X_i^*, σ_i^*) with h_{2i}^* on m_w^* as a valid delegation with original signer's identity ID_i. Based on Forking Lemma [32], ξ recovers another (m_w^*, R_i^*, l_{2i}^*) from the list $\mathcal{L}_2$ and then replays the random oracle with same random tape but different choice of hash value of H_2. i.e. on the same warrant m_w^*, ξ obtains another forged delegation (X_i, σ_i) with h_{2i} such that $h_{2i}^* \neq h_{2i}$ and $\sigma_i^* \neq \sigma_i$. Hence (X_i, σ_i) and (X_i^*, σ_i^*) are two valid delegations on the same warrant m_w^*. Hence the following equations holds. $\sigma_i P = l_{2i}(R_i + h_{1i} P_{Pub}) + X_i$ and $\sigma_i^* P = l_{2i}^*(R_i + h_{1i}^* P_{Pub}) + X_i$.

By r_i, s, we now denote discrete logarithms of R_i, P_{Pub} respectively, i.e. $R_i = r_i P, P_{Pub} = sP$. Here r_i, s are unknown to ξ. ξ solves these values from the above equations and outputs s as the solution of ECDLP.

Theorem 2: The proposed PF-IDBDPS scheme is existentially unforgeable against adaptive chosen message and identity attack by adversary ADV_3 of Type III and is used to solve ECDLP.

Proof: A Type III adversary ADV_3 knows the public keys of original signer and proxy signer and also knows the private key of original signer. Hence ADV_3 can generate a valid delegation but cannot generate a valid proxy signing key as it does not know proxy signers private key. Thus ADV_3 attempts to generate a valid proxy signature without the proxy signing key or private key of designated verifier. We now show that, if ADV_3 can generate a forged proxy signature, then there exist an algorithm ξ that can use ADV_3 to solve an instance of ECDLP. Then for a given random instance $(P, sP) \in G$ where $s \in Z_q^*$, ξ can compute s. To solve ECDLP, ξ sets master key as s which is unknown to ξ and master public key as $P_{Pub} = sP$.

Initialization Phase: ξ generates the systems parameters τ by running the Setup algorithm and sends it to ADV_3 and answers ADV_3 queries as follows.

Sets $P_{Pub} = sP$ and runs **Master Key Gen** to generate *params*. ξ then gives *params* and master public key to ADV_3 and keeps s secretly.

Queries Phase: ADV_3 performs the oracle simulation and ξ responds to these oracles as follows.

1. **Queries on oracle** $H_1(H_1(ID_i, R_i, P_{Pub}))$: ξ maintains a list $\mathcal{L}_1$ which is initially empty. It contains tuples of the form $(ID_i, R_i, P_{Pub}, l_{1i})$. After receiving a query on $H_1(ID_i, R_i, P_{Pub})$, if there is tuple $(ID_i, R_i, P_{Pub}, l_{1i})$ on $\mathcal{L}_1$, ξ returns l_{1i}. Otherwise, ξ picks a random l_{1i} and adds to $\mathcal{L}_1$. Finally, ξ returns l_{1i}.

