Journal of King Saud University – Computer and Information Sciences (2013) **25**, 51–61

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

Provably secure certificateless strong designated verifier signature scheme based on elliptic curve bilinear pairings

SK Hafizul Islam *, G.P. Biswas

Department of Computer Science and Engineering, Indian School of Mines, Dhanbad 826 004, Jharkhand, India

Received 10 March 2012; revised 1 May 2012; accepted 26 June 2012
Available online 5 July 2012

KEYWORDS

Certificateless cryptography;
Elliptic curve cryptosystem;
Designated verifier;
Digital signature;
Bilinear pairing

Abstract Diffie and Hellman first invented the public key cryptosystem (PKC) wherein the public key infrastructure (PKI) is used for the management of public keys; however, the PKI-based cryptosystems suffer from heavy management trouble of public keys and certificates. An alternative solution to the PKI is Shamir's identity-based cryptosystems (IBC), which eliminate the need of public key certificates; however, the most important shortcoming of IBC is the *key escrow problem*. To cope with these problems, Al-Riyami and Paterson proposed a novel scheme of certificateless PKC (CL-PKC) by combining the advantages of PKI and IBC. Since then, several certificateless signature schemes have been designed and most of them have been analyzed and proven insecure against different types of adversaries. Besides, the researchers have given very less attention to the certificateless strong designated verifier signature (CL-SDVS) scheme. Therefore, we proposed a CL-SDVS scheme using elliptic curve bilinear parings in this paper. Our scheme, which is provably secure in the random oracle model with the intractability of BDH and CDH assumptions, supports all desirable security necessities of the CL-SDVS scheme such as *strongness*, *source hiding* and *non-delegatability*. The rigorous security analysis and comparison with others guarantee the better performance of the proposed scheme.

1. Introduction

Diffie and Hellman (1976) first invented the public key cryptosystem (PKC) and after which a public key infrastructure (PKI) has been developed, where different methods are used for generating and maintaining the public keys using the corresponding public key certificates. However, the PKI-based cryptosystems mainly suffer from problems such as (1) the sender must authenticate the public key of the receiver by verifying the corresponding certificate prior to the communication, thereby requiring additional computational cost; (2) PKI-based systems have the serious problems of certificate creation, storage, revocation, delivery, etc. These problems can prevail over the identity-based cryptosystem (IBC) (Shamir, 1984), which can avoid the need of public key certificate. In IBC, a user's identity is used as public key and a trusted third party called PKG (private key generator) generates the corresponding private key by binding the user's identity and its own

* Corresponding author. Tel.: +91 8797369160; fax: +91 326 2296563.
E-mail addresses: hafi786@gmail.com, hafizul.ism@gmail.com (SK Hafizul Islam), gpbiswas@gmail.com (G.P. Biswas).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

1319-1578 © 2012 King Saud University. Production and hosting by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jksuci.2012.06.003

private key. Although Shamir proposed a smartcard-based digital signature scheme using IBC, no practical identity-based encryption/decryption scheme (IBE) was devised. Boneh and Franklin (2001) first designed a practical IBE scheme in the elliptic curve group (Miller, 1985; Koblitz, 1987) using bilinear pairings. In IBE, a public key can be revoked easily by binding the date to the public key (Boneh and Franklin, 2001), and there is no need for the exchange of public keys before the communication takes place. Despite the advantages of IBE, the inherent problem, called *private key escrow*, makes it unsuitable for real-life applications where privacy of users is of great concern in all respects. In IBE, the assumption for a PKG to be fully trusted is not possible for open networks. Because the PKG knows the private keys of all users, a malicious PKG can impersonate any user in the system easily.

With the advantages of both IBE and PKI, Al-Riyami and Paterson (2003) proposed the certificateless public key cryptosystem (CL-PKC) that has received a lot of attention in recent years. The CL-PKC eliminates the *private key escrow problem* of IBE and the need of public key certificate as found in PKI. In CL-PKC, each user has a full private key that consists of two secrets comprising the identity-based private key generated by PKG and the PKI-based private key chosen by the user himself. The PKG has no ability to get access to the full private key of the user, and thus the man-in-the-middle attack is impossible by the dishonest PKG.

The designated verifier signature (DVS) scheme (Jakobsson et al., 1996) allows the original signer, *Alice* (say) to generate a signature that only be verified by a designated verifier *Bob* (say). Further, in any DVS scheme, *Bob* can prove to a third party that the signature was generated by *Alice*. As a result, the public verification of DVS scheme annihilates the signer's privacy protection. To meet this goal, another signature scheme, called strong DVS (SDVS) has been proposed by Jakobsson et al. in the same paper (Jakobsson et al., 1996). In this scheme, *Bob* is unable to convince an outsider that the signature was generated by *Alice* or himself. This is because *Bob* can compute an identical signature that cannot be distinguished from the signatures created by *Alice,* and the private key of *Bob* is strictly required in the verification phase. It means the SDVS scheme satisfies the *strongness* and *repudiation* properties as required in other secure cryptosystems.

1.1. Literature review

Recently, several certificateless signature (CLS) schemes have been proposed in the literature. The first CLS scheme is proposed by Al-Riyami and Paterson in their seminal paper (Al-Riyami and Paterson, 2003), but the protocol is not secure as demonstrated by Huang et al. (2005), and then they built a model for the CLS scheme. Gorantla and Saxena (2005) proposed a new efficient and secure CLS scheme as they have claimed, but Cao et al. (2006) proved that it could not resist the *public key replacement* attack. Yum and Lee (2004) define the new construction of the CLS scheme however, Hu et al. (2006) demonstrated that the scheme was not secure against the *public key replacement* attack. To remove the security flaws of Yum and Lee (2004), Hu et al. (2006) proposed an improved CLS scheme. Two efficient CLS schemes (Yap et al., 2006; Choi et al., 2007) were also presented and their security is proven based on the model proposed in Huang's et al. (2005), but Zhang and Feng (2006) demonstrated that the former scheme (Yap et al., 2006) was vulnerable to the *public key replacement* attack. Zhang et al. (2006) proposed an improved CSL model using the model proposed in (Huang et al., 2005). Xu et al. (2008) proposed a CLS scheme for mobile wireless cyber-physical systems, and Chen et al. (2007) presented an efficient and secure CLS scheme for the environment where immediate revocation of the public keys is required. Yang et al. (2007) designed a certificateless universal designated verifier signature (CL-UDVS) scheme, which is proven to be secure in the random oracle model (Bellare and Rogaway, 1993) against different types of adversaries. However, Guozheng and Fan (2009) showed that the Xu et al. (2008) scheme is vulnerable to a *malicious PKG attack* and Yang et al. (2007) scheme is not secure against the *public key replacement* and *malicious PKG*attacks. Zhang and Zhang (2008) proposed an efficient CLS scheme, which is secure in the random oracle model based on the hardness assumption of BDH problem whereas Li and Liu (2011) proposed another CLS scheme and provided only informal security analysis.

Recently, certificateless short signature schemes have received great attention due to its shorter length. The short signature schemes are appropriate for low-bandwidth, low-storage and low-computation environments such as bar-coded digital signature on postage stamps. In 2006, Huang et al. (2006) first introduced the security notion and construction of a certificateless short designated verifier signature (CL-sDVS) scheme with the intractability of Gap Bilinear Diffie–Hellman (GBDH) problem (Boneh and Franklin, 2001) in the random oracle model. Unfortunately, their scheme is vulnerable to *malicious PKG attacks*. Du and Wen (2007), and Chen et al. (2008) proposed two CL-sDVS schemes based on parings and analyzed that the schemes were provably secured. However, Fan et al. (2009) proved that the scheme (Du and Wen, 2007) did not resist the *public key replacement* attack and then developed a modified scheme to improve the security of the earlier scheme. Tso et al. (2011) and Choi et al. (2011) independently proposed two provably secure CL-sDVS schemes, but the latter scheme is proven vulnerable against the *public key replacement* attack (Tian et al., 2011).

1.2. Motivations and contributions

Based on the pioneer work of Al-Riyami and Paterson (2003), many of CLS and CL-short signature schemes have been proposed in recent years; however, most of the previous schemes cannot meet desired security and computation efficiency. From the literature, it can be seen that the researchers have been given very less attention to the certificateless strong designated verifier signature (CL-SDVS) scheme. In 2009, Hongzhen and Qiaoyan proposed a CL-DVS scheme using pairings and states that is secure against all adversaries, and satisfies all the necessary properties of DVS scheme. However, the computation cost of the scheme is high as three parings and one paring-based exponentiation computation are required for the signature generation and verification. Recently, Yang et al. (2009) and Xiao et al. (2010) independently designed two CL-SDVS schemes based on elliptic curve bilinear pairings. However, Zhang and Xie (2011) analyzed that Xiao et al. scheme is vulnerable to both the *public key replacement* and *malicious PKG* attacks.

In this paper, we designed a CL-SDVS scheme based on ECC and bilinear pairings. The formal security analysis of our scheme proves that it is strongly secure against various adaptive chosen messages and identity adversaries in the random oracle model. We also compared our scheme with a number of related CLS schemes in terms of security and computation efficiency, and found improved performance.

