Journal of King Saud University – Computer and Information Sciences (2014) **26**, 55–67

King Saud University

Journal of King Saud University – Computer and Information Sciences

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

ORIGINAL ARTICLE

A provably secure identity-based strong designated verifier proxy signature scheme from bilinear pairings

SK Hafizul Islam *, G.P. Biswas

Department of Computer Science and Engineering, Indian School of Mines, Dhanbad 826004, Jharkhand, India

Received 21 October 2012; revised 17 January 2013; accepted 17 March 2013
Available online 26 March 2013

KEYWORDS

Elliptic curve cryptography;
Identity-based cryptosystem;
Bilinear pairing;
Proxy signature;
Strong designated verifier;
AVISPA tool

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. Although many identity-based proxy signature schemes have been proposed in the literature, only a few proposals for identity-based strong designated verifier proxy signature (ID-SDVPS) schemes are available. However, it has been found that most of the ID-SDVPS schemes that have been proposed to date are not efficient in terms of computation and security, and a computationally efficient and secured ID-SDVPS scheme using elliptic curve bilinear pairing has been proposed in this paper. The security of the scheme is mainly based on the hardness assumption of CDH and GBDH problems in the random oracle model, which is existentially unforgeable against different types of adversaries. Furthermore, the security of our scheme is simulated in the AVISPA (Automated Validation of Internet Security Protocols and Applications) software, a widely used automated internet protocol validation tool, and the simulation results confirm strong security against both active and passive attacks. In addition, because of a high processing capability and supporting additional security features, the scheme is suitable for the environments in which less computational cost with strong security is required.

1. Introduction

In a PKI (public key infrastructure)-based cryptosystem, the public key certificate that is generated and signed by a certificate authority (CA) is required for authentication of the public keys of the entities, and, as a result, it creates a heavy management burden for maintaining and using the public key certificate by developing a global infrastructure. As a remedy, Shamir (1984) proposed the concept of an identity-based cryptosystem (IBC) that supports the users' authentication through the use of a public identity. In other words, a user's public key in IBC is computed from an email identity, a social security number, a passport number or other identifiers and a private key generator (PKG); a trusted third party generates the user's private key by using the user's identity and his/her master private key. The private key generated by PKG is communicated to the user through a secure channel, for which its legitimacy can be verified by the user publicly. However, as such, no practical implementation for IBC was proposed by Shamir, and in 2001, Boneh and Franklin (2001) first proposed a bilinear pairing-based technique (Weil or Tate) that uses a super-singular elliptic curve based on the Bilinear Diffie

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

1319-1578 © 2013 Production and hosting by Elsevier B.V. on behalf of King Saud University.
http://dx.doi.org/10.1016/j.jksuci.2013.03.004

Hellman (BDH) assumption in the random oracle model (Bellare and Rogaway, 1993), which is called the identity-based encryption (IBE) technique. Subsequently, a number of IBE cryptosystems have been developed. In this paper, we have proposed an identity-based strong designated verifier proxy signature (ID-SDVPS) scheme that uses bilinear pairing for mapping from an elliptic curve additive cyclic group (Miller, 1985; Koblitz, 1987) to any multiplicative cyclic group of the same prime order. Next, the description of some signature schemes will be given.

In 1996, Jakobsson et al. (1996) first proposed a designated verifier signature (DVS) scheme for which the original signer *Alice* generates a signature, and only a designated verifier *Bob* can verify the signature. However, it can be seen that the signer's privacy protection is violated in DVS schemes because *Bob* can easily convince a third party that the message was signed by *Alice*. To remove this problem, Jakobsson et al. (1996) proposed another signature scheme, called the strong designated verifier signature (SDVS). In an SDVS scheme (Huang et al., 2008; Kang et al., 2009; Islam and Biswas, 2013), *Bob* cannot prove to an outsider that *Alice* is the original signer. This problem occurs because an identical signature can be generated by *Bob*, and it cannot be distinguished from the signatures of *Alice;* in addition, *Bob*'s private key is strictly required in the verification phase. Therefore, the SDVS scheme satisfies the *strongness* and *repudiable* properties. Since then, many SDVS schemes (Saeednia et al., 2004; Huang et al., 2008; Lee et al., 2010; Tang et al., 2011; Yoon, 2011) have been proposed by researchers for different applications of Network/Information Security.

In 1996, Mambo et al. (1996) proposed a 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. However, this proxy signature scheme allows public verification, which might not be suitable for applications in which the verification of the proxy signature for personal sensitive and/or important commercial documents must be performed by designated persons. Thus, a strong designated verifier proxy signature (SDVPS) scheme (Dai et al., 2003; Wang, 2004; Cao et al., 2005; Lin et al., 2011) is proposed for these environments. In this scheme, the proxy signer computes a proxy signature for the designated verifier, who only validates the signature but is unable to convince an outsider about the original signer and the proxy signer. The reason is that the designated verifier can also generate a simulated proxy signature that is intended for him, which is indistinguishable from the original proxy signature.

Proxy signatures have been suggested for many applications, including distributed systems (Neuman, 1993), grid computing (Foster et al., 1998), mobile agent systems (Kim et al., 2001), mobile communications (Park and Lee, 2001), and e-commerce (Dai et al., 2003; Wang, 2004). Based on the application areas, the proxy signature can be categorized into four types (Mambo et al., 1996; Wang, 2008; Yang, 2010; Islam and Biswas, 2012a), namely *full delegation*, *partial delegation*, *delegation by warrant* and *partial delegation by warrant*. Among these, the *partial delegation by warrant* satisfies all the security requirements of the proxy signatures. Additionally, the proxy signature is based on the proxy private key; the proxy signatures can be classified into two signature types, which are *proxy-unprotected* and *proxy-protected*. There is a repudiation dispute problem in the *proxy-unprotected* scheme because the proxy signature is created either by the original signer or the proxy signer. On the other hand, the repudiation dispute problem is absent in the *proxy-protected* scheme because only the proxy signer generates the proxy signature.

1.1. Recent studies

A number of new identity-based strong designated verifier proxy signature (ID-SDVPS) schemes have been proposed recently (Cao et al., 2005; Lal and Verma, 2006; Kang et al., 2009; Lee et al., 2010), and a short discussion of each is provided here. In 2003, Dai et al. (2003) proposed a designated verifier proxy signature (DVPS) scheme that is suitable for e-commerce environments. In 2005, Cao et al. (2005) proposed an identity-based DVPS (ID-DVPS) scheme that is based on Cha and Cheon's signature scheme (Cha and Cheon, 2003) and uses bilinear pairing, and Gu and Zhu (2005) proposed a new computational model for a provably secure identity-based proxy signature scheme. In 2006, Lal and Verma (2006) proposed an ID-DVPS scheme using bilinear pairings. However, Kang et al. (2009) proved that the scheme was insecure; then, they proposed an efficient scheme and claimed that the scheme was unforgeable. Later, in 2008, Gu and Zhu (2008) proposed an efficient version of Zhang and Kim's scheme (2003), which was based on the security model proposed in (Gu and Zhu, 2005). In 2010, Lee et al. (2010) demonstrated that the scheme proposed in (Kang et al., 2009) is universally forgeable, which means that anyone can forge a valid ID-DVPS on an arbitrary message without the knowledge of the secret key of either the signer or the designated verifier.

A new proxy signature scheme was also proposed by Wu et al. (2007), which improves the security aspects of an identity-based proxy signature scheme. Wang (2008) gave a new identity-based proxy signature scheme, which is secure against the *proxy key exposure attack* in the random oracle model (Bellare and Rogaway, 1993). In general, an ID-DVPS scheme needs a trusted PKG unconditionally; otherwise, a dishonest PKG can compute the private key of any user and can forge its proxy signature. In 2010, Yang et al., as a solution of the problem, proposed an ID-DVPS scheme without a trusted party. Later on, Reddy et al. (2010) also proposed an identity-based directed proxy signature scheme using bilinear pairings and the concept of Hess's identity-based signature scheme (Hess, 2002). In 2012, Islam and Biswas (2012a) proposed an efficient ID-based Short DVPS (ID-ShDVPS) scheme using elliptic curve bilinear pairing, which is a short signature scheme and is applicable to the environments with limited bandwidth, computing power and storage space. However, it always generates the same proxy signature on the same message, and its security is proven only heuristically.