2. **Queries on oracle** H_2 $\left(H_2(m_w, X_i, ID_i)\right)$: ξ maintains a list $\mathcal{L}_2$ which is initially empty. It contains tuples of the form (m_w, X_i, ID_i, l_{2i}). After receiving a query on $H_2(m_w, X_i, ID_i)$, if there is tuple (m_w, X_i, ID_i, l_{2i}) on $\mathcal{L}_2, \xi$ returns l_{2i}. Otherwise, ξ picks a random l_{2i} and adds to $\mathcal{L}_2$. Finally, ξ returns l_{2i}.
3. **Queries on oracle** H_3 $\left(H_3(m, ID_{ps}, ID_v, U_{ps}, R_{ps})\right)$**:** ξ maintains a list $\mathcal{L}_3$ which is initially empty. It contains tuples of the form $(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i})$. After receiving a query on $H_3(m, ID_{ps}, ID_v, U_{ps}, R_{ps})$, if there is tuple $(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i})$ on $\mathcal{L}_3, \xi$ returns l_{3i}. Otherwise, ξ picks a random l_{3i} and adds to $\mathcal{L}_3$. Finally, ξ returns l_{3i}.
4. **Queries on oracle** H_4 $\left(H_4(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, h_{3i})\right)$**:** ξ maintains a list $\mathcal{L}_4$ which is initially empty. It contains tuples of the form $(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i}, l_{4i})$ After receiving a query on $H_4(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, h_{3i})$ if there is tuple $(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i}, l_{4i})$ on $\mathcal{L}_4, \xi$ returns l_{4i}. Otherwise, ξ picks a random l_{4i} and adds to $\mathcal{L}_4$. Finally, ξ returns l_{4i}.
5. **Extraction Queries:** When ADV_3 makes this query on ID_i, ξ replies to ADV_3 as follows.
   i. If $i =$ proxy signer or designated verifier, it aborts.
   ii. If $i \neq$ proxy signer or designated verifier, then ξ chooses $r_i \in Z_q^*$ sets $R_i = r_i P - l_{1i} P_{Pub}$ and $d_i = r_i$.
6. **Delegation Generation Queries:** When ADV_3 makes this query to ξ on the warrant m_w with the original signer's identity ID_i, ξ computes the corresponding delegation. Here ξ knows the original signer's private key, and hence ξ can execute Delegation Generation queries on (ID_i, m_w) to compute the corresponding delegation (X_i, σ_i).
7. **Proxy Key Generation:** When ADV_3 queries a Proxy Key Gen of the proxy signer for m_w, ξ computes the proxy signing key $S_j = \sigma_i + d_j h_{2ps}$ and ξ responds to ADV_3 with S_j. (Here j represents the proxy signer and i represents the original signer).
8. **Proxy Signature Generation Queries:** When ADV_3 makes this query on (ID_j, m_j), with a verifier ID_v, ξ first makes queries on H_1, H_2, H_3, H_4 oracles and recovers the tuples $(ID_i, R_i, P_{Pub}, l_{1i})$, (m_w, X_i, ID_i, l_{2i}), $(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i})$, $(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i}, l_{4i})$ from $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4$ respectively and ADV_3 does as follows. ξ generates two random numbers $r_{1j}, r_{2j} \in Z_q^*$ and sets $k_j = h_3 S_j + h_4 r_{1j}$, $V_j = r_{1j} P$, $U_j = r_{2j} R_v$ and $W_j = r_{2j} P$.
   ξ returns the signature on m with m_w as $\Omega_i = (R_j, W_j, V_j, k_j, X_i, \sigma_i)$ to ADV_3. Clearly the signature Ω_j is valid signature as it satisfies the verification equation.

9. **Designated Verify Oracle** $\left(DV(ID_i)\right)$**:** When ADV_3 submits (ID_{ps}, ID_v, m) and $\Omega_i = (R_j, W_j, V_j, k_j, X_i, \sigma_i)$ to ξ, ξ recovers $(ID_v, R_v, P_{Pub}, l_{1i})$ from $\mathcal{L}_1$ list and then proceeds as follows.

i. If $ID_v \neq ID^*$ then ξ computes $U_j = r_v W_j$ and then recovers the entries $l_{3i} = H_3(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i})$ and $l_{4i} = H_4(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i}, l_{4i})$ from $\mathcal{L}_3$ and $\mathcal{L}_4$ lists.
If these entries does not exists, ξ selects $l_{3i}, l_{4i} \in Z_q^*$ and defines $H_3(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i}) = l_{3i}$ and $H_4(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i}, l_{4i}) = l_{4i}$. ξ then verifies the equation (1) to check the validity of $\Omega_i = (R_j, W_j, V_j, k_j, X_i, \sigma_i)$. It returns the verification result, which is either 1(if Ω_i valid) or 0 (if Ω_i invalid) to ADV_3.

ii. If $ID_v = ID^*$, ξ works on all possible entries $H_3(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i})$ and $H_4(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i}, l_{4i})$ for some U_j.