1.3. Roadmap of the paper

The remainder of the paper is structured as follows. Section 2 describes some preliminaries and Section 3 describes the definition and attack model of a CL-SDVS scheme. Section 4 explains the proposed CL-SDVS scheme and Section 5 discusses the formal security analysis of our scheme. The comparisons of the proposed scheme with other are presented in Section 6 and, finally, Section 7 concludes the paper.

2. Technical backgrounds

This section describes the concept of bilinear parings and some related computational problems as required in the paper.

2.1. Bilinear pairings

Let G_q be an additive cyclic group with prime order $q(q \geqslant 2^k)$, G_m be a multiplicative group of the same prime order q. Let $\hat{e} : G_q \times G_q \rightarrow G_m$ be an admissible bilinear mapping that satisfies the following properties:

- *Bilinearity:* The bilinear map $\hat{e} : G_q \times G_q \rightarrow G_m$ is said to be bilinear if for all P, $Q \in G_q$ and $a, b \in_R Z_q^*$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ holds.
- *Non-degeneracy:* There exists P, $Q \in G_q$ such that $\hat{e}(P, Q) \neq 1_m$, where 1_m is an identity element of G_m. Note that the map $\hat{e}$ does not send all pairs in $G_q \times G_q$ to the identity in G_m that is if P is a generator of the group G_q then $\hat{e}(P, P)$ is a generator of the group G_m.
- *Computability:* There must be an efficient algorithm, which can compute $\hat{e}(P, Q)$ for all P, $Q \in G_q$.

A bilinear map is called an admissible bilinear map if it satisfies the three properties defined above. In general, G_q is a group of points on an elliptic curve, and G_m is a multiplicative subgroup of a finite field E/F_q. The map $\hat{e}$ will be derived either from the modified Weil pairing or from Tate pairing over a finite field. A more comprehensive description about bilinear pairings, selection of suitable parameters, elliptic curves and these groups can be found in Boneh and Franklin (2001) for efficiency and security.

Bilinear Diffie–Hellman parameter generator (BDH-PG): A BDH-PG $\mathscr{G}$ is defined as a probabilistic polynomial time bounded algorithm that takes the security parameter $k \in \mathbb{Z}^+$ as input and outputs a uniformly random tuple $(q, \hat{e}, G_q, G_m, P)$ of bilinear parameters.

2.2. Complexity assumptions

In this section, we define the following computational problems, which are assumed hard to break by any polynomial time bounded algorithm, on the elliptic curve group.

Definition 1 (*Elliptic curve discrete logarithm problem (ECDLP)*). Given a random instance $(P, Q) \in G_q$, find an integer $a \in_R Z_q^*$ such that $Q = aP$.

Definition 2 (*Computational Diffie–Hellman (CDH) problem*). Given a random instance $(P, aP, bP) \in G_q$ for any $a, b \in_R Z_q^*$, computation of abP is hard to the group G_q.

Definition 3 (*Computational Diffie–Hellman (CDH) assumption*). A probabilistic polynomial time bounded adversary $\mathscr{A}$ is said to break the CDH problem with negligible probability, if for a given random instance $(P, aP, bP) \in G_q$ of the CDH problem, where $a, b \in_R Z_q^*$ are unknown to $\mathscr{A}$, the advantage $Adv_{\mathscr{A}, G_q}^{CDH} = \Pr\left[\mathscr{A}(P, aP, bP) = abP : a, b \in_R Z_q^*\right]$ of $\mathscr{A}$ in solving CDH problem is negligible.

Definition 4 (*Bilinear Diffie–Hellman (BDH) problem*). Given a random instance $(P, aP, bP, cP) \in G_q$ and for any $a, b, c \in_R Z_q^*$, it is impossible to compute $\hat{e}(P, P)^{abc}$.

Definition 5 (*Bilinear Diffie–Hellman (BDH) assumption*). If $\mathscr{G}$ is a BDH parameter generator, the polynomial time bounded adversary $\mathscr{A}$ breaks the BDH problem with negligible probability, if for a given random instance $(G_q, G_m, e, P, aP, bP, cP)$, where the tuple (G_q, G_m, e) is the output of the BDH-PG $\mathscr{G}$ for sufficiently large security parameter $k \in \mathbb{Z}^+$, $(P, aP, bP, cP) \in G_q$ and $a, b, c \in_R Z_q^*$, then the advantage $Adv_{G, \mathscr{A}}^{BDH}(k) = \Pr\left[\mathscr{A}(P, aP, bP, cP) = \hat{e}(P, P)^{abc} : a, b, c \in_R Z_q^*\right]$ of $\mathscr{A}$ in solving BDH problem is negligible.

3. Model of CL-SDVS scheme

3.1. Definition of CL-SDVS scheme

In any CL-SDVS scheme, three entities are involved, namely the PKG, a signer and a designated verifier. The CL-SDVS scheme consists of the following algorithms:

- *Setup:*It takes the security parameter $k \in \mathbb{Z}^+$ as inputs and then generates PKG's master secret key msk and the system's parameter Ω.
- *Partial-Private-Key-Extract:* It takes msk, and an identity ID_i of the user i as inputs and outputs the partial private key D_i for the user i.
- *Set-Secret-Value:* It takes the system's parameter Ω, an identity ID_i of the user i as inputs and outputs a random number x_i as the secret value of the user ID_i.
- *Set-Private-Key:* It takes the system's parameter Ω, partial private key D_i, secret value x_i of the user ID_i and outputs the full private key sk_i of ID_i.
- *Set-Public-Key:*It takes the system's parameter Ω, secret value x_i of ID_i as inputs and outputs the full public key pk_i of ID_i.
- *CL-SDVS-Sign:*It takes the system's parameter Ω, full private key sk_i of the signer ID_i, a message $m \in \{0, 1\}^*$, full public key pk_j of the designated verifier ID_j as inputs and then outputs a signature σ on the message m.

- *CL-SDVS-Verify:*It takes the system's parameter Ω, full public key pk_i of the signer ID_i, private key sk_j of the designated verifier ID_j and a message-signature pair (σ, m) as inputs and then outputs 'true' if the signature σ is valid. Otherwise, outputs 'false'.
- *CL-SDVS-Simulation:*The designated verifier ID_j executes this algorithm to produce an identical signature σ', which is indistinguishable from the signature σ generated by the original signer ID_i.

3.2. Attack model of CL-SDVS scheme

There are three types of adversaries with different capabilities in any certificateless CL-PKC system: the Type I adversary $\mathscr{A}_I$ is an outsider who can access the system's parameters only, the Type II adversary $\mathscr{A}_{II}$ is a dishonest user, and the Type III adversary $\mathscr{A}_{III}$ is a malicious PKG. The description of these adversaries is given below.

- *Type I adversary* $\mathscr{A}_I$.This type of adversary only knows the public parameters and tries to obtain PKG's master private key to impersonate any user in the system.
- *Type II adversary* $\mathscr{A}_{II}$.This type of adversary (public key replacement attacker) has no knowledge about users' partial-private-keys, but can replace users' public keys with a value of his own choice.
- *Type III adversary* $\mathscr{A}_{III}$.This type of adversary (malicious PKG) has the knowledge about users' partial-private-keys, but cannot replace users' public keys.

The resilience of the attack made by the adversary $\mathscr{A}_I$ is related to the security of the PKG key generation and this attack is not a strong attack, while the adaptive chosen message and identity attack caused by other two adversaries are considered as powerful attacks. The existential unforgeability of a CL-SDVS scheme against the adaptive chosen message and identity attack is defined (Yum and Lee, 2004; Hu et al., 2006; Yap et al., 2006; Zhang et al., 2006; Zhang and Feng, 2006; Huang et al., 2006; Choi et al., 2007; Du and Wen, 2007; Zhang and Zhang, 2008; Chen et al., 2008) by the following challenge-response games Game 1 and Game 2 played between a challenger $\mathscr{C}$ and the adversary $\mathscr{A}_{II}$ or $\mathscr{A}_{III}$.

Game 1: The challenger $\mathscr{C}$ plays this game with an adaptive chosen message and identity adversary $\mathscr{A}_{II}$ for breaking CL-SDVS scheme.