1.2. Our contributions

Elliptic curve cryptography and bilinear pairing are two efficient tools that are used to design secure cryptographic protocols for various applications (Zhang and Kim, 2003; Dai et al., 2003; Cao et al., 2005; Farash et al., 2012; Das, 2012). In the literature, many ID-SDVPS schemes using elliptic curve

bilinear pairing have been found. However, the main shortcoming of these schemes is the involvement of high computational cost because many bilinear pairing and pairing-based exponentiation operations are executed for their implementation. Moreover, the security of most of the previous schemes is argued only heuristically; thus, these schemes are prone to security attacks against active adversaries. Therefore, it is necessary to design a provably secure and computationally efficient ID-SDVPS scheme, and in this paper, we proposed such a scheme by using elliptic curve bilinear pairing, where it combines the idea of ID-SDVS schemes proposed in (Huang et al., 2008; Sun et al., 2010; Yoon, 2011). The schemes (Huang et al., 2008; Sun et al., 2010) are known to be provably secured, i.e., the schemes are existentially unforgeable under the adaptive chosen message and identity attacks in the random oracle model (Bellare and Rogaway, 1993), based on the intractability of GBDH and BDH assumptions, respectively. However, the main shortcoming of the scheme (Huang et al., 2008) is that it always produces the same signature on the same message. On the other hand, Yoon's ID-SDVS scheme (2011) is efficient with respect to the computations and the communications; however, it is not proven to be provably secured in the random oracle model. Moreover, our scheme has been analyzed using the random oracle model, and it has been found that the scheme is to be existentially unforgeable against adaptively chosen message and identity attacks based on the hardness assumption of CDH and GBDH assumptions. In addition, we have implemented our proposed ID-SDVPS scheme in the AVISPA software (AVISPA, 2005, 2013), which is a strong simulation tool that is used for the automated security analysis of cryptographic protocols and for formal security verification, and the results obtained through simulation prove it to be strongly secure against active and passive attacks. Additionally, it is computationally efficient because a smaller number of bilinear pairings is involved, and it satisfies all the necessary security requirements.

1.3. Organization of the paper

The remainder of this paper is organized as follows. Section 2 describes some preliminaries that are required in the paper. Section 3 illustrates our ID-SDVPS scheme, and the provable security analysis and formal verification of the scheme are addressed in Section 4. The comparison of our scheme with other schemes is given in Section 5 and the Section 6 concludes the paper.

2. Preliminaries

In this section, we briefly review the concepts of elliptic curve bilinear pairing, some relevant computational problems, the computational model definition and the security properties of an ID-SDVPS scheme.

2.1. Bilinear pairings

Let G_q be an additive cyclic group with the prime order $q(\geqslant 2^k$, where k is a security parameter), and let G_m be another multiplicative group of the same order q. In addition, let $\hat{e}: G_q \times G_q \rightarrow G_m$ be a bilinear mapping with the following properties:

- **Bilinearity:** $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all P, $Q \in G_q$ and $a, b \in_R Z_q^*$.
- **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.
- **Computability:** There is an efficient algorithm that can efficiently compute $\hat{e}(P, Q)$ for all P, $Q \in G_q$.

Bilinear Diffie–Hellman Parameter Generator (BDHPG): A BDHPG $\mathcal{G}$ is defined as a probabilistic polynomial time algorithm that takes the security parameter k 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 defined some computational problems on the elliptic curve group and bilinear pairings. Several cryptographic schemes have been proposed that are mainly based on the hardness of the following problems, which are assumed to be computationally intractable by any polynomial time bounded algorithm.

- **Computational Diffie–Hellman (CDH) Problem:** For any $a, b \in_R Z_q^*$, given (P, aP, bP) belongs to G_q, compute abP.
- **Decisional Diffie–Hellman (DDH) Problem:** For any $a, b, c \in_R Z_q^*$, given (P, aP, bP, cP) belongs to G_q, decide whether $c = ab \bmod q$ holds.
- **Bilinear Diffie–Hellman (BDH) Problem:** For any $a, b, c \in_R Z_q^*$, given (P, aP, bP, cP) belongs to G_q, compute $\hat{e}(P, P)^{abc}$.
- **Bilinear Diffie–Hellman (BDH) Assumption:** If $\mathcal{G}$ is a BDH parameter generator, the advantage $Adv_{\mathcal{G},\mathcal{C}}^{BDH}(k)$ of an algorithm $\mathcal{C}$ for solving the BDH problem is defined to be the probability that the algorithm $\mathcal{C}$ outputs $\hat{e}(P, P)^{abc}$ from the input (G_q, G_m, e, P, aP, bP, cP), where (G_q, G_m, e) are the outputs of $\mathcal{G}$ for a sufficiently large security parameter k, (P, aP, bP,cP) $\in G_q$ and $a, b, c \in_R Z_q^*$. The BDH assumption is that the advantage $Adv_{\mathcal{G},\mathcal{C}}^{BDH}(k)$ is negligible for all the efficient algorithms $\mathcal{C}$.
- **Decisional Bilinear Diffie–Hellman (DBDH) Problem:** Given a randomly chosen $P \in G_q$ and (aP, bP, cP) $\in G_q$, where $a, b, c \in_R Z_q^*$ and $g \in G_m$, decide whether $g = \hat{e}(P, P)^{abc}$ holds.
- **Gap Bilinear Diffie–Hellman (GBDH) Problem:** Given a randomly chosen $P \in G_q$ as well as (aP, bP, cP) $\in G_q$ for any $a, b, c \in_R Z_q^*$, compute $\hat{e}(P, P)^{abc}$ with the help of the DBDH oracle.

2.3. Definition of an ID-SDVPS scheme

In general, an ID-SDVPS scheme comprises four entities, such as the PKG, the original signer, the proxy signer and a designated verifier. The participating entities and their roles in the ID-SDVPS scheme are defined as follows:

- **Private Key Generator (PKG):** The PKG is a trusted authority who is responsible for setting up the system's parameter and generating the private key for all the entities that exist in the system.
- **Original signer:** The original signer first defines a warrant and then delegates his signing capability to a proxy signer.

- **Proxy signer:** The proxy signer computes a proxy private key that is based on the original signer's warrant and delegation information. He then signs some messages for the designated verifier on behalf of the original signer.
- **Designated verifier:** The designated verifier checks the correctness of the received proxy signature according to the predefined verifying equation and then convince himself about the agreement of the original signer on the signed message.

An ID-SDVPS scheme comprises the following algorithms:

- **Setup:** It takes a security parameter $k \in Z^+$ as input and outputs the system's parameter Ω and a pair of master private/public key (msk, mpk).
- **Extract:** It takes a security parameter k, a system parameter Ω and the master private key msk as input, and it outputs the valid private/public key pair (S_i, Q_i) for an entity ID_i.
- **DGen:** On input of the system's parameter Ω, the original signer's private key S_i and a warrant m_w, the ***DGen*** algorithm outputs a valid delegation W for the proxy signer ID_j.
- **DVerify:** It takes the original signer's public key Q_i and a delegation W as input and outputs *accept* if W is valid; otherwise, it outputs *reject*.
- **PKGen:** Given the proxy signer's private key S_j and a delegation W, it outputs a valid proxy private/public key pair (S_P, Q_P).
- **PSGen:** It takes the proxy private key S_P, the delegation W, the designated verifier's public key Q_k and a signed message $m \in \{0,1\}^*$ as input and generates a proxy signature σ for the designated verifier ID_k.
- **PSVerify:** This algorithm accepts a message $m \in \{0,1\}^*$, a warrant W, a signature σ, the public key pair (Q_i, Q_j) of the original signer and the proxy signer, the designated verifier's private key S_k and returns *accept* if the signature σ is valid; otherwise, it returns *reject*.
- **Transcript simulation**: This algorithm takes a message $m \in \{0,1\}^*$, a warrant W and the designated verifier's private key S_k to generate a simulated proxy signature σ', which is identical to the original designated verifier proxy signature σ that was generated by the proxy signer.

2.4. Security properties of an ID-SDVPS scheme

The ID-SDVPS scheme is expected to satisfy *distinguishability*, *strong verifiability*, *strong undeniability*, *strong identifiability*, *prevention of misuse* and *strong unforgeability* security properties, which are briefly given below:

- **Distinguishability:** The designated verifier can distinguish the proxy signature from the original signer's normal signature easily.
- **Strong verifiability:** The designated verifier is convinced of the agreement of the original signer on the signed message.
- **Strong undeniability:** Once the proxy signer generates a valid proxy signature on behalf of the original signer, however, he cannot deny the signature generation at the later time.
- **Strong identifiability:** The identities of the original signer and the proxy signer can be determined by anyone from the proxy signature.
- **Prevention of misuse:** The proxy private key cannot be used to sign any message, i.e., the proxy signer can sign only those messages that have been approved by the original signer.
- **Strong unforgeability:** The original signer and other third parties, except for the designated verifier, cannot generate a valid proxy signature without the proxy private key or the designated verifier's private key.

3. The proposed ID-SDVPS scheme

In this section, the description of our proposed ID-SDVPS scheme is given. In our scheme, it is assumed that *Alice* is the original signer and has the identity ID_A, *Bob* is the proxy signer and has the identity ID_B and *Cindy* is the designated verifier and has the identity ID_C. We denote them as ID_i, where $i \in \{A, B, C\}$, and we consider (Q_i, S_i) to be their public/private key pair. As stated earlier, the ideas of the schemes (Huang et al., 2008; Sun et al., 2010; Yoon, 2011) are combined in this paper to construct a new ID-SDVPS scheme that has strong security in the random oracle model and less computational cost, which is described as follows:

- **Setup:** Given a security parameter $k \in Z^+$, the PKG does the following:
  (a) Choose an additive cyclic group $(G_q, +)$ of prime order q, a multiplicative group $(G_m, \cdot)$ of the same order q and an admissible bilinear map $\hat{e} : G_q \times G_q \rightarrow G_m$.
  (b) Choose a number $s \in_R Z_q^*$ and a generator P of G_q, and then compute $P_{pub} = sP$, where $(msk, mpk) = (s, sP)$ is a master private/public key pair of PKG.
  (c) Choose three secure and one-way cryptographic hash functions, which are defined as $H_1 : \{0,1\}^* \rightarrow G_q$, $H_2 : \{0,1\}^* \times G_q \rightarrow Z_q^*$ and $H_3 : \{0,1\}^* \times G_m \rightarrow Z_q^*$.
  (d) Publish the system's parameter $\Omega = \{G_q, G_m, \hat{e}, q, P, P_{pub}, H_1, H_2, H_3\}$ while keeping the master private key $msk = s$ secret.