- For each possible entry $H_3(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i}) = l_{3i}$ and $H_4(m, ID_{ps}, ID_v, U_{ps}, R_{ps}, l_{3i}, l_{4i}) = l_{4i}$ for some U_j, ξ evaluates the equation (1) and returns 1(if Ω_i valid) or 0 (if Ω_i invalid) to ADV_3.
- If the above operation does not lead ξ to return an answer for ADV_3, ξ then returns 0 (invalid) to ADV_3.

10. **Public Verify Oracle:** ADV_3 submits (ID_{ps}, ID_v, m) and $\Omega_i = (R_j, W_j, V_j, k_j, X_i, \sigma_i)$ to ξ. It then performs the same operation as in the simulation of *Designated Verify Oracle*. The only difference is; when ξ judges $\Omega_i = (R_j, W_j, V_j, k_j, X_i, \sigma_i)$ is valid (i.e., returns 1 in the *Designated Verify Oracle*); it returns $Aid = U_j = r_{2j} R_v = r_v W_j = Y_j$ (say) to ADV_3. When ξ judges $\Omega_i = (R_j, W_j, V_j, k_j, X_i, \sigma_i)$ is invalid (i.e., returns 0 in the *D.Verify Oracle*); it returns $\perp$ to ADV_3.

11. ***Forgery:*** Finally ADV_3 outputs $ID_s^*, ID_v^*, m^*, \sigma_i^*$ as its forgery where $\sigma_i^* = (R_i^*, W_i^*, V_i^*, k_i^*)$.

If $ID_i \neq ID_s^*, \xi$ stops simulation. Otherwise, ξ looks up at $\mathcal{L}_{PSK}$ & $\mathcal{L}_{Cuser}$ separately. Let $\sigma_i^{(1)} = (R_i, W_i^{(1)}, V_i, k_i^{(1)})$ denote $\sigma_i = (R_i, W_i, V_i, k_i)$. From Forking Lemma [32], if we have a replay of ξ with same random tape but different choice of H_2, H_3, ADV_3 will output another three signatures $\sigma_i^{(j)} = (R_i, W_i^{(j)}, V_i, k_i^{(j)})$ for $j = 2, 3, 4$, and the following equation holds.

$$\left(k_i^{(j)} P - (R_i + l_{1i}^{(j)} P_{Pub} + X_i) l_{2i}^{(j)}\right) l_{3i}^{-1(j)} = V_i \text{ for } j = 1, 2, 3, 4.$$

By r_i, x_i, s, v_i, we now denote discrete logarithms of R_i, X_i, P_{Pub}, V_i respectively, i.e. $R_i = r_{1i} P, X_i = x_i P, P_{Pub} = sP, V_i = v_i P$. From the above equation, we get four equations as below.

$$\left(k_i^{(j)} - (r_i + l_{1i}^{(j)} s + x_i) l_{2i}^{(j)}\right) l_{3i}^{-1(j)} = v_i^{(j)} \text{ for } j = 1, 2, 3, 4.$$

In these equations, only, r_i, x_i, s, v_i are unknown to ξ. ξ solves these values from the above four linear independent equations and outputs s as the solution of ECDLP.

Theorem 3 (Invisibility): In the ROM, the proposed PF-IDBDPS scheme is invisible against the distinguisher DSTG with the assumption that the ECDLP is hard.

Proof: We construct a ECDLP solver ξ using DSTG. Let ξ be given a random ECDLP instance $(P, aP) \in G$. ξ simulates a challenger for DSTG as mentioned in security model for invisibility.

Initialization Phase and Phase 1: These are same as in Theorem 2. In this phase DSTG performs the oracle queries and these are answered by ξ as in Theorem 2.

Phase 2: DSTG again adaptively performs several oracle queries as it did in Phase 1, subjected to the conditions mentioned in security model of invisibility and finally DSTG outputs a bit $b' \in \{0,1\}$. ξ succeeds in solving the ECDLP and outputs a. This shows that it is infeasible for any fourth party to decide whether a signature was indeed produced by a proxy signer ID_{ps}, designated to ID_v, on message m.