- *Setup:* The challenger $\mathscr{C}$ takes the security parameter $k \in \mathbb{Z}^+$ and runs the setup algorithm to generate the PKG's secret key *msk* and the system's parameter Ω. Then $\mathscr{C}$ sends Ω to $\mathscr{A}_{II}$ while keeping *msk* secret.
- *Hash queries*: The adversary $\mathscr{A}_{II}$ can request the hash value for any input.
- *Partial-Private-Key-Extract queries:* The adversary $\mathscr{A}_{II}$ can ask for the partial private key of the user ID_i, $\mathscr{C}$ then computes the corresponding D_i and sends it to $\mathscr{A}_{II}$.
- *Set-Secret-Value queries:* The adversary $\mathscr{A}_{II}$ can ask for the secret value of the user ID_i, $\mathscr{C}$ then computes the corresponding secret value x_i and sends it to $\mathscr{A}_{II}$.
- *Set-Public-Key queries:* The adversary $\mathscr{A}_{II}$ can asks for the full public key of the user ID_i, $\mathscr{C}$ then computes the corresponding pk_i and send it to $\mathscr{A}_{II}$.
- *Public-Key-Replacement queries:* The adversary $\mathscr{A}_{II}$ can choose a new public key P_i' for the user ID_i and asks to replace the old public key P_i by P_i'. Then $\mathscr{A}_{II}$ sets P_i' as the new public key of ID_i and $\mathscr{C}$ records it.
- *CL-SDVS-Sign queries:* The adversary $\mathscr{A}_{II}$ can ask for a signature of the chosen message m_i with signer's identity ID_i and designated verifier's identity ID_j. Then $\mathscr{C}$ executes *CL-SDVS-Sign* algorithm and outputs a message-signature pair (m_i, σ_i) and returns it to $\mathscr{A}_{II}$.
- *CL-SDVS-Verify queries:* The adversary $\mathscr{A}_{II}$ can ask for the verification of a message-signature pair (m_i, σ_i) with signer's identity ID_i and verifier's identity ID_j, $\mathscr{C}$ then runs *CL-SDVS-Verify* algorithm and outputs true if (m_i, σ_i) is valid. Otherwise, outputs false.
- *Forgery:* At the end of game $\mathscr{A}_{II}$ outputs a valid forged tuple (m^*, σ^*) with signer's identity ID_i and verifier's identity ID_j. The adversary $\mathscr{A}_{II}$ can win the above game if the following holds:
  (a) The signature σ^* is a valid signature of the message m^* with signer's identity ID_i and verifier's identity ID_j.
  (b) The message m^* has never been submitted to the *CL-SDVS-Sign* oracle with signer's identity ID_i and verifier's identity ID_j.
  (c) The identities ID_i and ID_j have never been submitted to the oracle *Partial-Private-Key-Extract* and *Set-Secret-Value-Extract*.

Definition 6. The probability that an adversary $\mathscr{A}_{II}$ can forge the CL-SDVS scheme under the adaptively chosen message and identity is defined as $Adv^{CMA}_{CL-SDVS,\mathscr{A}_{II}}(k)$. The CL-SDVS signature scheme is secure against the polynomial time bounded adversary $\mathscr{A}_{II}$, if the probability of success to win game Game 1 defined above is negligible, i.e., $Adv^{CMA}_{CL-SDVS,\mathscr{A}_{II}}(k) \leqslant \varepsilon$.

Game 2: This challenge-response game is played between the challenger $\mathscr{C}$ and an adaptive chosen message and identity adversary $\mathscr{A}_{III}$.

- *Setup:* The challenger $\mathscr{C}$ takes the security parameter $k \in \mathbb{Z}^+$ and runs the setup algorithm to generate the PKG's secret key *msk* and the system's parameter Ω. Then $\mathscr{C}$ sends Ω to $\mathscr{A}_{III}$ while keeping the *msk* secret.
- *Hash queries*: The adversary $\mathscr{A}_{III}$ can request the hash value for any input.
- *Partial-Private-Key-Extract queries:* The adversary $\mathscr{A}_{III}$ can ask for the partial private key of the user ID_i, $\mathscr{C}$ then computes the corresponding D_i and sends it to $\mathscr{A}_{III}$.
- *Set-Secret-Value queries:* The adversary $\mathscr{A}_{III}$ can ask for the secret value of the user ID_i, $\mathscr{C}$ then computes the corresponding secret value x_i and sends it to $\mathscr{A}_{III}$.
- *Set-Public-Key queries:* The adversary $\mathscr{A}_{III}$ can ask for the full public key of the user ID_i, $\mathscr{C}$ then computes the corresponding pk_i and sends it to $\mathscr{A}_{III}$.
- *CL-SDVS-Sign queries:* The adversary $\mathscr{A}_{III}$ can ask for a signature for the chosen message m_i with signer's identity ID_i and verifier's identity ID_j. Then $\mathscr{C}$ executes *CL-SDVS-Sign* algorithm to output a message-signature pair (m_i, σ_i) and then returns it to $\mathscr{A}_{III}$.

- *CL-SDVS-Verify queries:* The adversary $\mathscr{A}_{III}$ can ask for the verification of a message-signature pair (m_i, σ_i) with signer's identity ID_i and verifier's identity ID_j, C then runs *CL-SDVS-Verify* algorithm and outputs true if (m_i, σ_i) is valid. Otherwise, outputs false.
- *Forgery:* At the end of this game, $\mathscr{A}_{III}$ outputs a forged tuple (m^*, σ^*) with signer's identity ID_i and verifier's identity ID_j. The adversary $\mathscr{A}_{III}$ can win the above game if the following holds:
  (a) The signature σ^* is the valid signature of the message m^* with signer's identity ID_i and verifier's identity ID_j.
  (b) The message m^* has never been submitted to the *CL-SDVS-Sign* oracle with signer's identity ID_i and verifier's identity ID_j.
  (c) The identities ID_i and ID_j have never been submitted to the oracles *Partial Set-Secret-Value* and *Public-Key-Replacement*.

Definition 7. Now, we define $Adv^{CMA}_{CL-SDVS,\mathscr{A}_{III}}(k)$ as the probability that an adaptively chosen message and chosen identity adversary $\mathscr{A}_{III}$ can win the Game 2. The CL-SDVS scheme is secure against $\mathscr{A}_{III}$ adversary if $Adv^{CMA}_{CL-SDVS,\mathscr{A}_{III}}(k)$ is negligible, i.e.,$Adv^{CMA}_{CL-SDVS,\mathscr{A}_{III}}(k) \leqslant \varepsilon$.

Definition 8. The CL-SDVS scheme is existentially unforgeable against adaptive chosen message and chosen identity attack if it is secure against the adversaries $\mathscr{A}_{II}$ and $\mathscr{A}_{III}$, i.e., if both $Adv^{CMA}_{CL-SDVS,\mathscr{A}_{II}}(k) \leqslant \varepsilon$ and $Adv^{CMA}_{CL-SDVS,\mathscr{A}_{III}}(k) \leqslant \varepsilon$ hold simultaneously.

4. The proposed CL-SDVS scheme

In this section, we describe our secure and computationally efficient CL-SDVS scheme, which is developed based on elliptic curve bilinear pairings. We assume that *Alice*, the original signer with identity ID_A, has the full private key $sk_A = (D_A, x_A)$ and public key $pk_A = (Q_A, P_A)$, and *Bob* is the designated verifier who has the identity ID_B, full private key $sk_B = (D_B, x_B)$ and full public key $pk_B = (Q_B, P_B)$. The proposed CL-SDVS scheme consists of the following algorithms:

- *Setup:* This algorithm takes a security parameter $k \in \mathbb{Z}^+$ as input, and returns a system's parameter and a master key. Given k, the PKG does the following:
  (a) Choose an additive elliptic curve cyclic group $(G_q, +)$ of prime order $q \geqslant 2^k$, a multiplicative group $(G_m, \cdot)$ of the same prime order q and an admissible bilinear pairing map $\hat{e} : G_q \times G_q \to G_m$ as defined in Boneh and Franklin (2001).
  (b) Choose a number $s \in_R Z_p^*$ and a generator point P of the group G_q, and compute $P_0 = sP$, where $(s, P_0 = sP)$ is the private/public key pair of PKG.
  (c) Choose three secure and one-way cryptographic hash functions $H_1 : \{0,1\}^* \to G_q$, $H_2 : \{0,1\}^* \times G_q \to Z_q^*$ and $H_3 : \{0,1\}^* \times G_m \times G_m \to Z_q^*$. The hash function H_1 is called the *map-to-point* hash function (Boneh and Franklin, 2001), which is used to map a random binary string of arbitrary length to a point of the elliptic curve group G_q, while H_2 and H_3 are the general hash functions (i.e., SHA family, MD family, etc.).
  (d) The system parameters, represented by the vector $\Omega = \{G_q, G_m, \hat{e}, q, P, P_0, H_1, H_2, H_3\}$, is known to everyone associated with the system; however, the master key $msk = s$ is private and only known to PKG.
- *Partial-Private-Key-Extract:* This algorithm takes the system's parameter Ω, the master private key s and a user's identity $ID_i \in \{0,1\}^*$ as inputs and generates the partial private key for the user ID_i as follows:
  (a) Compute $Q_i = H_1(ID_i)$.
  (b) Compute the partial private key $D_i = sQ_i$ and send to the user ID_i through a secure and authenticated channel.
- *Set-Secret-Value:* The user ID_i picks a number $x_i \in_R Z_q^*$, computes $P_i = x_iP$ and sets x_i as his secret value.
- *Set-Private-Key:* The user ID_i sets $sk_i = (D_i, x_i)$ as his full private key.
- *Set-Public-Key:*The user ID_i sets $pk_i = (Q_i, P_i)$ as his full public key.
- *CL-SDVD-Sign:* The signer *Alice* chooses a number $r \in_R Z_q^*$ and then generates a designated verifier signature on a message $m \in \{0,1\}^*$ for the designated verifier *Bob* in the following way:
  (a) Compute $g = \hat{e}(D_A, Q_B)$ and $h = H_2(m, x_AP_B)$.
  (b) Compute $R = \hat{e}(P_0, Q_B)^{rh}$ and $t = H_3(m, g, R)$.
  (c) Compute $V = rhP_0 + tD_A$ and $S = \hat{e}(V, Q_B)$.
  (d) Output the signature (R, S) and send to *Bob* for verification.
- *CL-SDVD-Verify:* On receiving the signature (R, S) and the message m, the verifier *Bob* does the following:
  (a) Compute $g = \hat{e}(Q_A, D_B)$ and $t = H_3(m, g, R)$.
  (b) Compute $S'' = R \cdot g^t$ and check whether $S'' = S$ holds. If so, *Bob* accepts the signature (R, S), otherwise he rejects the signature.
- *CL-SDVS-Simulation: Bob* can efficiently produce a valid signature $(\widehat{R}, \widehat{S})$ on a message m intended for himself as follows:
  (a) Choose a number $\hat{r} \in_R Z_q^*$, compute $\hat{g} = \hat{e}(Q_A, D_B)$ and $\hat{h} = H_2(m, x_BP_A)$.
  (b) Compute $\widehat{R} = \hat{e}(P_0, Q_B)^{\hat{r}\hat{h}}$ and $\hat{t} = H_3(m, \hat{g}, \widehat{R})$.
  (c) Compute $\widehat{S} = \widehat{R} \cdot g^{\hat{t}}$ and output the signature $(\widehat{R}, \widehat{S})$.
  It can be verified that the simulated $(\widehat{R}, \widehat{S})$ is a valid signature on a given message m.
- *Correctness of the proposed scheme:* Since $g = \hat{e}(D_A, Q_B) = \hat{e}(Q_A, D_B)$ and $t = H_3(m, g, R)$. Therefore, we obtain