- **Extract:** Given an identity ID_i, PKG does the following:
  (a) Compute a private key $S_i = sQ_i$ and send it to the user ID_i through a secure channel, where the user's public key $Q_i = H_1(ID_i)$ is easily computed from ID_i by anyone. Note that the hash function H_1 is called the Map-To-Point function, and it converts a random string ID_i to an elliptic curve point Q_i of the group G_q (Boneh and Franklin, 2001; Islam and Biswas, 2012b).
  (b) Accordingly, *Alice*, *Bob* and *Cindy* obtain their key pairs (S_A, Q_A),(S_B, Q_B) and (S_C, Q_C) from PKG.

- **DGen:** The original signer *Alice* creates a warrant m_w that keeps the record of the identities of *Alice* and *Bob*, the types of messages that *Bob* can sign, the proxy validity period and so on. Then, *Alice* chooses a number $x \in_R Z_q^*$ and generates the delegation that is based on the signature proposed in (Sun et al., 2010), as follows:
  (a) Compute $R = xP$ and $h = H_2(m_w, R)$, where m_w and R are concatenated before applying the hash function H_2.

(b) Compute $V = xP_{pub} + hS_A$.
(c) Output (R, V) as a delegation on the warrant m_w and send it to *Bob* for verification and proxy key generation.

- **DVerify:** Given a delegation (R, V), *Alice*'s identity ID_A and the public key Q_A, *Bob* does the following:
  (a) Computes $h = H_2(m_w, R)$.
  (b) Checks whether the equation $\hat{e}(V, P) = \hat{e}(R + hQ_A, P_{pub})$ holds. If so, *Bob* accepts the delegation (R, V); otherwise, he rejects it.

- **PKGen:** After validating the delegation (R, V), *Bob* generates the proxy private/public key pair with the help of *Alice*'s delegation and his private key S_B, as follows:
  (a) The proxy private key is computed as $S_P = V + hS_B$. Note that

$$\begin{aligned} S_P &= V + hS_B \\ &= xP_{pub} + hS_A + hS_B \\ &= xsP + h(sQ_A + sQ_B) \\ &= s[xP + h(Q_A + Q_B)] \\ &= s[R + h(Q_A + Q_B)] \\ &= sQ_P \end{aligned}$$

  (b) Then, the proxy public key can be computed as $Q_P = R + h(Q_A + Q_B)$.

- **PSGen:** To generate a valid strong designated verifier proxy signature on a given message $m \in \{0,1\}^*$, *Bob* computes the message according to the ID-SDVS schemes proposed in (Huang et al., 2008; Yoon, 2011), as follows:
  (a) Choose a number $r \in_R Z_q^*$ and compute $T = \hat{e}(S_P, Q_C)^r$.
  (b) Compute $\sigma = H_3(m, m_w, T)$.
  (c) Then, send the proxy signature (m_w, m, R, r, σ) to *Cindy* for verification.

- **PSVerify:** To *accept* or *reject* the proxy signature (m_w, m, R, r, σ), *Cindy* executes the following operations:
  (a) Check that the type of message m is the same as defined in the m_w; continue if the message and the warrant are valid and correspond to each other; *reject* otherwise.
  (b) Compute $Q_P = R + h(Q_A + Q_B)$ and $\overline{T} = \hat{e}(rQ_P, S_C)$.
  (c) Compute $\bar{\sigma} = H_3(m, m_w, \overline{T})$.
  (d) *Accept* the proxy signature (m_w, m, R, r, σ) if $\bar{\sigma} = \sigma$ holds; otherwise, *reject* it.

- **Transcript simulation:** For a given message $m \in \{0,1\}^*$, *Cindy* uses his/her secret key S_C to generate a valid simulated designated verifier proxy signature, as given below:
  (a) Choose a number $\hat{y} \in_R Z_q^*$ and compute $\widehat{T} = \hat{e}(\hat{y}Q_P, S_C)$.
  (b) Compute $\hat{\sigma} = H_3(m, m_w, \widehat{T})$.
  (c) Then, the simulated signature $(m_w, m, R, \hat{y}, \hat{\sigma})$ is a valid designated verifier proxy signature, and it can be easily verified by the verification equation $\bar{\sigma} = \sigma$.

The proposed ID-SDVPS scheme is further illustrated in Figure 1.

4. Analysis of the proposed ID-SDVPS scheme

In this section, we first check the correctness of the proposed ID-SDVPS scheme, and then, we analyze the scheme formally, using the random oracle model (Bellare and Rogaway, 1993) and the AVISPA tool (AVISPA, 2005, 2013).

4.1. Correctness of the proposed ID-SDVPS scheme

The correctness of the proposed ID-SDVPS scheme is as follows:

(a) Given $R = xP$, $h = H_2(m_w, R)$ and $V = xP_{pub} + hS_A$, the delegation (R, V) is valid because we have

$$\begin{aligned} &\hat{e}(R + hQ_A, P_{pub}) \\ &= \hat{e}(xP + hQ_A, sP) \\ &= \hat{e}(xsP + hsQ_A, P) \\ &= \hat{e}(xP_{pub} + hS_A, P) \\ &= \hat{e}(V, P) \end{aligned}$$

Therefore, the proxy private and public keys $S_P = V + hS_B = sQ_P$ and $Q_P = R + h(Q_A + Q_B)$ are also valid.
(b) The proxy signature (m_w, m, R, r, σ) is also correct and consistent because we obtain

$$\begin{aligned} \overline{T} &= \hat{e}(rQ_P, S_C) \\ &= \hat{e}(Q_P, S_C)^r \\ &= \hat{e}(R + h(Q_A + Q_B), S_C)^r \\ &= \hat{e}(xP + h(Q_A + Q_B), sQ_C)^r \\ &= \hat{e}(sxP + sh(Q_A + Q_B), Q_C)^r \\ &= \hat{e}(xP_{pub} + h(S_A + S_B), Q_C)^r \\ &= \hat{e}(xP_{pub} + hS_B + hS_A, Q_C)_r \\ &= \hat{e}(xP_{pub} + hS_A + hS_B, Q_C)^r \\ &= \hat{e}(V + hS_B, Q_C)^r \\ &= \hat{e}(S_P, Q_C)^r \\ &= T \end{aligned}$$

Thus, we can have $\bar{\sigma} = H_3(m, m_w, \overline{T}) = H_3(m, m_w, T) = \sigma$, and hence, the correctness of the proposed ID-SDVPS scheme is proved.

4.2. Security analysis

In this section, we prove that the proposed scheme satisfies all the security properties of an ID-SDVPS scheme, as defined in (Lee et al., 2001; Cha and Cheon, 2003; Wu et al., 2007; Lin et al., 2011). We consider the random oracle model (Bellare and Rogaway, 1993) and show that the proposed ID-SDVPS scheme is strongly unforgeable against all types of adversaries.

- **Distinguishability:** The proxy signatures generated by the proxy signer are distinguishable by everyone of the normal signatures of the proxy signer. Because the normal signature generated by *Bob* for *Cindy* can be verified only by *Bob*'s public key and *Cindy*'s private key, in our proposed scheme, *Cindy* uses his private key and *Bob*'s proxy public key to verify the validity of the proxy signature (m_w, m, R, r, σ). Note that the proxy public key (Q_P) is computed