VI. Efficiency analysis

In this section we present the performance analysis of our PF-IDBDPS scheme. We compare our scheme with the relevant schemes [27, 28, 29]. To evaluate the performance of the proposed scheme, we consider various cryptographic operations and their notations which are presented in Table no 2. The conversions of cryptographic operations are taken from the experimental results [33, 34, 35, 36].

Table no 2: Notations and descriptions of various cryptographic operations

Notations	**Descriptions** (in terms of time complexity to execute the following operation)
T_{MM}	modular multiplication
T_{SM}	Scalar multiplication or elliptic curve point multiplication $T_{SM} = 29T_{MM}$
T_{BP}	Bilinear pairing $T_{BP} = 87T_{MM}$
T_{PEX}	Pairing-based exponentiation $T_{PEX} = 43.5T_{MM}$
T_{INV}	Modular inversion operation
T_{H}	simple hash function
T_{MTPH}	Map to point hash function $1T_{MTPH} = 1T_{SM} = 29T_{MM}$
T_{MX}	Modular exponentiation operation $T_{MX} = 240T_{MM}$
T_{PA}	Elliptic curve point addition $T_{PA} = 0.12T_{MM}$

The comparison of our PF-IDBDPS scheme with the existing directed proxy signature schemes [27, 28, 29], in terms of computational complexity and the results are presented in Table no 3. Since the proposed PF-IDBDPS scheme is pairing free, it does not involve any pairing operations. From Table no 3, it is clear that the total computation cost of our PF-IDBDPS scheme requires $430.76T_{MM}$ which is 52.87% less than B. U. Prasad et al. scheme [27], 81.52% less than Ming Yang et al. scheme [29] and 82.33% less than L. Pang et al. scheme [28]. Hence, our PF-IDBDPS scheme is more efficient than the existing directed proxy signature schemes.

Table no 3: Comparison of the proposed PF-IDBDPS scheme with the related scheme.

Scheme	Proxy Signing Cost	Direct Verify cost	Public Verfy Cost	Total Cost
B.U.Prasad et al. [27] (2013)	$3T_{SM} + 1T_{BP} + 1T_{MTPH} + 1T_{PA} + 1T_{PEX} = 246.62T_{MM}$	$2T_{BP} + 3T_{MTPH} + 1T_{SM} + 2T_{PA} + 1T_{PEX} = 333.74T_{M}$	$2T_{BP} + 3T_{MTPH} + 1T_{SM} + 2T_{PA} + 1T_{PEX} = 333.74T_{MM}$	$914.1T_{MM}$
Yang et al. [29] (2011)	$5T_{MX} + 1T_{BP} + 1T_{PEX} = 1330.5T_{MM}$	$6T_{BP} + 1T_{PEX} = 565.5T_{MM}$	$5T_{BP} = 435T_{MM}$	$2331T_{MM}$
Pang et al.[28] (2016)	$4T_{MX} + 5T_{BP} + 1T_{PEX} = 1438.5T_{MM}$	$6T_{BP} + 1T_{PEX} = 565.5T_{MM}$	$5T_{BP} = 435T_{MM}$	$2439T_{MM}$
Our Scheme	$3T_{SM} + 2T_{PA} = 87.24T_{MM}$	$6T_{SM} + 6T_{PA} + 1T_{INV} = 186$	$5T_{SM} + 5T_{PA} + 1T_{INV} = 157.2T_{MM}$	$430.76T_{MM}$

VII. Conclusions

In this paper, we have presented a novel and efficient Identity based directed proxy signature scheme without using bilinear pairings over elliptic curves. The proposed scheme can be applied where the signed message is sensitive to the proxy signature receiver. To the best of our knowledge, the proposed scheme is the first directed proxy signature scheme without pairings in identity based frame work. This scheme is unforgeable and invisible under the hardness of ECDLP. The efficiency analysis shows that our Identity based directed proxy signature scheme is computationally more efficient than the well-known existing directed proxy signature schemes.

