An Efficient Certificate-based Verifiable Encrypted Signature Scheme Without Pairings

Rufen Huang* and Qiang Nong

The College of Computer Science, Minnan Normal University, Zhangzhou, China, 363000

Abstract: The verifiable encrypted signature is one of the basic fair exchange protocols. There are important applications, such as e-commerce and other cryptographic protocols. We incorporate the verifiable encrypted signature into the certificate-based signature to propose an efficient certificate-based verifiable encrypted signature scheme in the paper, which does not require any bilinear pairing operations. Then we analyze the scheme’s security under the elliptic curve discrete logarithm problem over a finite field. The analytic results show that our proposed scheme is proven secure, and our scheme simplifies the management of certificates and solves the problem of private key escrow. Compared with the other existing secure verifiable encrypted signature schemes, our certificate-based verifiable encrypted signature scheme provides greater efficiency and greatly reduces the cost of computation and communication, and achieves the same security level as other existing verifiable encrypted signature scheme.

Keywords: Certificate-based signature, discrete logarithm problem, ECC, provably secure, random oracle model, verifiable encrypted signature.

1. INTRODUCTION

Digital signature, which can prove authentication, integrity and non-repudiation, is one of the key techniques of information security. In recent years, various signature systems were proposed gradually, such as conventional Public Key Signature (PKS), Identity-based Public Key Signature (IB-PKS) [1], Certificateless Public Key Signature (CL-PKS) [2], and Certificate-based Public key Signature (CB-PKS) [3], etc. In addition, a lot of methods and tools for digital signatures have been invented as patents in order to promote the application of signature. Such as Patent US 7502934, titled “Electronic signatures” [4], is a method to the generation of digital signatures, Patent US 20080222418, titled “Signature Generation Device and Signature Verification Device” [5], provided a signature generation apparatus capable of preventing transcript attack on signature data, and Patent US Application 20100174910, titled “Public Key Encryption with Digital Signature Scheme” [6], is an improved encryption and digital signature system and method advantageously reduces byte size of the digital signature and reduction of costly computation overhead, and so on.

The conventional Public Key Cryptography (PKC) and PKS are generally considered to be costly to use and manage. Identity-based Public Key Cryptography (IB-PKC) and IB-PKS were introduced by Shamir [1] in 1984 to ease the certificate management of conventional PKC, and Patent US 7711113, titled “ID-based signature, encryption system and encryption method” [7] was invented, which is an ID-based encryption and signature technique according to which more efficient and higher speed processing is possible. But key escrow is IB-PKC’s inherent problem. Certificateless Public Key Cryptography (CL-PKC) and CL-PKS were proposed by Al-Riyami and Paterson in 2003 [2], whose original motivation is to solve the key escrow problem in IB-PKC and to simplify the certificate management process in conventional PKC. The related patent for the application of CL-PKC was invented in 2012, which is Patent US Application 20120023336, titled “System and method for designing secure client-server communication protocols based on certificateless public key infrastructure” [8], is a system and method for facilitating secure client-server communication using certificateless public key infrastructure etc. But the “trust level” [9] of CL-PKS is lower than the conventional PKS, and only reaches level 2. The Certificate-based Public Key Cryptography (CB-PKC) was introduced first by Gentry [10] in Eurocrypt’03, it is another cryptography primitive whose original motivation is the same as CL-PKC to simplify certificate’s management and to eliminate key escrow problem. A CB-PKC scheme combined a PKC scheme and an IB-PKC scheme to retain their respective advantages. Soon after, some patents about CB-PKC have continued to be invented, such as Patent US 7185195, titled “Certificate based digital rights management” [11], is a client device, in which the certificate is associated with one or more secure components, and Patent US Application 20130173914, titled “Method for Certificate-Based Authentication” [12], is a method for certificate-based authentication. The CB-PKC is similar to CL-PKC, the CB-PKC uses a certificate to replace the partial secret key of CL-PKC, while it does not require the use of any certificates in CL-PKC. In 2003, Kang, Park and Hahn extended CB-PKC to CB-PKS [3]. A CB-PKS scheme is a compromise between IB-PKS and PKS. It consists of a certifier and users, each user generates his own
private and public key, and request a certificate from the Certificate Authority (CA), and the certificate is implicit and can be used as a part of the signing key. The CB-PKS schemes [13-15] inherit merits of IB-PKS and PKS. Simplifies the management of certificates in traditional PKS, solves the problem about private key escrow in IB-PKS and overcomes the problem of lower trust level in CL-PKS.

The verifiable encrypted signature (VES) is a special extension of ordinary signature primitive that was first proposed by Asokan et al. [16], which can construct optimistic fair exchange protocol and there are useful for many cryptographic protocols. A VES involves three participants, namely a signer, a verifier and an adjudicator. The signer creates a VES by encrypting an ordinary signature with adjudicator’s public key. Anyone can confirm that a VES is the encryption of an ordinary signature, but only the adjudicator can resume the ordinary signature from a VES. A secure VES should insure that the verifier obtains nothing except a valid VES. The adjudicator does not participate in the actual exchange protocol in normal cases, but is needed in case of dispute. The VES schemes are useful in fair exchange protocols [16, 17] and also in other cryptographic protocols [18-20].