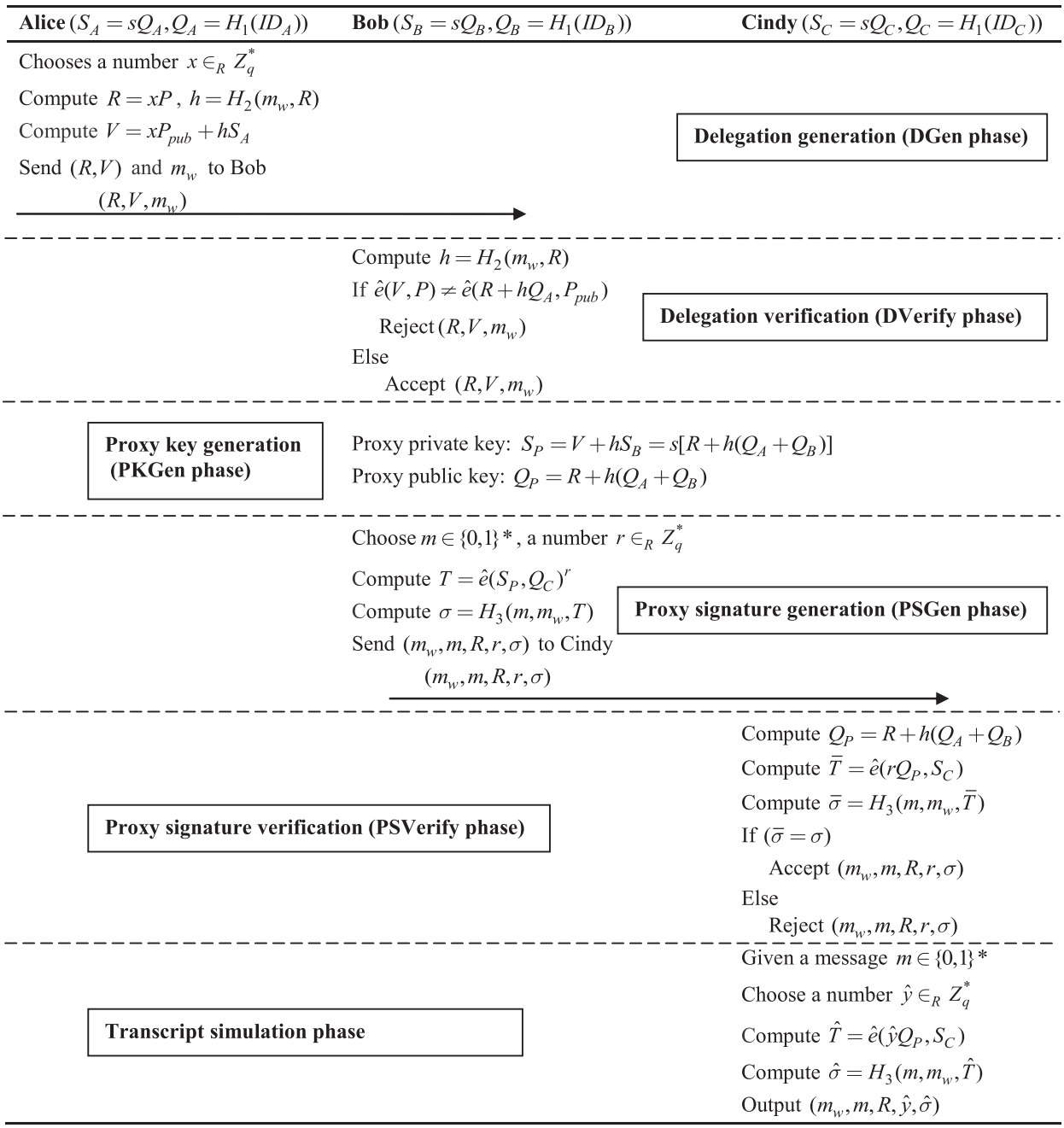

Figure 1 The proposed ID-SDVPS scheme.

by using the public keys of *Bob* and *Alice* and other public parameters. Therefore, the verification process of a proxy signature and a normal signature are not identical. Hence, *Cindy* can distinguish the normal signature from the proxy signature easily.

- **Strong verifiability:** The designated verifier *Cindy* is convinced about the agreement of *Alice* on the signed message. Now, the received proxy signature (m_w, m, R, r, σ) is valid if the equation $\bar{\sigma} = \sigma$ holds. To verify this proxy signature, *Cindy* computes $Q_P = R + h(Q_A + Q_B), \overline{T} = \hat{e}(rQ_P, S_C)$ and $\bar{\sigma} = H_3(m, m_w, \overline{T})$, which show the requirement of the warrant m_w. Thus, *Cindy* can find *Alice*'s identity from the warrant m_w, and *Alice*'s agreement on the signed message is directly understood by *Cindy*.
- **Strong undeniability:** On behalf of *Alice*, *Bob* generates a valid proxy signature for *Cindy*; however the signature generation at the later time cannot be denied by *Bob*. In our scheme, *Bob* generates the proxy signature (m_w, m, R, r, σ) for *Cindy*, where, $\sigma = H_3(m, m_w, T)$ and $T = \hat{e}(S_P, Q_C)^r$, and this signature can be verified by using the verification equation $\bar{\sigma} = \sigma$, where $\overline{T} = \hat{e}(rQ_P, S_C)$ and $\bar{\sigma} = H_3(m, m_w, \overline{T})$. Because the final verification of the proxy signature requires *Cindy*'s private key and *Bob*'s proxy public key, this in turn proves that the signature was created by *Bob* because he can only compute the proxy signature by using his proxy private key S_P, which was created by his private key and *Alice*'s delegation information.

- **Strong identifiability:** Note that anyone can determine the identity of the corresponding proxy signer from a valid proxy signature. In our scheme, the delegation information (R, V) is generated by *Alice* from the warrant m_w, which specifies the identities of *Alice* and *Bob*, the types of messages that *Bob* can sign, the validity period and more. Therefore, *Cindy*, from the warrantm_w, can determine the identity of *Bob* directly.
- **Prevention of misuse:** The proxy private key cannot be used to sign other messages; those messages are not authorized by the original signer. In our scheme, *Bob* cannot sign any messages that have not been authorized by *Alice*. For this reason, *Alice* generates a warrant m_w that retains the necessary records of the proxy information, such as the identities of *Alice* and *Bob*, the restrictions on the messages that *Bob* can sign, a validity period for the delegation of the signing power and more items of relevance. Therefore, if *Bob* signs other messages that have not been authorized by *Alice*, *Cindy* can detect this action easily by checking the warrant m_w. Hence, the property *prevention of misuse* is achieved in our scheme.
- **Strong unforgeability:** On behalf of the original signer, the proxy signer can create a valid proxy signature for the designated verifier, but a proxy signature cannot be generated by the original signer and other third parties, except for the designated verifier. This situation occurs because the outsider does not know either the proxy private key or the private key of the designated verifier. Before discussing the strong unforgeability of our proxy signature scheme in the random oracle model, we first addressed the following types of adversaries that could exist in the system:
  (a) **Type I:** The adversary knows only the public keys of *Alice* and *Bob*.
  (b) **Type II:** The adversary knows only the public keys of *Alice* and *Bob* and also knows the private key of *Bob*.
  (c) **Type III:** The adversary knows only the public keys of *Alice* and *Bob* and also knows the private key of *Alice*.

From the above, we can say that, if the proxy signature scheme is unforgeable against Type II and III adversaries, it is also unforgeable against a Type I adversary.

4.2.1. Unforgeability against a Type II adversary

Theorem 1. *If there exists a probabilistic polynomial time bounded Type II adversary $\mathcal{A}_{II}$ who can break our proposed ID-SDVPS scheme under the adaptively chosen message and identity attacks in the random oracle model, then there exists an algorithm $\mathcal{C}$ that can be used by $\mathcal{A}_{II}$ to solve the CDH problem in the elliptic curve group.*

Proof. The Type II adversary $\mathcal{A}_{II}$ knows the public keys of *Alice* and *Bob* and also knows the private key of *Bob*. The unforgeability of the proposed proxy signature scheme against Type II adversary $\mathcal{A}_{II}$ requires that it is difficult to generate a valid delegation without *Alice*'s private key. If $\mathcal{A}_{II}$ generates a valid delegation, then he can compute the valid proxy private key easily and a valid strong designated verifier proxy signature as well. We can show that, if there exists an $\mathcal{A}_{II}$ who can forge a valid delegation of our scheme, then there exists an algorithm $\mathcal{C}$ to solve an instance of the CDH problem. Thus, $\mathcal{C}$ can compute abP for a given random instance (P, aP, bP) $\in G_q$, where $a, b \in_R Z_q^*$. To solve the CDH problem, $\mathcal{C}$ sets (msk, mpk) = (a, aP), *Bob*'s private/public key (S_B, P_B) = (acP, cP) and gives (S_B, P_B) to $\mathcal{A}_{II}$. Here, P is a generator of G_q, and $\{H_1, H_2, H_3\}$ is considered to be a random oracle. Then, $\mathcal{C}$ generates the system's parameter Ω by running the *Setup* algorithm and sends it to $\mathcal{A}_{II}$. Next, $\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries in the following way:

H_1 queries: $\mathcal{C}$ maintains a list LH_1 to record the hash queries and the corresponding output. When $\mathcal{A}_{II}$ submits a query on ID_i to the oracle H_1, then $\mathcal{C}$ looks into LH_1 and responds as

$$Q_i = H_1(ID_i) = \begin{cases} bP & \text{for} \quad i = A \\ r_i P & \text{otherwise}, \quad r_i \in_R Z_q^* \end{cases}$$

Extract queries: When $\mathcal{A}_{II}$ issues an *Extract* query on ID_i, $\mathcal{C}$ first makes a query on ID_i to the oracle H_1 and recovers Q_i from the LH_1list. Then, $\mathcal{C}$ replies to $\mathcal{A}_{II}$ as follows:

$$S_i = \begin{cases} \perp & \text{for} \quad i = A \\ r_i a P & \text{otherwise} \end{cases}$$

H_2 queries: Suppose that $\mathcal{A}_{II}$ submits an H_2 query on (m_w, R_i), $\mathcal{C}$ searches the list LH_2 and returns the previous value if such a value is found in LH_2; otherwise, $\mathcal{C}$ selects an $h_i \in_R Z_q^*$, outputs it as the answer and adds (m_w, R_i, h_i) to LH_2.