$$
\begin{aligned}
S &= \hat{e}(V, Q_B) \\
&= \hat{e}(rhP_0 + tD_A, Q_B) \\
&= \hat{e}(rhP_0, Q_B)\hat{e}(tD_A, Q_B) \\
&= \hat{e}(P_0, Q_B)^{rh}\hat{e}(D_A, Q_B)^t \\
&= \hat{e}(P_0, Q_B)^{rh}\hat{e}(Q_A, D_B)^t \\
&= R \cdot g^t
\end{aligned}
$$

This proves the correctness and soundness of the proposed CL-SDVS scheme. □

5. Security analysis of the proposed CL-SDVS scheme

In this section, we present the security proof of the proposed CL-SDVS scheme against different adversaries with different powers in the random oracle model (Bellare and Rogaway,

1993). We also show that our scheme satisfies other security properties such as *strongness*, *source hiding* and *non-delegatability* of an SDVS scheme (Yang et al., 2009; Xiao et al., 2010). Note that the security of the proposed scheme depends on the hardness assumption of ECDLP, BDH and CDH problems in the elliptic curve group.

5.1. Unforgeability against Type I adversary

Theorem 1. *The proposed CL-SDVS scheme is existentially unforgeable against Type I adversary $\mathscr{A}_I$ with the hardness assumption of ECDLP in the elliptic curve group.*

Proof. The adversary $\mathscr{A}_I$ has the knowledge about system's parameter $\Omega = \{G_q, G_m, \hat{e}, q, P, P_0, H_1, H_2, H_3\}$, where $P_0 = sP$. The adversary $\mathscr{A}_I$ can impersonate any user ID_i in the system if the master private key $msk = s$ is known to him. Note that there is no use of certificate of the public key $P_i = x_iP$; therefore, knowing the maser private key msk, A_I can impersonate the user ID_i by choosing a random x_i of his choice. To know s, A_I tries to obtain it from the PKG's public key P_0, however, it is computationally infeasible to derive s from P_0 due to the difficulties of solving the ECDLP in the elliptic curve group G_q. As far as we know, there is no polynomial time bounded algorithm so far, which can solve the ECDLP problem with non-negligible advantage. Thus, the proposed CL-SDVS scheme is secure against the adversary $\mathscr{A}_I$ under the ECDLP problem. □

5.2. Unforgeability against Type II adversary (resilience against public key replacement attack)

Theorem 2. *The proposed CL-SDVS scheme is existentially unforgeable under the adaptive chosen message and identity attacks against the adversary $\mathscr{A}_{II}$ in random oracle model if the BDH problem is hard to break by a polynomial time bounded algorithm.*

Proof. Assume that the probabilistic polynomial time bounded adversary $\mathscr{A}_{II}$ who can breach our CL-SDVS scheme under the adaptively chosen message and identity attacks in the random oracle model with probability ε and within a time bound t, then we show that there exists an algorithm $\mathscr{C}$ that can use $\mathscr{A}_{II}$ as black box to solve the BDH problem with a non-negligible probability and within the polynomial time t. That is, for given a random instance $(P, aP, bP, cP) \in G_q$ and for the unknowns $a, b, c \in_R Z_q^*$, $\mathscr{C}$ can compute $\hat{e}(P,P)^{abc}$. To solve the BDH problem, $\mathscr{C}$ sets $\Omega = \{G_q, G_m, \hat{e}, q, P, P_0 = aP, H_1, H_2, H_3\}$, returns it to $\mathscr{A}_{II}$ and then replies in the following way:

- *Hash queries to H_1*: Assume that $\mathscr{A}_{II}$ is allowed to ask at most q_{H1} times H_1 queries and $\mathscr{C}$ maintains an initial-empty list L_{H1}^{list} that consists of the tuple of the form (ID_i, Q_i, d_i) to avoid any collision and inconsistency. Suppose the adversary $\mathscr{A}_{II}$ asks a H_1 query with ID_i, $\mathscr{C}$ then responds as follows:
  (a) If $ID_i = ID_A$, then return $Q_i = H_1(ID_i) = bP$ and add the tuple $(ID_i, Q_i, \perp)$ to the list L_{H1}^{list}.
  (b) Else, if $ID_i = ID_B$, then return $Q_i = H_1(ID_i) = cP$ and add the tuple $(ID_i, Q_i, \perp)$ to the list L_{H1}^{list}.
  (c) Else, return $Q_i = H_1(ID_i) = d_iP$, and add the tuple (ID_i, Q_i, d_i) to the list L_{H1}^{list}, where $d_i \in_R Z_q^*$.
- *Partial-Private-Key-Extract queries:* The algorithm $\mathscr{C}$ maintains an initial empty-list L_{ppke}^{list}, which contains the tuple of the form (ID_i, D_i, x_i, P_i). On receiving a *Partial-Private-Key-Extract* query on ID_i, $\mathscr{C}$ first recovers the tuple (ID_i, Q_i, d_i) from the list L_{H1}^{list} and then replies as follows:
  (a) If $(ID_i = ID_A, ID_B)$, then $\mathscr{C}$ aborts the protocol execution.
  (b) Else $(ID_i \neq ID_A, ID_B)$, $\mathscr{C}$ executes the following:
  – If the list L_{ppke}^{list} contains a tuple (ID_i, D_i, x_i, P_i) and if $D_i \neq \perp$, $\mathscr{C}$ then returns D_i as answer to $\mathscr{A}_{II}$. Else, $\mathscr{C}$ chooses a number $d_i \in_R Z_q^*$ and returns $D_i = d_i(aP)$ as the *partial private key* to ID_i, and then adds it to the list L_{ppke}^{list}.
  – If the list L_{ppke}^{list} does not contain a tuple (ID_i, D_i, x_i, P_i), $\mathscr{C}$ then chooses a number $d_i \in_R Z_q^*$ and returns $D_i = d_i(aP)$ to ID_i. Afterwards, $\mathscr{C}$ sets $(x_i, P_i) = (\perp, \perp)$ and includes (ID_i, D_i, x_i, P_i) in the list L_{ppke}^{list}.
- *Public-Key-Extract queries:* When $\mathscr{A}_{II}$ submits a *Public-Key-Extract* query for the user ID_i, $\mathscr{C}$ searches the list L_{ppke}^{list} for a tuple (ID_i, D_i, x_i, P_i), which is indexed by ID_i and then returns the output as follows:
  (a) If a tuple (ID_i, D_i, x_i, P_i) is found in the list L_{ppke}^{list} and
  – If $P_i \neq \perp$, then $\mathscr{C}$ returns P_i as an answer.
  – Else $(P_i = \perp)$, $\mathscr{C}$ chooses a number $x_i \in_R Z_q^*$, computes $P_i = x_iP$, returns P_i as an answer and then inserts (x_i, P_i) into the list L_{ppke}^{list}.
  (b) Else (there is no such (ID_i, D_i, x_i, P_i) in L_{ppke}^{list}), $\mathscr{C}$ chooses a $x_i \in_R Z_q^*$, sets $D_i = \perp$ and $P_i = x_iP$, outputs P_i as an answer and inserts the tuple (x_i, P_i) to the list L_{ppke}^{list}.
- *Set-Secret-Value queries:* Assume that $\mathscr{A}_{II}$ asks a *Set-Secret-Value* query for the user ID_i, $\mathscr{C}$ first looks into the list L_{ppke}^{list} and then replies in the following way:
  (a) If $(ID_i = ID_A, ID_B)$, then $\mathscr{C}$ aborts the simulation.
  (b) Else $(ID_i \neq ID_A, ID_B)$, $\mathscr{C}$ searches the list L_{ppke}^{list} and
  – If the list L_{ppke}^{list} contains a tuple of the form (ID_i, D_i, x_i, P_i), $\mathscr{C}$ checks whether $D_i = \perp$ holds. If so, $\mathscr{C}$ executes the *Partial-Private-Key-Extract query* to obtain D_i. If $P_i = \perp$, $\mathscr{C}$ executes the *Public-Key-Extract query* to obtain $(x_i, P_i = x_iP)$. Then, $\mathscr{C}$ returns x_i and adds the full private key (D_i, x_i) to the list L_{ppke}^{list}.
  – Else, (there is no tuple such (ID_i, D_i, x_i, P_i) in L_{ppke}^{list}), $\mathscr{C}$ executes the *Partial-Private-Key-Extract query* to obtain D_i and the *Public-Key-Extract query* to obtain (x_i, P_i). Then, $\mathscr{C}$ returns x_i and adds the full private key (D_i, x_i) to the list L_{ppke}^{list}.
- *Public-Key-Replacement queries:* Suppose the adversary $\mathscr{A}_{II}$ makes a *Public-Key-Replacement* query on (ID_i, P_i'), $\mathscr{C}$ then searches the list L_{ppke}^{list} and returns the output as follows:
  (a) If the list L_{ppke}^{list} contains a tuple of the form (ID_i, D_i, x_i, P_i), $\mathscr{C}$ sets $P_i = P_i' = x_i'P$ and then updates the tuple (ID_i, D_i, x_i, P_i) to (ID_i, D_i, x_i', P_i').
  (b) Else (there is no such tuple in the list L_{ppke}^{list}), $\mathscr{C}$ sets $D_i = \perp, P_i = P_i' = x_i'P$ and then adds the tuple $(ID_i, \perp, x_i', P_i')$ to the list L_{ppke}^{list}.
- *Hash queries to H_2:* The algorithm $\mathscr{C}$ maintains an initial-empty list L_{H2}^{list}, which contains the tuple of the form $(ID_i, ID_j, m_i, R_{1i}, h_i)$ and $\mathscr{A}_{II}$ is allowed to ask at most q_{H2} times H_2 queries. If $\mathscr{A}_{II}$ requests a H_2 hash value for