References

[1]. Diffe, W., Hellman, M.E., New directions in cryptography. IEEE Transactions in Information Theory, vol. 22, pp. 644-654, 1976.

[2]. Shamir, A., Identity-based Cryptosystems and Signature Schemes. In: Blakley G. R., Chaum D. (Eds.) Advances in Cryptology, Crypto'84, Springer, Berlin, Heidelberg, LNCS 196, pp. 47-53, 1984,.

[3]. Neuman, B.C., Proxy-based authorization and accounting for distributed systems. In: Proceedings of the 13th International Conference on Distributed Computing Systems, pp. 283–291, 1993.

[4]. Foster, I., Kesselman, C., Tsudik, G., Tuecke, S., A security architecture for computational grids. In: Proceeding of the 5th ACM conference on Computers and Communication Security, pp. 83–92, 1998.

[5]. Kim, H., Baek, J., Lee, B., Kim, K., Secret computation with secrets for mobile agent using one-time proxy signatures. In: Proceedings of the Cryptography and Information Security (CIS'01), pp. 845–850, 2001.

[6]. Park, H. U., Lee, L.-Y., Digital nominative proxy signature scheme for mobile communications. In: Proceedings of the Information and Communication Security. LNCS, vol. 2229. Springer, Berlin, pp. 451–455, 2001.