DGen queries: On receiving a *DGen* query on the warrant m_w with the original signer's identity ID_i, $\mathcal{C}$ first recovers the values (ID_i, Q_i) and (m_w, R_i, h_i) from LH_1 and LH_2, respectively, and then performs the following:

(a) If $ID_i \neq ID_A$, then $\mathcal{C}$ computes the private key $S_i = r_i aP$, chooses $x_i \in_R Z_q^*$ and computes the delegation as
  (i) $R_i = x_i P$.
  (ii) $V_i = x_i P_{pub} + h_i S_i$.

(b) Otherwise, quit the protocol.

Finally, $\mathcal{C}$ returns (R_i, V_i) as the delegation on m_w, with the original signer's identity ID_i.

DVerify queries: On receiving a *DVerify* query (R_i, V_i) on m_w with the original signer's identity ID_i, $\mathcal{C}$ recovers (ID_i, Q_i) and (m_w, R_i, h_i) from LH_1 and LH_2, respectively, and performs the following:

(a) If $ID_i = ID_A$ holds, then $\mathcal{C}$ aborts.
(b) Otherwise, $\mathcal{C}$ verifies the correctness of the delegation (R_i, V_i) with the equation $\hat{e}(V_i, P) = \hat{e}(R_i + h_i Q_i, P_{pub})$ and then outputs the result. Note that the delegation (R_i, V_i) is valid if ID_i and m_w have never been queried during the *Extract* and *DGen* oracles, respectively.

Finally, $\mathcal{A}_{II}$ outputs (R_i^*, V_i^*) with h_i * on m_w^* as a valid delegation with the original signer's identity ID_i. Based on the *forking lemma* (Pointcheval and Stern, 2000), $\mathcal{C}$ recovers another tuple (m_w^*, R_i^*, h_i^*) from the list LH_2 and then replays the random oracle with the same random tape, but with different choices of the hash value of H_2. In other words, on the same warrant m_w^*, $\mathcal{C}$ obtains another forged delegation (R_i^*, V_i) with h_i such that $h_i^* \neq h_i$ and $V_i^* \neq V_i$. Finally, $\mathcal{C}$ has two valid delegations (R_i^*, V_i^*) and (R_i^*, V_i) on the same warrant m_w^*. Therefore, the verifying equations $\hat{e}(V_i^*, P) =$

$\hat{e}\left(R_i^* + h_i^* bP, aP\right) = \hat{e}\left(a\left[R_i^* + h_i^* bP\right], P\right)$ and $\hat{e}(V_i, P) = \hat{e}\left(R_i^* + h_i bP, aP\right) = \hat{e}\left(a\left[R_i^* + h_i bP\right], P\right)$ hold; in other words, we have $V_i^* = a\left[R_i^* + h_i^* bP\right]$ and $V_i = a\left[R_i^* + h_i bP\right]$. Then, $\mathcal{C}$ can solve the CDH problem with the instance (P, aP, bP), as follows: $V_i^* - V_i = \left(h_i^* - h_i\right) abP$ and $abP = \left(V_i^* - V_i\right)\left(h_i^* - h_i\right)^{-1}$. □

4.2.2. Unforgeability against a Type III adversary

Theorem 2. *The proposed ID-SDVPS scheme is existentially unforgeable against the adaptive chosen message and the identity adversary $\mathcal{A}_{III}$ of Type III provided by the GBDH problem is intractable in the elliptic curve group.*

Proof. The Type III adversary $\mathcal{A}_{III}$ knows the public keys of *Alice* and *Bob* and also knows the private key of *Alice*. Therefore, $\mathcal{A}_{III}$ can generate a valid delegation, but not a valid proxy private key because he does not know *Bob*'s private key. Thus, $\mathcal{A}_{III}$ attempts to generate a valid proxy signature without the proxy private key or the private key of *Cindy*. We now show that, if $\mathcal{A}_{III}$ can generate a forged proxy signature, then there must exist an algorithm $\mathcal{C}$ that can use $\mathcal{A}_{III}$ to solve an instance of a GBDH problem.Thus, for a given random instance $(P, aP, bP, cP) \in G_q$, where $a, b, c \in_R Z_q^*$, $\mathcal{C}$ can compete $\hat{e}(P, P)^{abc}$ with the help of the DBDH oracle. To solve a GBDH problem, $\mathcal{C}$ sets $(msk, mpk) = (a, aP)$, $S_A = aa'P$, $Q_A = a'P$, $Q_B = aP$ and $Q_C = bP$, and sends $(Q_A, Q_B, Q_C, Q_P, S_A)$ to $\mathcal{A}_{III}$, where $a', a, b, c \in {}_R Z_q^*$. Next, $\mathcal{C}$ generates the system's parameter Ω by running the *Setup* algorithm and sends it to $\mathcal{A}_{III}$ and answers $\mathcal{A}_{III}$'s queries, as follows:

H_1 queries: In this case, $\mathcal{A}_{III}$ can ask at most q_{H1} times H_1 hash queries, and $\mathcal{C}$ maintains a list LH_1 to record the hash queries and the corresponding outputs. When $\mathcal{A}_{III}$ submits a query ID_i to the oracle H_1, then $\mathcal{C}$ looks into the list LH_1 and responds as follows:

$$Q_i = H_1(ID_i) = \left\{ \begin{array}{lll} a'P & for\ i = A & \\ bP & \text{for} \quad i = B & \\ cP & \text{for} \quad i = C & \\ r_i P & \text{otherwise} & r_i \in_R Z_q^* \end{array} \right\}$$

Extract queries: In this case, $\mathcal{A}_{III}$ can ask at most q_e *Extract* queries. When $\mathcal{A}_{III}$ submits an *Extract* query on ID_i, $\mathcal{C}$ recovers the tuple (ID_i, Q_i) from LH_1 and then responds as follows:

$$S_i = \left\{ \begin{array}{ll} \perp & \text{for} \quad i = A, B, C \\ r_i aP & \text{otherwise} \end{array} \right\}$$

H_2 queries: In this case, $\mathcal{A}_{III}$ can ask at most q_{H2} times H_2 hash queries. When receiving an H_2 query on (m_w, R_i), $\mathcal{C}$ first searches the list LH_2 and returns the old value defined in LH_2 if such a value is found. Otherwise, $\mathcal{C}$ selects $h_i \in_R Z_q^*$, sets $H_2(m_w, R_i) = h_i$ and includes the tuple (m_w, R_i, h_i) into the list LH_2.

DGen queries: In this case, $\mathcal{A}_{III}$ can ask at most q_d times *DGen* queries. Assume that $\mathcal{A}_{III}$ makes a *DGen* query to $\mathcal{C}$ on the warrant m_w with the original signer's identity ID_i. Here, $\mathcal{C}$ knows the original signer's private/public key pair $(S_A = aa'P, Q_A = a'P)$, and then $\mathcal{C}$ executes *DGen* queries on (ID_i, m_w) to compute the corresponding delegation (R_i, V_i).