(m_i, R_{1i}) with the signer identity ID_i and designated verifier's identity ID_j, $\mathscr{C}$ returns the outputs in the following way:

(a) If $(ID_i, ID_j, m_i, R_{1i}, h_i)$ is found in the list L_{H2}^{list}, $\mathscr{C}$ then outputs h_i as an answer.

(b) Else (no such tuple is found there), $\mathscr{C}$ chooses a number $h_i \in_R Z_q^*$, returns it as the answer and inserts the tuple $(ID_i, ID_j, m_i, R_{1i}, h_i)$ into the list L_{H2}^{list}.

- *Hash queries to H_3:* We assume that $\mathscr{A}_{II}$ can ask H_3 queries at most q_{H3} times and $\mathscr{C}$ maintains an initial-empty list L_{H3}^{list} of tuple $(ID_i, ID_j, m_i, g_i, R_i, t_i)$. When $\mathscr{A}_{II}$ makes a H_3 queries on (m_i, g_i, R_i) with the signer identity ID_i and designated verifier's identity ID_j, $\mathscr{C}$ returns the answer as follows:
  (a) If a tuple $(ID_i, ID_j, m_i, g_i, R_i, t_i)$ exists in L_{H3}^{list}, $\mathscr{C}$ then returns t_i as the output.
  (b) Else, $\mathscr{C}$ chooses a $t_i \in_R Z_q^*$, returns it as the answer and then adds $(ID_i, ID_j, m_i, g_i, R_i, t_i)$ to the list L_{H3}^{list}.
- *CL-SDVD-Sign queries:* Assume that $\mathscr{A}_{II}$ can ask at most q_S times *CL-SDV-Sign* queries to $\mathscr{C}$. When $\mathscr{A}_{II}$ asks for a signature on a chosen message $m_i \in \{0,1\}^*$ with the signer's identity ID_i and designated verifier's identity ID_j, $\mathscr{C}$ then responds as follows:
  (a) If $ID_i \neq ID_A$ and $ID_j \neq ID_B$, $\mathscr{C}$ computes the private key $(D_i, x_i) = (d_i(aP), x_i)$ of ID_i, chooses a number $r_i \in_R Z_q^*$, the values h_i, t_i from the lists L_{H2}^{list} and L_{H3}^{list}, respectively, and then computes the signature as given below:
  – Compute $g_i = \hat{e}(D_i, Q_j)$ and set $H_2(m_i, x_i P_j) = h_i$
  – Compute $R_i = \hat{e}(P_0, Q_j)^{r_i h_i}$ and set $H_3(m_i, g_i, R_i) = t_i$.
  – Compute $V_i = r_i h_i P_0 + t_i D_i$ and $S_i = \hat{e}(V_i, Q_j)$.
  – Output the signature (R_i, S_i).
  (b) If $ID_j \neq ID_B$ and $ID_i \neq ID_A$, $\mathscr{C}$ computes ID_j's private key $(D_j, x_j) = (d_j(aP), x_j)$, chooses a number $r_j \in_R Z_q^*$, collects the values h_j, t_j from the lists L_{H2}^{list} and L_{H3}^{list}, respectively, and then computes the signature as follows:
  – Compute $g_j = \hat{e}(Q_i, D_j)$ and set $H_2(m_i, x_j P_i) = h_j$.
  – Compute $R_j = \hat{e}(P_0, Q_j)^{r_j h_j}$ and set $H_3(m_i, g_j, R_j) = t_j$.
  – Compute $S_j = R_j \cdot g_j^{t_j}$ and output the signature (R_j, S_j).
  (c) Otherwise, abort the protocol's execution.
- *CL-SDVS-Verify queries:* Assume that $\mathscr{A}_{II}$ can ask at most q_V times *CL-SDVD-Verify* queries to $\mathscr{C}$. When $\mathscr{A}_{II}$ makes a *CL-SDVS-Verify* query on the input signature (R, S) with the signer's identity ID_i and designated verifier's identity ID_j, $\mathscr{C}$ checks whether $(ID_i, ID_j) = (ID_A, ID_B)$ or $(ID_i, ID_j) = (ID_B, ID_A)$ holds.
  (a) If it holds, $\mathscr{C}$ aborts the protocol execution.
  (b) Else, $\mathscr{C}$ computes the private key $(D_j, x_j) = (d_j(aP), x_j)$ of ID_j and verifies the signature using the proposed *CL-SDVS-Verify* algorithm.
- *Forgery:* Finally, the adversary $\mathscr{A}_{II}$ generates a valid forged signature (R^*, S^*) on a chosen message m^* for the signer's identity ID_i and designated verifier's identity ID_j with non-negligible probability. If $(ID_i, ID_j) \neq (ID_A, ID_B)$ or $(ID_i, ID_j) \neq (ID_B, ID_A)$ holds, $\mathscr{C}$ outputs 'failure' and terminates the protocol execution; otherwise outputs a valid signature (R^*, S^*) with t^* on the chosen message m^*. Therefore, from the valid signature (R^*, S^*), $\mathscr{C}$ can compute

$$S^* = R^* \cdot e(Q_A, S_B)^{t^*}$$
$$\Rightarrow e(Q_A, S_B) = (S^*/R^*)^{1/t^*}$$
$$\Rightarrow e(bP, acP) = (S^*/R^*)^{1/t^*}$$
$$\Rightarrow e(P,P)^{abc} = (S^*/R^*)^{1/t^*}$$

Thus, the BDH problem $e(P,P)^{abc} = (S^*/R^*)^{1/t^*}$ is solved for the given random tuple $(P, aP, bP, cP) \in G_q$, where $a, b, c \in_R Z_q^*$ are unknown to $\mathscr{C}$. However, it is known that the BDH problem is unsolvable by any probabilistic polynomial time bounded algorithm, and hence, our CL-SDVS scheme is secure in the random oracle model under the adaptive chosen message and identity attacks against the adversary $\mathscr{A}_{II}$. □

5.3. Unforgeability against Type III adversary (resilience against malicious PKG attack)

Theorem 3. *The proposed CL-SDVS scheme is existentially unforgeable against the adversary $\mathscr{A}_{III}$ under the adaptive chosen message and identity attacks in the random oracle model provided the CDH problem is intractable in the elliptic curve group G_q.*

Proof. Suppose that an adversary $\mathscr{A}_{III}$ can forge the proposed CL-SDVS scheme with probability ε and within a time bound t, then there exists a probabilistic polynomial time bounded algorithm $\mathscr{C}$, which can solve an instance of the CDH problem with non-negligible probability. That is, for given a random instance $(P, aP, bP) \in G_q$, where $a, b \in_R Z_q^*$, $\mathscr{C}$ outputs abP. To solve the CDH problem, $\mathscr{C}$ selects a number $\lambda \in_R Z_q^*$ and sets the system's parameter $\Omega = \{G_q, G_m, \hat{e}, q, P, P_0 = \lambda P, H_1, H_2, H_3\}$, returns (Ω, λ) to $\mathscr{A}_{III}$ and then answers $\mathscr{A}_{III}$'s queries in the following way. In order to avoid collisions and consistently respond to $\mathscr{A}_{III}$'s queries, $\mathscr{C}$ keeps an initial-empty *Partial-Private-Key-Extract* list L_{ppke}^{list}, which consists of a tuple of the form (ID_i, x_i, P_i).