[7]. Mambo, M., Usuda, K., Okamoto, E., Proxy signatures for delegating signing operation. In: 3rd ACM Conference on Computer and Communications Security (CCS'96), New York: ACM Press, pp. 48-57, 1996.
[8]. Kim, S., Park, S., Won, D., Proxy signatures, revisited. In Proc. of ICICS 97, LNCS1334, Springer-Verlag, pp.223-232, 1997.
[9]. Lee, B., Kim, H., Kim, K., Secure mobile agent using strong non-designated proxy signature. In: Information Security and Privacy (ACISP'01) LNCS 2119, Springer-Verlag, pp.474-486, 2001.
[10]. Okamoto, T., Tada, M., Okamoto, E., Extended proxy signatures for smart cards. In: Information Security Workshop (ISW'99), LNCS 1729, Springer-Verlag, pp.247-258, 1999.
[11]. Zhang, K., Threshold proxy signature schemes. Information Security Workshop, Japan, Sep 1997, pp.191-197, 1997.
[12]. Lal, S., Awasthi, A. K., Proxy blind signature scheme. Cryptology e-Print Archive, Report 2003/072, Available at http://eprint.iacr.org, 2003.
[13]. Lal, S., Awasthi, A. K., A scheme for obtaining a warrant message from the digital proxy signatures. Cryptology e-Print Archive, Report 2003/073, Available at *http://eprint.iacr.org*., 2003.
[14]. Park, H. U. Lee, I.Y., A digital nominative proxy signature scheme for mobile communications. In: Information and Communication Security (ICICS'01), LNCS 2229, Springer- Verlag, pp.451-455, 2001.
[15]. Shum, K., Wei, V. K., A strong proxy signature scheme with proxy signer privacy protection. In Eleventh IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002.
[16]. Lim, C.H., Lee, P.J., Modified Maurer-Yacobi's scheme and its applications. In: Advances in Cryptology – AUSCRYPT'92, LNCS vol.718, pp. 308–323, 1992.
[17]. Lal, S., Kumar, M., A directed signature scheme and its applications. Available at: http://arxiv.org/abs/cs/0409035. 2004.
[18]. Laguillaumie, F., Paillier, P., Vergnaud, D., Universally convertible directed signatures, in: Advances in Cryptology - ASIACRYPT'05, LNCS 3788, pp. 682–701, 2005.
[19]. Lu, R., Cao, Z., A directed signature scheme based on RSA assumption. International Journal of Network Security, 2 (3), 182–421, 2006.
[20]. Ismail, E.S., Abu-Hassan, Y., A Directed Signature Scheme Based on Discrete Logarithm Problems. Journal Teknologi, 47 (C), 37–44, 2007.
[21]. Wei, Q., He, J., Shao, H., Directed Signature Scheme and its Application to Group Key Initial Distribution. In: ICIS-2009. ACM, Seoul, Korea, pp. 24–26, 2009.
[22]. Ramlee, N.N., Ismail, E.S., A New directed signature scheme with hybrid problems. Appl. Math. Sci. 7 (125), 6217–6225, 2013.
[23]. Sun, X., Li, J., Chen, G., Yung, S., Identity-Based Directed Signature Scheme from Bilinear Pairings. Available at https:// eprint.iacr.org/2008/305.pdf. 2008.
[24]. Zhang, J., Yang, Y., Niu, X., Efficient Provable Secure ID-Based Directed Signature Scheme without Random Oracle. In: Yu, W., He, H., Zhang, N. (Eds.), Advances in Neural Networks-ISSN 2009. Springer, Berlin, Heidelberg. LNCS, pp.318–327, 2009..
[25]. Uma Prasada Rao, B., Vasudeva Reddy, P., Gowri, T., An efficient ID-Based Directed signature scheme from bilinear pairings. Available at https://eprint.iacr.org/2009/617.pdf. 2009.
[26]. Ku, J., Yun, D., Zheng, B., Wei, S., An efficient ID-Based Directed signature scheme from optimal Eta pairing, Computational Intelligence and Intelligent Systems, Communications in Computer and Information Science. Springer, Berlin, Heidelberg, pp. 440-448, 2012.
[27]. Vasudeva Reddy, P., Umaprasad, B., Gowri, T., ID-based directed proxy signature scheme from bilinear pairings, Journal of Discrete Mathematical Sciences and Cryptography, vol. 13, no 5, pp. 487-500, 2010.
[28]. Pang L., Hu Y., Zhou X., Wang Y., Li H., Directed proxy signature with fast revocation proven secure in the standard model, IET Information Security, 2016.
[29]. Ming Y., Wang Y.M., Directed Proxy signature in the standard model, Journal of Shanghai Jiaotong University (Sci.), vol. 16 (6), pp. 663-671, 2011.
[30]. Anoop, M.S., Elliptic curve cryptography: An implementation Guide. http://www.infosecwriters.com/text_resources/pdf/Elliptic_Curve_AnnopMS.pdf.
[31]. Huang, X., Susilo, W., Mu, Y. & Wu, W., Proxy signature without random oracles. In J. Cao, I. Stojmenovic, X. Jia & S. Das (Eds.), The Second International Conference on Mobile Ad Hoc and Sensor Networks (MSN 2006) (pp. 473-484). Berlin, Germany: Springer-Verlag, 2006.
[32]. Pointcheval, D., Stern, J., Security arguments for digital signatures and blind signatures. Journal of Cryptology, vol. 13 (3), 361–369, 2000.
[33]. Barreto, P., Kim, H.Y., Lynn, B., et al., Efficient Algorithms for Pairing based Cryptosystems. LNCS, Springer-Verlag 2442, 354–368, 2002.
[34]. Cao, X., Kou, W., Du, X., A Pairing –free Identity Based Authenticated Key Agreement Protocol with Minimal Message Exchanges. Information Sciences, vol. 180 (15), 2895–2903,2010..
[35]. Tan, S.Y., Heng, S.H., Goi, B.M., Java Implementation for Pairing-based Cryptosystems. In: Taniar D., Gervasi O., Murganate B., Pardede E., Apduhan B. O. (Eds.), Computational Science and its Applications- ICCSA-2010, LNCS, Springer, Berlin, Heidelberg, 6019, pp. 188-198, 2010.
[36]. MIRACL Library. Available at http://certivox.org/display/EXT/MIRACL.http://www.iosrjournals.org/manuscript-guidelines.html.

R. R. V. Krishna Rao." Efficient and Secure Pairing-free ID-based Directed Proxy Signature Scheme." IOSR Journal of Mathematics (IOSR-JM) 14.6 (2018): 73-84.