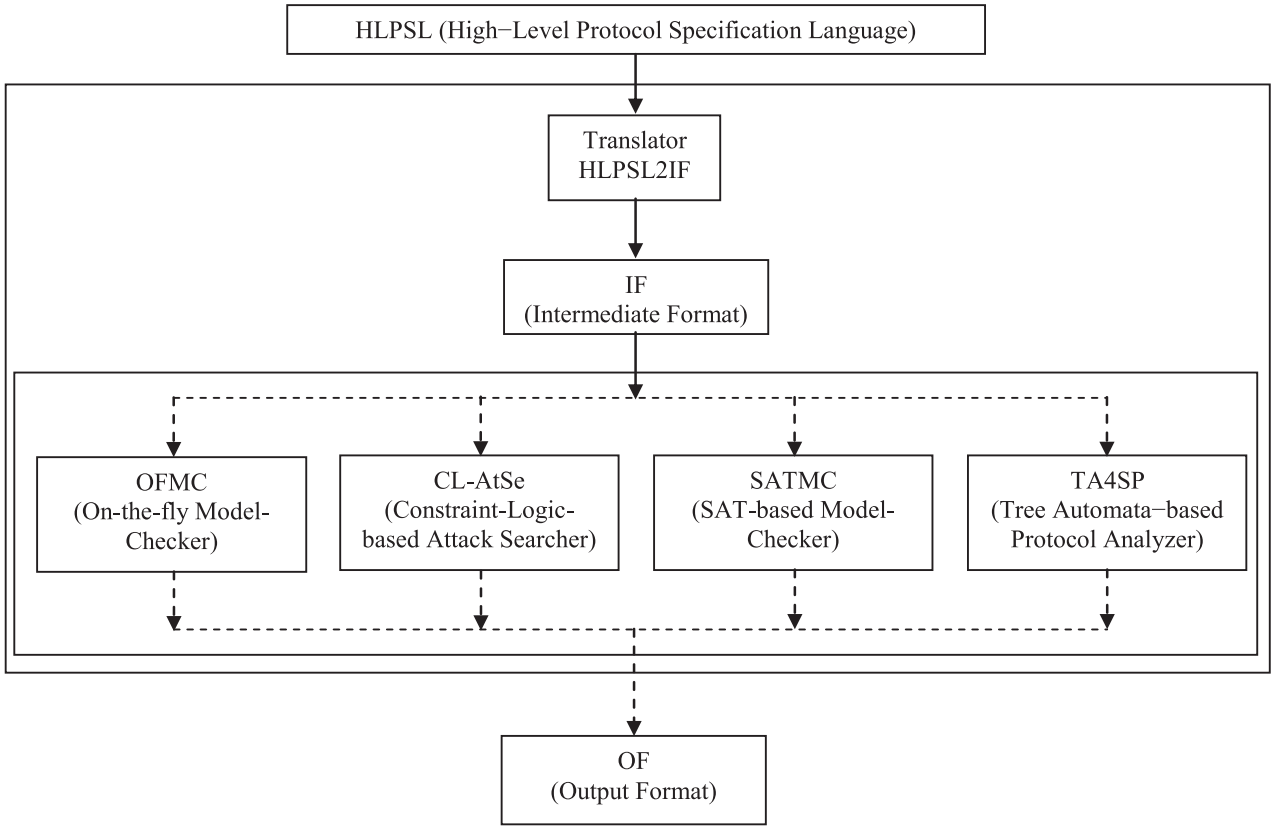

Figure 2 Architecture of the AVISPA tool.

```
role alice(
  A, B, C : agent,
  P, Ppub: symmetric_key,
  H2, H3 : hash_func,
  Ebilinear, Minus, Union, Pred : hash_func,
  Snd, Rcv: channel(dy))

played_by A
  def=
    local State : nat,
    Sa, Qa, K, Xr, V, Mw, Sig, Lt : text
    const alice_bob_na, bob_alice_nb, bob_cindy_nc,
    cindy_bob_nd, subs1, subs2 : protocol_id

init State := 0

transition

1. State = 0 /\ Rcv(start) =|>
State' := 1
/\ K':= new()
/\ secret({K', Sa}, subs1, A)
/\Xr' := Pred(K', P)
/\Lt' :=H2(Mw, Xr')
/\V' := Union(Pred(K, Ppub),  Pred(Lt', Sa))
/\ Snd(Mw.Xr'.V')
/\ witness(A, B, bob_alice_nb, K')
end role
```

Figure 3 Role specification of the original signer (*Alice*) in HLPSL language.

PKGen queries: When $\mathcal{A}_{III}$ queries a *PKGen* of the proxy signer for m_w, $\mathcal{C}$ computes the private key $S_j = r_j aP$ of the proxy signer, the proxy private key $S_{Pj} = V_i + h_i S_j$ and $\mathcal{A}$ responds to $\mathcal{A}_{III}$ with S_{Pj}.

H_3 queries: The adversary $\mathcal{A}_{III}$ is allowed to ask at most q_{H3} times H_3 hash queries, and $\mathcal{C}$ maintains an H_3 oracle list LH_3 that contains the tuples of the form (m_i, t_i, σ_i, $coin_i$). Here, (m_i, t_i) and σ_i are the input and output, respectively. Next, if $t_i \cdot \hat{e}(V_i + h_i S_j, Q_k)^{1/h_i} \cdot \hat{e}(V_i, -Q_k)^{1/h_i} = \hat{e}(P,P)^{abc}$ holds, then $coin_i = 1$; otherwise, $coin_i = 0$. On receiving an H_3 query on (m_i, t_i), $\mathcal{C}$ submits $(t_i \cdot \hat{e}(V_i + h_i S_j, Q_k)^{1/h_i} \cdot \hat{e}(V_i, -Q_k)^{1/h_i}, aP, bP, cP)$ to the DBDH oracle, which informs $\mathcal{C}$ whether $t_i \cdot \hat{e}(V_i + h_i S_j, Q_k)^{1/h_i} \cdot \hat{e}(V_i, -Q_k)^{1/h_i} = \hat{e}(P,P)^{abc}$ holds and performs the following:

(a) If $t_i \cdot \hat{e}(V_i + h_i S_j, Q_k)^{1/h_i} \cdot \hat{e}(V_i, -Q_k)^{1/h_i} = \hat{e}(P,P)^{abc}$ holds, then $\mathcal{C}$ sets $coin_i = 1$ and looks into the list LH_3 and produces the following:
   (i) If a tuple of the form (m_i, $\perp$, σ_i,1) is found in LH_3, then $\mathcal{C}$ outputs σ_i.
   (ii) Otherwise, $\mathcal{C}$ chooses $\sigma_i \in_R Z_q^*$ such that there is no tuple of the form ($\cdot$, $\cdot$, σ_i, $\cdot$) in LH_3 and inserts (m_i, t_i, σ_i,1) into LH_3 and returns σ_i as the output.

(b) If $t_i \cdot \hat{e}(V_i + h_i S_j, Q_k)^{1/h_i} \cdot \hat{e}(V_i, -Q_k)^{1/h_i} \neq \hat{e}(P,P)^{abc}$, then $\mathcal{C}$ sets $coin_i = 0$ and chooses $\sigma_i \in_R Z_q^*$ such that there is no tuple of the form ($\cdot$, $\cdot$, σ_i, $\cdot$) in the list LH_3; then, $\mathcal{C}$ inserts (m_i, t_i, σ_i, 0) into LH_3 and returns σ_i as the output.

PSGen queries: The adversary $\mathcal{A}_{III}$ is allowed to ask at most q_s times *PSGen* queries. Suppose that $\mathcal{A}_{III}$ selects a message m_i and then submits a *PSGen* query to $\mathcal{C}$ on m_i. Then, $\mathcal{C}$ performs the following:

(a) If the tuple (m_i, t_i, σ_i,1) is found in LH_3, then $\mathcal{C}$ returns σ_i as the proxy signature.
(b) Otherwise, $\mathcal{C}$ chooses $\sigma_i \in_R Z_q^*$ such that there is no item ($\cdot$, $\cdot$, σ_i, $\cdot$) in LH_3. Then, $\mathcal{C}$ adds (m_i, $\perp$, σ_i,1) into the list LH_3 and returns σ_i as the output.

Thus, $\mathcal{A}_{III}$ gets σ_i as the proxy signature for the message m_i.

PSVerify queries: In this case, $\mathcal{A}_{III}$ can ask at most q_v times *PSVerify* queries. When receiving a *PSVerify* query on (m_i, σ_i), $\mathcal{C}$ searches the list LH_3 and answers $\mathcal{A}_{III}$'s query as follows:

(a) If there is no tuple of the form ($\cdot$, $\cdot$, σ_i, $\cdot$) in LH_3, then $\mathcal{C}$ rejects the signature σ_i.
(b) Otherwise, there is an item ($\cdot$, $\cdot$, σ_i, $\cdot$) in the list LH_3, and the conclusion is as follows:
   (i) If the item has the form (m_i, $\perp$, σ_i,1) or (m_i, t_i, σ_i,1), then $\mathcal{C}$ accepts the signature σ_i as a valid signature.
   (ii) Otherwise, $\mathcal{C}$ rejects the signature σ_i.

Therefore, (m_i, σ_i) is a valid proxy signature provided that σ_i has never been queried by the oracle H_3. At the end of this game, $\mathcal{A}_{III}$ outputs a valid forged proxy signature (m^*, σ^*), while m^* has never been queried to the *PSGen* oracle, and $\mathcal{C}$ returns σ^* as $H_3(m^*, t^*)$, i.e., we conclude that (m^*, t^*, σ^*,1) is in the LH_3 list and $t^* \cdot \hat{e}(V_i + h_i S_j, Q_k)^{1/h_i} \cdot \hat{e}(V_i, -Q_k)^{1/h_i} = \hat{e}(P,P)^{abc}$ holds. Thus, for a given random instance (P, aP, bP, cP) belongs to G_q where $a, b, c \in_R Z_q^*$, $\mathcal{C}$ solves the GBDH problem by computing $\hat{e}(P,P)^{abc}$ with the help of the DBDH oracle. □