- *Hash queries to H_1:* Assume that $\mathscr{A}_{III}$ can ask at most q_{H1} times H_1 queries to $\mathscr{C}$. The algorithm $\mathscr{C}$ maintains an initial-empty list L_{H1}^{list} that contains the tuple of the form (ID_i, Q_i, d_i). When $\mathscr{A}_{III}$ makes a H_1 query on ID_i, $\mathscr{C}$ behaves as follows:
  (a) Search the list L_{H1}^{list} and respond with the previous value d_i if a tuple (ID_i, Q_i, d_i) is found.
  (b) Else, return $Q_i = H_1(ID_i) = d_i P$, and include the tuple (ID_i, Q_i, d_i) into the list L_{H1}^{list}, where $d_i \in_R Z_q^*$.
- *Public-Key-Extract queries:* When $\mathscr{A}_{III}$ queries on input ID_i, $\mathscr{C}$ searches the list L_{ppke}^{list} and answers as follows:
  (a) If the list L_{ppke}^{list} contains a tuple of the form (ID_i, x_i, P_i) and
  – If $ID_i = ID_A$, $\mathscr{C}$ returns $P_i = aP$ as an answer and inserts $(ID_i, \perp, P_i)$ into L_{ppke}^{list}.
  – If $ID_i = ID_B$, $\mathscr{C}$ returns $P_i = bP$ as an answer and inserts $(ID_i, \perp, P_i)$ into L_{ppke}^{list}.
  – If $ID_i \neq ID_A, ID_B$, $\mathscr{C}$ returns the previous P_i as the answer.
  (b) Else (there is no such tuple), $\mathscr{C}$ picks a number $x_i \in_R Z_q^*$, returns $P_i = x_i P$ as the answer and then adds the tuple (ID_i, x_i, P_i) to the list L_{ppke}^{list}.
- *Set-Secret-Value queries:* When $\mathscr{A}_{III}$ asks for a *Set-Secret-Value* query with ID_i, $\mathscr{C}$ performs as follows:
  (a) If $(ID_i = ID_A, ID_B)$, then $\mathscr{C}$ terminates the process.
  (b) Else $(ID_i \neq ID_A, ID_B)$, $\mathscr{C}$ searches the list L_{ppke}^{list} and answers as follows:
  – If the list L_{ppke}^{list} contains a tuple (ID_i, x_i, P_i) and if $P_i = \perp$, $\mathscr{C}$ executes the *Public-Key-Extract* query in order to obtain (ID_i, x_i, P_i), returns x_i as the answer and inserts the tuple (ID_i, x_i, P_i) to the list L_{ppke}^{list}.

– Else (there is no tuple such tuple in L_{ppke}^{list}), $\mathscr{C}$ then executes the *Public-Key-Extract* query to obtain (ID_i, x_i, P_i). Subsequently,$\mathscr{C}$ saves the value and adds the secret key x_i to the list L_{ppke}^{list}.

- *Hash queries to* H_2: The adversary $\mathscr{A}_{III}$ is allowed to ask utmost q_{H2} times H_2 queries to $\mathscr{C}$. The algorithm $\mathscr{C}$ maintains an initial-empty list L_{H2}^{list} that contains tuples of the form $(ID_i, ID_j, m_i, R_{1i}, h_i)$. When $\mathscr{A}_{III}$ makes H_2 queries on (m_i, R_{1i}) with the signer identity ID_i and designated verifier's identity ID_j, $\mathscr{C}$ returns the previous value from the list L_{H2}^{list} if a tuple of the form $(ID_i, ID_j, m_i, R_{1i}, h_i)$ is found; otherwise, it chooses a $h_i \in_R Z_q^*$, returns it as the answer and then adds $(ID_i, ID_j, m_i, R_{1i}, h_i)$ to the list L_{H2}^{list}.
- *Hash queries to* H_3*:* The adversary $\mathscr{A}_{III}$ is allowed to ask utmost q_{H3} times H_3 queries to $\mathscr{C}$. The algorithm $\mathscr{C}$ maintains an initial-empty list L_{H3}^{list} that contains tuples of the form $(ID_i, ID_j, m_i, g_i, R_i, t_i)$. When $\mathscr{A}_{III}$ makes H_3 queries on (m_i, g_i, R_i) with the signer identity ID_i and designated verifier's identity ID_j, C returns the previous value t_i from L_{H3}^{list} if a tuple of the form $(ID_i, ID_j, m_i, g_i, R_i, t_i)$ existed; otherwise, $\mathscr{C}$ chooses a number $t_i \in_R Z_q^*$, returns it as the answer and then adds the tuple $(ID_i, ID_j, m_i, g_i, R_i, t_i)$ into the list L_{H3}^{list}.
- *CL-SDVD-Sign queries:* Assume that $\mathscr{A}_{III}$ can ask at most q_S times *CL-SDV-Sign* queries to $\mathscr{C}$. When $\mathscr{A}_{III}$ asks for a signature on message $m_i \in \{0,1\}^*$ chosen by him with the signer's identity ID_i and designated verifier's identity ID_j, $\mathscr{C}$ responds as follows:
  (a) If $ID_i \neq ID_A$ and $ID_j \neq ID_B$, $\mathscr{C}$ computes the private key $(D_i, x_i) = (\lambda(d_i P),\ x_i)$ of ID_i, chooses a number $r_i \in_R Z_q^*$, recovers h_i and t_i from the lists L_{H2}^{list} and L_{H3}^{list} respectively, and then computes the signature as follows:
    – Compute $g_i = \hat{e}(D_i, Q_j)$ and set $H_2(m_i, x_i P_j) = h_i$.
    – Compute $R_i = \hat{e}(P_0, Q_j)^{r_i h_i}$ and set $H_3(m_i, g_i, R_i) = t_i$.
    – Compute $V_i = r_i h_i P_0 + t_i D_i$ and $S_i = \hat{e}(V_i, Q_j)$.
    – Output the signature (R_i, S_i).
  (b) If $ID_j \neq ID_B$ and $ID_i \neq ID_A$, $\mathscr{C}$ computes the private key $(D_j, x_j) = (\lambda(d_j P),\ x_j)$ of ID_j, chooses a number $r_i \in_R Z_q^*$, recovers h_i and t_i from the lists L_{H2}^{list} and L_{H3}^{list} respectively, and then computes the signature as follows:
    – Compute $g_j = \hat{e}(Q_i, D_j)$ and set $H_2(m_i, x_j P_i) = h_j$.
    – Compute $R_j = \hat{e}(P_0, Q_j)^{r_j h_j}$ and set $H_3(m_i, g_j, R_j) = t_j$.
    – Compute $S_j = R_j \cdot g_j^{t_j}$ and output the signature (R_j, S_j).
  (c) Otherwise, terminate the protocol execution.
- *CL-SDVS-Verify queries:* Assume that $\mathscr{A}_{III}$ can ask at most q_V times *CL-SDVD-Verify* queries to $\mathscr{C}$. When $\mathscr{A}_{III}$ makes a *CL-SDVS-Verify* query on the input signature (R, S) with the signer's identity ID_i and verifier's identity ID_j, then $\mathscr{C}$ checks whether $(ID_i, ID_j) = (ID_A, ID_B)$ or $(ID_i, ID_j) = (ID_B, ID_A)$ holds.
  (a) If it holds, $\mathscr{C}$ terminates the protocol execution.
  (b) Otherwise, $\mathscr{C}$ computes the private key $(D_j, x_j) = (r_j(\lambda P), x_j)$ of ID_j to verify the signature by using the *CL-SDVS-Verify* algorithm.
- *Forgery:* Eventually, the adversary $\mathscr{A}_{III}$ outputs a valid forged signature (R^*, S^*) on the chosen message m^* for the signer ID_i and designated verifier ID_j with non-negligible probability. If $(ID_i, ID_j) \neq (ID_A, ID_B)$ or $(ID_i, ID_j) \neq (ID_B, ID_A)$ holds, $\mathscr{C}$ outputs 'failure' and stops the protocol execution. Otherwise, $\mathscr{C}$ searches the list L_{H2}^{list} for the tuple of the form $\left(m^*, x_i^* P_j^*, h_i^*\right)$. If no such tuple is found, $\mathscr{C}$ outputs 'failure'; otherwise, $\mathscr{C}$ outputs $abP = x_i^* P_j^*$ as the solution of the CDH problem.

Hence, $\mathscr{C}$ breaks the CDH problem for given $(P, aP, bP) \in G_q$, where $a, b \in_R Z_q^*$ are unknown. Thus, the proposed CL-SDVS scheme is secure in the random oracle model under the adaptive chosen message and identity attacks against the adversary $\mathscr{A}_{III}$. □

Theorem 4. *The proposed CL-SDVS scheme satisfies the strongness, source hiding and non-delegatability properties of a strong designate verifier signature scheme.*

Proof. The proposed CL-SDVS scheme is a *strong designated verifier signature* scheme. For the verification of the signature (R, S), the private key $sk_B = (D_B, x_B)$ of *Bob* (designated verifier) is required and thus, an outsider can neither verify the signature (R, S) nor prove that *Alice* (original signer) or *Bob* has generated the signature. Therefore, the *strongness* property is satisfied in our proposed CL-DSVS scheme. It can be noted that *Bob* can generate a simulated signature $(\widehat{R}, \widehat{S})$, which is indistinguishable from the normal signature (R, S) created by *Alice*. Therefore, an outsider cannot identify that (R, S) is signed by *Alice* or *Bob* even if all the private keys are known. Thus, the *source hiding* property is achieved in our scheme. In addition, *Bob* cannot prove to an outsider that the signature (R, S) is computed either by him or *Alice*. Suppose that (R', S') is chosen at random from the set of all valid signatures designed for *Bob*, then the probability $\Pr[(R, S) = (R', S')] = 1/(q-1)$, since the signature (R, S) is computed based on the integer r, which is chosen randomly from the set Z_q^*. Similarly, we can write $\Pr[(\widehat{R}, \widehat{S}) = (R', S')] = 1/(q-1)$. Therefore, the simulated signature and the original signature have the identical distribution, and thus, they are indistinguishable from each other. Hence, the proposed CL-SDVS scheme satisfies the *non-delegatability* property through the *CL-SDVS-Simulation* algorithm. □

For a further security discussion, three trust levels, defined by Girault (1992) to reduce the degree of trust that users need to have in the PKG, can easily be achieved in our proposed CL-SDVS scheme. The Theorems 2 and 3 discussed earlier confirm that the proposed scheme achieves Girault's first two trust levels 1 and 2. The discussions about the Girult's trust level 3 made by Al-Riyami and Paterson (2003) and Gorantla and Saxena (2005) indicate that the adversary $\mathscr{A}_{III}$(i.e., untrusted PKG) can still impersonate a user ID_i without knowing the corresponding secret key x_i, i.e., the PKG can forge a valid signature by replacing the original public key $P_i = x_i P$ with a fake public key $P_i'' = x_i'' P$ of its own choice. Because the PKG can calculate the partial private key $D_i = sQ_i$ and generate another valid key pair (D_i, P_i'') for the user ID_i, so the PKG can impersonate the valid user ID_i without being detected, and it happened because the user with ID_i can also generate another valid key pair (D_i, P_i') corresponding to the partial private key D_i as the partial private key D_i and the public key P_i are generated independently. Note that this type of attack may occur only by the adversary who has access to the PKG's master private keys, i.e., Type III adversary $\mathscr{A}_{III}$ in our scheme as defined earlier.

The proposed scheme can also achieve trust level 3 just by using the alternate key generation technique as described in Section 5.1 of Al-Riyami and Paterson (2003). In this technique, a user ID_i first generates his secret value and the corresponding public key as x_i and $P_i = x_iP$ respectively, and makes a partial-private-key request with (ID_i, P_i) to the PKG. Then PKG calculates the partial private key as $D_i = s\text{-}Q_i$, where $Q_i = H_1(ID_i, P_i)$ is the corresponding public key. It can be noted that the user ID_i has only one partial private key D_i corresponding to the public key $P_i = x_iP$ and he cannot create another public key $P_i' = x_i'P$ while keeping the same partial private key D_i. The user ID_i can do so if he knows the PKG's master private key $msk = s$, which is impossible as proven in theorem 1. Note that the PKG still can create another pair (D_i, P_i'') for the user ID_i; however, the impersonation of any user by the PKG can easily be detected because this is its only ability. Thus, our CL-SDVS scheme also meets the trust level 3.

6. Comparison of the proposed scheme with others

This section compares the proposed CL-SDVS scheme with other competitive ones in terms of security, computation and communication aspects. Recently, Cao et al. (2010) has estimated the running time of different cryptographic operations such as elliptic curve bilinear pairing, elliptic curve scalar point multiplication and elliptic curve bilinear pairing-based exponentiation operations using MIRACAL software (Shamus Software Ltd.) implemented on a Pentium IV 3 GHZ processor with 512 MB RAM and the Windows XP operating system. Table 1 shows the experimental running time of different cryptographic operations obtained by Cao et al. (2010). Since the proposed scheme mainly uses four operations as mentioned in Table 1, the running cost of the proposed technique is estimated as follows: where the time required for system parameters is not considered as a trusted third party, PKG generates them once in the background mode. In the proposed scheme, the map-to-point hash function (Boneh and Franklin, 2001) is avoided in the signature generation and verification phases, which helps to improve the overall performance of our scheme. It can be noted that both the signer and verifier can pre-compute $\hat{e}(D_A, Q_B)$ and $\hat{e}(P_0, Q_B)$, and therefore, these two computations are ignored in the comparison part. In our scheme, the signature generation and verification algorithms execute one bilinear pairing, four scalar point multiplications and two pairing-based exponentiations and thus, the total computing time is $(1T_{BP} + 4T_{EM} + 2T_{PX})$ and the running time according to Table 1 is $(20.01 + 6.38 \times 4 + 11.20 \times 2) \approx 67.93$ ms. Similarly, the running time of other methods are calculated and shown in Table 2, which shows that the proposed scheme is the most computationally efficient. It is to be noted that the time needed to execute the addition of any two elliptic curve points is neglected, since its computation time is very small (Chung et al., 2007).

In order to assess the communication efficiency, we follow the same reasoning made by Cao et al. (2010) that for achieving the 1024-bit RSA level of security, the pairing-based schemes employ the Tate pairing (Boneh and Franklin, 2001) defined over the super singular elliptic curve $E/F_p: y^2 = x^3 + x$ (Miller, 1985; Koblitz, 1987) with embedding degree 2, where q is a 160-*bit* Solinas prime $q = 2^{159} + 2^{17} + 1$ and p is at least a 512-*bit* prime number satisfying the relation $p + 1 = 12qr$ (Solinas, 2011) for the said field; whereas to meet the same level of security, the pairing-free ECC-based schemes use Koblitz elliptic curve $y^2 = x^3 + ax^2 + b$ (Koblitz, 1987) defined over the field F_2^{163} of prime numbers with $a = 1$ and b is a 163-*bit* random prime number. Thus, a 163-bit random num-

Table 1 Notation, description and running time (ms) of different cryptographic operations.

Notations	Descriptions
T_{EM}	Time complexity for executing the elliptic curve scalar point multiplication, $1T_{EM} \approx 6.38$ ms
T_{BP}	Time complexity for executing the bilinear pairing operation, $1T_{BP} \approx 20.01$ ms
T_{PX}	Time complexity for executing pairing-based exponentiation, $1T_{PX} \approx 11.20$ ms
T_{EA}	Time complexity for executing the addition of two elliptic curve points, which is negligable

Table 2 The performance comparison of the proposed scheme with other related schemes.

Schemes/parameters	Computing time	Running time (ms)	Signature length (byte)	Complexity assumptions	Is SDVS scheme
Chen et al. (2007)	$6T_{BP} + 13T_{EM}$	171.10	128	BDH	No
Yang et al.'s, 2007	$4T_{BP} + 8\ T_{EM}$	131.08	128	BDH	No
Hongzhen and Qiaoyan (2009)	$3T_{BP} + 8\ T_{EM} + 1T_{PX}$	122.27	192	CDH	No
Gorantla and Saxena (2005)	$4T_{BP} + 6\ T_{EM}$	118.32	128	BDH	No
Xiao et al. (2010)	$2T_{BP} + 6\ T_{EM} + 1T_{PX}$	102.26	128	BDH & CDH	Yes
Xu et al. (2008)	$2T_{BP} + 8\ T_{EM}$	91.06	128	CDH	No
Yang et al. (2009)	$2T_{BP} + 8\ T_{EM}$	91.06	128	BDH & CDH	Yes
Zhang and Zhang (2008)	$2T_{BP} + 7T_{EM}$	84.68	128	CDH	No
Choi et al. (2007)	$2T_{BP} + 7T_{EM}$	84.68	128	CDH	No
Li and Liu (2011)	$2T_{BP} + 6T_{EM}$	78.30	128	BDH	No
Proposed	$1T_{BP} + 4T_{EM} + 2T_{PX}$	67.93	128	BDH & CDH	Yes

BDH: Bilinear Diffie-Hellman Problem; *CDH:* Computational Diffie-Hellman Problem; *SDVS Scheme:* Strong designated verifier signature scheme.

ber in a pairing-free scheme is equivalent in security with a 512-bit random number in a pairing-based scheme. In our scheme, since the final signature consists of two elliptic curve points from the pairing-based group of super singular elliptic curve E/F_p: $y^2 = x^3 + x$, the length of the signature is $(512 + 512)/8 = 128$ bytes as shown in Table 2.

Finally, the security comparison of the different schemes is done and Table 2 indicates that the two schemes corresponding to Yang et al. (2009) and Xiao et al. (2010), including our proposed one are CL-SDVS scheme, whereas the rest are only CLS schemes. We also list different signature schemes and their demerits in Table 3, which demonstrates that Xiao et al. scheme (2010) is insecure against all types of adversaries present in the CL-PKC system, but Yang et al. (2009) and our CL-SDVS scheme offer provable security in the random oracle model (Bellare and Rogaway, 1993). Thus, compared with existing schemes, the proposed scheme is computationally efficient, provably secure and more suitable for practical applications.