4.3. Formal security validation using the AVISPA tool

Recently, the automated security validation tool for internet protocols and its applications have become widely employed in many cryptographic protocols, to analyze security systems formally. In the literature, many such tools have been found, and AVISPA (Automated Validation of Internet Security Protocols and Applications) (Oheimb, 2005; AVISPA, 2013) is one of the commonly used automated security validation tools (Farashei al., 2012; Basu et al., 2012; Das, 2012; Das et al., 2013). AVISPA is a push-button tool that was developed based on the Dolev-Yao intruder model (Dolev and Yao, 1983) for the automated validation of cryptographic protocols. In this model, an intruder has full control over the network, i.e., all the messages sent by principals go to the intruder, who can forward, modify, replay, suppress and synthesize them and can send them anywhere. In addition, an intruder can play the roles of the legitimate principals and can gain knowledge of the compromised principals; however, he cannot break the cryptography. The AVISPA tool has a modular and expressive formal language, called HLPSL (High Level Protocol

```
role bob(
  A, B, C : agent,
  P, Ppub : symmetric_key,
  H2, H3 : hash_func,
  Ebilinear, Minus, Union, Pred : hash_func,
  Snd, Rcv: channel(dy))

played_by B
  def=
    local State : nat,
    Sa, Sb, Sc, Sp, Qa, Qb, Qc, Qp, K, Xr, R,
    V, M, Mw, Sig, Lt : text
    const alice_bob_na, bob_alice_nb, bob_cindy_nc,
    cindy_bob_nd, subs1, subs2 : protocol_id

init State := 0

transition
1. State = 0
/\Rcv(Mw.Pred(K', P).Union(Pred(K, Ppub),  Pred(H2(Mw, Pred(K', P)), Sa)))=|>

State' := 1
/\Lt' :=H2(Mw, Xr)
/\Sp' := Union(V, Pred(Lt', Sb))
/\Qp':=Union(Xr, Pred(Lt', Union(Qa, Qb)))
/\M' := new()
/\R':= new()
/\ secret({K', Sa}, subs1, A)
/\ secret({Sp', R'}, subs2, B)
/\Sig':=H3(M, Mw, Ebilinear(Pred(Sp', R'), Qc))
/\ Snd(Mw.M'. Xr.R'.Sig')
/\ witness(B, C, cindy_bob_nd, Sp')
end role
```

Figure 4 Role specification of the proxy signer (*Bob*) in HLPSL language.

Specification Language), which is used for implementing the protocols and analysis of various security properties such as secrecy of keys, authentication, freshness and robustness against replay attacks. In addition, the AVISPA tool supports four back-ends/model checkers (See Figure 2), namely OFMC (On-the-fly Model-Checker), CL-AtSe (Constraint-Logic-based Attack Searcher), SATMC (SAT-based Model-Checker) and TA4SP (Tree Automata-based Protocol Analyzer) (AVISPA, 2005), each of which is briefly described below:

- **OFMC:** This back-end builds the infinite tree defined by the protocol analysis problem and executes different symbolic techniques to search the state space in a demand-driven way, i.e., on-the-fly. OFMC helps to detect attacks and verify the correctness of the protocol for a bounded number of sessions, but without limiting the number of messages that an intruder can generate.
- **CL-AtSe:** This back-end is used to detect the attacks on the protocol by using a set of constraints that are obtained by translating the security protocol specification written in Intermediate Format (IF). The detection of attacks and the translation of protocol specifications, which are designed based on the adversary's knowledge, are fully automated and are internally performed by the CL-AtSe model checker.
- **SATMC:** This back end is used to explore the state space through several symbolic techniques. Note that it also detects attacks on protocols and validates the security requirements by using a bounded number of sessions.
- **TA4SP:** Based on the propositional formulae and regular tree languages, this back-end approximates the intruder knowledge (over or under), using the unbounded number of sessions.

To analyze a cryptographic protocol with the AVISPA tool, the following steps must be executed:

Step 1. The protocol is to be coded in the HLPSL specification, which is a role-based language that helps to describe each participant's role and the composition roles for the representation of the scenarios of the basic roles.
Step 2. Using the translator HLPSL2IF, the HLPSL code is to be translated into IF, which contains some information about IF syntax for back-ends, the description of mathematical properties of

```
role cindy(
  A, B, C : agent,
  P, Ppub : symmetric_key,
  H2, H3 : hash_func,
  Ebilinear, Minus, Union, Pred : hash_func,
  Snd, Rcv : channel(dy))

played_by C
  def=
    local State : nat,
    Sa, Sb, Sp, Sc, Qa, Qb, Qp, Qc, K, Xr, R,
    V, M, Mw, Sig, Lt : text
    const alice_bob_na, bob_alice_nb, bob_cindy_nc,
    cindy_bob_nd, subs1, subs2 : protocol_id

init State := 0

transition
1. State = 0
   /\Rcv(Mw.M'.Xr'.R'.Sig')=|>
State' := 1
/\ secret({K, Sa}, subs1, A)
/\secret({Sp, Sb}, subs2, B)
end role
```

Figure 5 Role specification of the designated verifier (*Cindy*) in HLPSL language.

operators (e.g., exponentiation, bit-wise XOR, etc.) and the intruder's behavior.

Step 3. Finally, the IF specification is given to the back-ends of the AVISPA tool to analyze whether there are any active or passive attacks.

To validate and examine the security properties of our ID-SDVPS scheme, we implemented it using the HLPSL language in the AVISPA tool, and the role specifications of the original signer (*Alice*), the proxy signer (*Bob*) and the designated verifier (*Cindy*) are given in Figs. 3–5. We have simulated our scheme using SPAN (Security Protocol ANimator) for AVISPA. The proposed scheme is analyzed in the OFMC and CL-AtSe back-ends, and the results are shown in Figs. 6 and 7. From these simulation results, the proposed ID-SDVPS scheme indeed shows its strong security assurance against both passive and active attacks.

5. Efficiency analysis

In this section, we compare the efficiency of the proposed scheme with other schemes that were proposed in (Cao et al., 2005; Kang et al., 2009; Yang, 2010; Lee et al., 2010; Reddy et al., 2010). In our comparison, the following three time complexities for the different computations mentioned are used.

- $\mathbf{T_{EM}}$: Time complexity for executing an elliptic curve scalar point multiplication.
- $\mathbf{T_{BP}}$:Time complexity for executing a bilinear pairing operation.
- $\mathbf{T_{PX}}$: Time complexity for executing a pairing-based exponentiation.

To generate the proxy signature, the proxy signer computes $T = \hat{e}(S_P, Q_C)$ a priori, and hence, the time complexi-

```
% OFMC
% Version of 2006/02/13
SUMMARY
   SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS

PROTOCOL
 C:\progra~1\SPAN\testsuite\results\IDSDVPS Scheme.if

GOAL
  as_specified

BACKEND
  OFMC

COMMENTS
STATISTICS
   parseTime: 0.00s
   searchTime: 437.28s
   visitedNodes: 29565 nodes
   depth: 7 plies
```

Figure 6 Results of the formal security analysis of the proposed scheme using OFMC back-end.

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 C:\progra~1\SPAN\testsuite\results\IDSDVPS Scheme.if