7. Conclusion

In this paper, an efficient and secure CL-SDVS scheme using elliptic curve cryptography and bilinear pairings has been proposed. The proposed scheme achieves different trust levels defined by Girault in order to reduce the degree of trust that users need to have in the PKG. The scheme also satisfies the necessary security properties of strong designated verifier signatures in the CL-PKC system. It has been shown that the proposed scheme is unforgeable under the adaptive chosen message and identity attacks against various adversaries in the random oracle model based on the intractability of BDH and CDH assumptions, which assures a wider applicability in various applications.

Table 3 The security comparison of the proposed scheme with other related schemes.

Schemes/attacks	Resilience against $\mathcal{A}_{II}$ adversary (public key replacement attack)	Resilience against $\mathcal{A}_{III}$ adversary (malicious PKG attack)
Yum and Lee (2004)	No	Yes
Gorantla and Saxena (2005)	No	Yes
Huang et al. (2006)	Yes	No
Yap et al. (2006)	No	Yes
Chen et al. (2007)	No	Yes
Du and Wen (2007)	No	Yes
Xu et al. (2008)	Yes	No
Xiao et al. (2010)	No	No
Choi et al. (2011)	No	Yes
Proposed	Yes	Yes

Adversary $\mathcal{A}_{II}$: the adversary $\mathcal{A}_{II}$ has no knowledge about users' partial-private-keys, but can replace users' public keys with a value of his own choice.*Adversary* $\mathcal{A}_{III}$: the adversary $\mathcal{A}_{III}$ has the knowledge about users' partial-private-keys, but cannot replace users' public keys.*Yes:* resists the attack.*No:* does not resist the attack.

Acknowledgements

The authors are grateful to the Editor-in-Chief, Prof. Mansour M. Alsulaiman and the anonymous reviewers for their valuable comments and suggestions, which helped improve this paper. This research work is supported by the Department of Science and Technology (DST), Govt. of India under the INSPIRE fellowship Ph.D program (Reg. No. IF10247) and the Department of Information Technology (DIT), Ministry of Communication and Information Technology, Govt. of India under the Information Security Education and Awareness (ISEA) program (Project No. MIT(2)/2006-08/189/CSE). The authors would also like to express their gratitude and heartiest thanks to the Department of Computer Science and Engineering, Indian School of Mines, Dhanbad-826004, India for providing their research support, without which this work could not be carried out.

References

Al-Riyami, S., Paterson, K., 2003. Certificateless public key cryptography. In: Proceedings of the Asiacrypt'03. LNCS, vol. 2894. Springer-Verlag, pp. 452–473.

Bellare, M., Rogaway, P., 1993. Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the ACM CCCS'93, pp. 62–73.

Boneh, D., Franklin, M.K., 2001. Identity-based encryption from the Weil pairing. In: Proceedings of the Crypto'01. LNCS, vol. 2139. Springer-Verlag, pp. 213–229.

Cao, X., Paterson, K.G., Kou, W., 2006. An attack on a certificateless signature scheme. Cryptology ePrint Archive, Report 2006/367.

Cao, X., Kou, W., Du, X., 2010. A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. Information Sciences 180 (15), 2895–2903.

Chen, Y., Wen-ping, M., Xin-mei, W., 2007. Secure mediated certificateless signature scheme. The Journal of China Universities of Posts and Telecommunications 14 (2), 75–78.

Chen, H., Song, R., Zhang, F., Song, F., 2008. An efficient certificateless short designated verifier signature scheme. In: Proceedings of the International Conference on WiCOM'08, Dalian, pp. 1-6.

Choi, K.Y., Park, J., Hwang, J., Lee, D., 2007. Efficient certificateless signature schemes. In: Proceedings of the ACNS'07. LNCS, vol. 4521. Springer-Verlag, pp. 443–458.

Choi, K.Y., Park, J.H., Lee, D.H., 2011. A new provably secure certificateless short signature scheme. Computers and Mathematics with Applications 61 (7), 1760–1768.

Chung, YF., Huang, KH., Lai, F., Chen, TS., 2007. ID-based digital signature scheme on the elliptic curve cryptosystem. Computer Standards & Interfaces 29, 601–604.

Diffie, W., Hellman, M., 1976. New directions in cryptography. IEEE Transactions on Information Theory 22 (6), 644–654.

Du, H., Wen, Q., 2007. Efficient and provably-secure certificateless short signature scheme from bilinear pairings. Cryptology ePrint Archive, Report 2007/250, 2007.

Fan, C-I., Hsu, R-H., Ho, P-H., 2009. Cryptanalysis on Du-Wen certificateless short signature scheme. In: Proceedings of the JWIS'09, Institute of Electrical and Electronics Engineers, Kaohsiung, pp. 1–7.

Girault, M., 1992. Self-certified public keys. In: Proceedings of the Advances in Cryptology (Eurocrypt'91). LNCS, vol. 547. Springer-Verlag, pp. 490–497.

Gorantla, M.C., Saxena, A., 2005. An efficient certificateless signature scheme. In: Proceedings of the ICCIS'05. LNAI, vol. 3802. Springer-Verlag, pp. 110–116.

Guozheng, H., Fan, H., 2009. Attacks against two provably secure certificateless signature schemes. In: Proceedings of the WASE International Conference on Information Engineering, pp. 246–249.

Hongzhen, D., Qiaoyan, W., 2009. Efficient certificateless designated verifier signatures and proxy signatures. Chinese Journal of Electronics 18 (1), 95–100.

Hu, B., Wong, D., Zhang, Z., Deng, X., 2006. Key replacement attack against a generic construction of certificateless signature. In: Proceedings of the ACISP'06. LNCS, vol. 4058. Springer-Verlag, pp. 235–346.

Huang, X., Susilo, W., Mu, Y., Zhang, F., 2005. On the security of a certificateless signature scheme. In: Proceedings of the CANS'05. LNCS, vol. 3810. Springer-Verlag, pp. 13–25.

Huang, X., Susilo, W., Mu, Y., Zhang, F., 2006. Certificateless designated verifier signature schemes. In: Proceedings of the 20th International Conference on AINA'06, vol. 2, pp. 15–19.

Jakobsson, M., Sako, K., Impagliazzo, R., 1996. Designated verifier proofs and their applications. In: Proceedings of the Eurocrypt'96. LNCS, vol. 1070. Springer-Verlag, pp. 143–154.

Koblitz, N., 1987. Elliptic curve cryptosystem. Journal of Mathematics of Computation 48 (177), 203–209.

Li, F., Liu, P., 2011. An efficient certificateless signature scheme from bilinear parings. In: Proceedings of the International Conference on NCIS'11, Guilin, China, pp. 35–37.

Miller, V.S., 1985. Use of elliptic curves in cryptography. In: Proceeding of the Crypto'85. Springer-Verlag, New York, pp. 417–426.

Shamir, A., 1984. Identity based cryptosystems and signature schemes. In: Proceedings of the Crypto'84. LNCS, vol.196. Springer-Verlag, pp. 47–53.

Shamus Software Ltd. Miracl library. <http://www.shamus.ie/index.php?page=home>.

Solinas, J.A., 2011. Generalized Mersenne Prime. Encyclopedia of Cryptography and Security. (2nd ed. pp. 509–510.

Tian, M., Huang, L., Yang, W., 2011. On the security of a certificateless short signature scheme. Cryptology ePrint Archive, <http://eprint.iacr.org/2011/419>.

Tso, R., Yi, X., Huang, X., 2011. Efficient and short certificateless signatures secure against realistic adversaries. Journal of Supercomputer 55, 173–191.

Xiao, Z., Yang, B., Li, S., 2010. Certificateless strong designated verifier signature scheme. In: Proceedings of the 2nd International Conference on EBISS'10, Wuhan, pp. 1–5.

Xu, Z., Liu, X., Zhang, G., He, W., Dai, G., Shu, W., 2008. A certificateless signature scheme for mobile wireless cyber-physical systems. In: Proceedings of the ICDCS'08, pp. 489–494.

Yang, M., Shem, X-Q., Wang, Y-M., 2007. Certificateless universal designated verifier signature scheme. The Journal of China Universities of Posts and Telecommunications 14 (3), 85–94.

Yang, B., Hu, Z., Xiao, Z., 2009. Efficient certificateless strong designated verifier signature scheme. In: Proceedings of the International Conference CIS'09, pp. 432–436.

Yap, W., Heng, S., Goi, B., 2006. An efficient certificateless signature scheme. In: Proceedings of the EUC Workshops 2006, LNCS, vol. 4097. Springer-Verlag, pp. 322–331.

Yum, D., Lee, P., 2004. Generic construction of certificateless signature. In: Proceedings of the ACISP'04. LNCS, vol. 3108. Springer-Verlag, pp. 200–211.

Zhang, Z., Feng, D., 2006. Key replacement attack on a certificateless signature scheme. Cryptology ePrint Archive, Report 2006/453.

Zhang, Z., Wong, D., Xu, J., Feng, D., 2006. Certificateless public-key signature: security model and efficient construction. In: Proceedings of the ACNS 2006, LNCS, vol. 3989. Springer-Verlag, pp. 293–308.

Zhang, L., Zhang, F., 2008. A new provably secure certificateless signature scheme. In: Proceedings of the IEEE International Conference on Communications (ICC'08), Beijing, China, pp. 1685–1689.

Zhang, J., Xie, J., 2011. Breaking a certificateless strong designated verifier signature scheme. In: Proceedings of the International Conference on CECNet'11, pp. 130–133.