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS
 Analysed  : 0 states
 Reachable  : 0 states
 Translation: 0.02 seconds
 Computation: 0.00 seconds
```

Figure 7 Results of the formal security analysis of the proposed scheme using CL-AtSe back-end.

Table 1 Comparison of the proposed scheme with other schemes.

Scheme	Cao et al. (2005)	Lee et al. (2010)	Kang et al. (2009)	Yang (2010)	Reddy et al. (2010)	Proposed
DGen	$2T_{EM}$	$2T_{BP} + 2T_{EM} + 1T_{PX}$	$1T_{BP} + 2T_{EM}$	$2T_{EM}$	$1T_{PX} + 2T_{EM}$	$3T_{EM}$
DVerify	$2T_{BP} + 1T_{EM}$	$2T_{BP} + 1T_{EM}$	$1T_{BP}$	$3T_{BP} + 1T_{PX}$	$2T_{BP} + 1T_{PX}$	$2T_{BP} + 1T_{EM}$
PKGen	$2T_{EM}$	–	–	$1T_{EM}$	$1T_{EM}$	$1T_{EM}$
PSGen	$1T_{BP} + 1T_{PX} + 3T_{EM}$	$2T_{BP} + 3T_{EM} + 1T_{PX}$	$1T_{BP} + 3T_{EM}$	$2T_{BP} + 2T_{PX} + 2T_{EM}$	$2T_{BP} + 1T_{PX} + 4T_{EM}$	$1T_{PX}$
PSVerify	$3T_{BP} + 2T_{EM}$	$3T_{BP} + 1T_{EM}$	$2T_{BP}$	$5T_{BP} + 4T_{PX}$	$3T_{BP} + 1T_{PX} + 1T_{EM}$	$1T_{BP} + 1T_{EM}$
Total	$6T_{BP} + 1T_{PX} + 10T_{EM}$	$7T_{BP} + 7T_{EM} + 2T_{PX}$	$5T_{BP} + 5T_{EM}$	$8T_{BP} + 7T_{PX} + 5T_{EM}$	$7T_{BP} + 4T_{PX} + 8T_{EM}$	$3T_{BP} + 1T_{PX} + 6T_{EM}$
Length	$4\|G_q\|$	$3\|G_q\|$	$3\|G_q\|$	$\|Z_q^*\| + 3\|G_q\|$	$2\|Z_q^*\| + 3\|G_q\|$	$2\|Z_q^*\| + \|G_q\|$
Security	Heuristic	Provable	Heuristic	Heuristic	Heuristic	Provable

ties of the proposed scheme actually result from the execution of six elliptic curve scalar point multiplications, three bilinear pairings and one pairing-based exponentiation operation. In Table 1, we compare the efficiency of the proposed scheme with other schemes based on the computation cost, the communication cost (length of the signature) and the security, which show that our scheme is more computation and communication efficient than others (Cao et al., 2005; Kang et al., 2009; Yang, 2010; Lee et al., 2010; Reddy et al., 2010). In addition, it has been found that the proposed scheme is unforgeable against active and passive attacks and is provably secure under the adaptively chosen message and identity attacks in the random oracle model based on CDH and GBDH assumptions.

6. Conclusions

Recently, elliptic curve cryptography and bilinear pairing have been extensively used to design different secure and efficient schemes for a variety of applications. Several ID-SDVPS schemes based on elliptic curve bilinear pairing have been proposed in recent years; however, they are neither secure against different attacks nor computationally efficient. In this paper, we proposed an efficient ID-SDVPS scheme, which is demonstrated to be provably secure with the hardness assumption of CDH and GBDH problems in the random oracle model against an adaptive chosen message and identity attacks under the different types of adversaries. Additionally, the formal validation of the proposed ID-SDVPS scheme is performed by using an automated validation tool called AVISPA, and the simulation results show that the scheme is unforgeable against active and passive adversaries. Compared with other existing schemes, the proposed scheme also provides better computation and communication efficiencies and, thus, is suitable for real-life applications.

Acknowledgments

We would like to thank Editor-in-Chief, Prof. Mansour M. Alsulaiman and anonymous reviewers for their valuable comments and suggestions on our work, which help to improve this paper. We gratefully acknowledge the financial supporters 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 (No. MIT (2)/2006-08/189/CSE). We are also thankful to the Department of Computer Science and Engineering, the Indian School of Mines, Dhanbad, India, for their research support because without their help, this work would not have been possible.

References

AVISPA, 2005. The AVISPA User Manual. http://www.avispa-project.org/publications.html.

AVISPA Web tool. Automated Validation of Internet Security Protocols and Applications.www.avispa-project.org/web-interface/. (Accessed on January, 2013).

Basu, A., Sengupta, I., Sing, J.K., 2012. Formal security verification of secured ECC based signcryption scheme. In: Proceedings of the Advances in Computer Science. Engineering & Applications, vol. 167. Springer-Verlag, Berlin Heidelberg, pp. 713–725.

Bellare, M., Rogaway, P., 1993. Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on Computer and Communications Security (CCS '93), pp. 62–73.

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

Cao, T., Lin D., Xue, R., 2005. ID-based designated-verifier proxy signatures. IEE Proceedings Communications 152 (6), 989–994.

Cha, J.C., Cheon, J.H., 2003. An identity-based signature from Gap Diffie–Hellman groups. In: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography (PKC '03), LNCS, vol. 2567, pp. 18–30.

Dai, J.Z., Yang, X.H., Dong, J.X., 2003. Designated-receiver proxy signature scheme for electronic commerce. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 1, pp. 384–389.

Das, A.K., 2012. A secure and effective user authentication and privacy preserving protocol with smart cards for wireless communications. Networking Science. http://dx.doi.org/10.1007/s13119-012-0009-8.

Das, A.K., Massand, A., Patil, S., 2013. Novel proxy signature scheme based on user hierarchical access control policy. Journal of King Saud University – Computer and Information Sciences. Elsevier, Vol. 25, 219–228.

Dolev, D., Yao, A.C., 1983. On the security of public-key protocols. IEEE Transactions on Information Theory 2 (29), 198–208.

Farash, M.S., Attari, M.A., Atani, R.E., Jami, M., 2012. A new efficient authenticated multiple-key exchange protocol from bilinear pairings. Computers & Electrical Engineering. http://dx.doi.org/10.1016/j.compeleceng.2012.09.004.

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

Gu, C., Zhu, Y., 2005. Probable security of ID-based proxy signature schemes. In: Proceedings of the ICCNM '05. LNCS, vol. 3619. Springer-Verlag, pp. 1277–1286.

Gu, C., Zhu, Y., 2008. An efficient ID-based proxy signature scheme from pairing. In: Proceedings of the Information Security and Cryptology. LNCS, vol. 4990. Springer, Berlin, Heidelberg, pp. 40–50.

Hess, F., 2002. Identity based signature schemes based on pairings. In: Proceedings of the 9th Annual International Workshop in Selected Areas in Cryptography (SAC '02). LNCS, vol. 2595. Springer-Verlag, pp. 310–324.

Huang, X., Susilo, W., Mu, Y., Zhang, F., 2008. Short designated verifier signature scheme and its identity-based variant. International Journal of Network Security 6 (1), 82–93.

Islam, S.H., Biswas, G.P., 2012a. Design of an efficient ID-based short designated verifier proxy signature scheme. In: Proceedings of the 1st International Conference on Recent Advances in Information Technology, pp. 48–53.

Islam, S.H., Biswas, G.P., 2012b. An Efficient and provably-secure digital signature scheme based on elliptic curve bilinear pairings. Theoretical and Applied Informatics 24 (2), 109–118.

Islam, S.H., Biswas, G.P., 2013. Provably secure certificateless strong designated verifier signature scheme based on elliptic curve bilinear pairings. Journal of King Saud University – Computer and Information Sciences 25, 51–61.

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

Kang, B., Boyd, C., Dawson, E., 2009. Identity-based strong designated verifier signature schemes: Attacks and new construction. Computers & Electrical Engineering 35 (1), 49–53.

Kim, H., Baek, J., Lee, B., Kim, K., 2001. 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.

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

Lal, S., Verma, V., 2006. Identity base strong designated verifier proxy signature schemes. Cryptography ePrint Archive Report 2006/394.Available at http://eprint.iacr.org/complete/2006/394.pdf.

Lee, J.S., Chang, J.H., Lee, D.H., 2010. Forgery attacks on Kang et al's identity-based strong designated verifier signature scheme and its improvement with security proof. Computers & Electrical Engineering 36 (5), 948–954.

Lee, B., Kim, H., Kim, K., 2001. Secure mobile agent using strong non-designated proxy signature. In: Proceedings of the ACISP2001. LNCS, vol. 2119. Springer-Verlag, pp. 474–486.

Lin, H.-Y., Wu, T.-S., Huang, S.-K., 2011. An efficient strong designated verifier proxy signature scheme for electronic commerce. Journal of Information Science and Engineering 28, 1–15.

Mambo, M., Usuda, K., Okamoto, E., 1996. Proxy signatures: delegation of the power to sign messages. IEICE Transactions on Fundamentals E79-A (9), 1338–1354.

Miller, V.S., 1985. Use of elliptic curves in cryptography. In: Proceeding of the Advances in Cryptology (CRYPTO '85). Springer-Verlag, pp. 417–426.

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

Oheimb, D.V., 2005. The high-level protocol specification language HLPSL developed in the EU project AVISPA. In: Proceedings of APPSEM '05 Workshop.

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

Pointcheval, D., Stern, J., 2000. Security arguments for digital signatures and blind signatures. Journal of Cryptology 13, 361–396.

Reddy, P.V., Rao, B.U., Gowri, T., 2010. ID-based directed proxy signature scheme from bilinear pairings. Journal of Discrete Mathematical Sciences & Cryptography 13 (5), 487–500.

Saeednia, S., Kremer, S., Markowitch, O., 2004. An efficient strong designated verifier signature scheme. In: Proceedings of the Information Security and Cryptology (ICISC '03). LNCS, vol. 2971. Springer-Verlag, pp. 40–54.

Shamir, A., 1984. Identity-based cryptosystems and signature schemes. In: Proceedings of the Advances in Cryptology (CRYPTO '84). Springer-Verlag, pp. 47–53.

Sun, S., Wen, Q., Jin, Z., Zhang, H., 2010. A new efficient ID-based strong designated verifier signature scheme. In: Proceedings of the 3rd International Symposium on Information Science and Engineering, pp. 137–141.

Tang, F., Lin, C., Li, Y., Zhang, S., 2011. Identity-based strong designated verifier signature scheme with full non-delegatability. In: Proceedings of the IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, pp. 800–805.

Wang, G.L. 2004. Designated-verifier proxy signatures for e-commerce. In: Proceedings of the IEEE 2004 International Conference on Multimedia and Expo, pp. 1731–1734.

Wang, B., 2008. A new identity based proxy signature scheme. Cryptology ePrint Archive Report. Available at: http://www.eprint.iacr.org/2008/323.

Wu, W., Mu, Y., Susilo, W., Seberry, J., Huang, X., 2007. Identity based proxy signature from pairing. In: Proceedings of the ATC-2007. LNCS, vol. 4610. Springer-Verlag, pp. 22–31.

Yang, Y., 2010. ID-based designated-verifier proxy signature scheme without a trusted party. In: Proceedings of the International Conference on Computer Application and System Modeling (ICCASM '10), vol. 7, pp. 191–193.

Yoon, E-J., 2011. An efficient and secure identity-based strong designated verifier signature scheme. Information Technology and Control 40 (4), 323–329.

Zhang, F., Kim, K., 2003. Efficient ID-based blind signature and proxy signature from bilinear pairing. In: Proceedings of the ACISP '03. LNCS, vol. 2727. Springer-Verlag, pp. 312–323.