The early VES scheme is based on zero-knowledge proofs, and which is inefficient. In 2003, Boneh et al. first proposed a VES scheme with bilinear pairings in [21], which is based on traditional PKC. Since then, several new constructions of VES scheme [22-25] have been proposed including PKC-VES, ID-based VES and CL-VES, but there is still few Certificate-based Verifiable Encrypted Signature (CVES) scheme, and few VES scheme without pairings. The bulk of the VES scheme is constructed with bilinear pairings. On the other hand, the relative computation cost of the bilinear pairing operation is regarded as costly operations compared with other operations, such as scalar multiplication and exponentiation etc. Further, most of PKC-VES scheme are generally considered to be costly to use and managed, and ID-based VES scheme has an inherent drawback of key escrow, as the private key generator holds any user’s private key, and the trust level of CL-VES is lower.

Recently, elliptic curve cryptography (ECC) [26, 27] has received increasing attention from researchers’ because of its high intensity security and efficient algorithm, and has been widely used in practical application in the information security, and a number of ECC relevant Patent were presented, including Patent US 7218735, titled “Cryptography Method on Elliptic Curves” [28], is a cryptography method for generating probabilistic digital signatures etc., Patent US 8117447, titled “Authentication method employing elliptic curve cryptography” [29], is an authentication method employing elliptic curve cryptography (ECC), and WIPO Patent Application WO/2010/146302, titled “Cryptography on an Elliptical Curve” [30], is an electronic component in which a cryptographic calculation on an Elliptical Curve is performed.

In this paper, we propose an efficient and secure CVES scheme based on the elliptic curve group by incorporating the verifiable encrypted signature into the CB-PKS. Our scheme is provable secure under the elliptic curve discrete logarithm problem over a finite field in the random oracles, and it does not require any pairing operations. In proposed CVES scheme, we set CA as an adjudicator to avoid a dishonest signer creating a CVES which can be verified but can not be resumed by the adjudicator using a replaced adjudicator’s public key. Compared with the other existing secure VES scheme, our scheme enjoys less running time, operation cost and communication cost, as well as simple use and easy management, and thus have the merits of efficiency in performance.

The rest of the paper is organized as follows: Section 2 gives the background concepts on elliptic curve group and some related mathematical problems which help realize our CVES scheme. Section 3 describes the formal definition of CVES. Section 4 proposes a new efficient CVES scheme without pairings. Section 5 gives a security proofs under the random oracle model, and Section 6 gives our analysis. Finally, we conclude in Section 7.

2. MATHEMATICAL PROBLEMS

In this section, we would like to review some related mathematical problems [26, 27].

Elliptic Curve Group: Assume $E$ denotes an elliptic curve and $G$ denotes an elliptic curve group, $F_q$ denotes a prime finite field, assume the order of group $G$ is $q$, we let $E/F_q$ be an elliptic curve $E$ over a prime finite field $F_q$, which is defined by an equation and a discriminant as follows:

$$ y^2 = x^3 + ax + b, a,b \in F_q $$

$$ \Delta = 4a^3 + 27b^2 \neq 0 $$

We can define the point addition and scalar multiplication as follows:

- **The point addition**: Let $P, Q \in G$, $l$ be a line containing $P$ and $Q$, if $P = Q$, then $l$ is a tangent line to $E/F_q$, let $R$, be a third point of intersection with $l$ and $E/F_q$. Then $P + Q$ is the point such that $l$ intersects $E/F_q$ at $R$ and $O$, namely $R = P + Q$.

- **The scalar multiplication**: The scalar multiplication over $E/F_q$ can be defined as follows:

$$ tP = P + P + ...... + P \ (t \ times) $$

**Complexity Assumptions**: Elliptic curve discrete logarithm problem (ECDLP), for $x \in \mathbb{Z}_p^*$, and $P \in G$ is a random generator of $G$. Given $Q = xP$ to compute $x$.

The ECDLP defined over $G$ is assumed to be intractable within polynomial time.

3. DEFINITION OF CVES

3.1. Formal Definition of CVES

There are three parties in a CVES scheme including a signer, a verifier and an adjudicator. We define a CVES as follows:
• **Setup** \((k)\): The algorithm takes a security parameter \(k\) as its input and returns the system parameters \(params\) and the system master-key \(msk\).

• **UserKeyGen** \((params,ID_a)\): The algorithm takes the system parameters \(params\) and a signer’s identity \(ID_a\) as its input, and returns the signer’s private/public key pair \((PK_a,SK_a)\).

• **CertGen** \((params,msk,ID_a,PK_a)\): The algorithm takes the system parameters \(params\), the system master-key \(msk\), a signer’s identity \(ID_a\) and his public key \(PK_a\) as its input, and returns a certificate \(Cert_a\) corresponding to the signer \(ID_a\).

• **Sign** \((m,ID_a,Cert_a,SK_a)\): The algorithm takes a message \(m\) to be signed, a signer’s identity \(ID_a\) and his private key \(SK_a\), certificate \(Cert_a\) as its input, and outputs an ordinary signature \(\sigma\) on the message \(m\).

• **Verify** \((params,m,\sigma,ID_a,Cert_a,PK_a)\): The algorithm takes the system parameters \(params\), a message/ordinary CBS pair \((m,\sigma)\), a signer’s identity \(ID_a\) and his public key \(PK_a\), certificate \(Cert_a\) as its input, and returns true or false.

• **CVES-Sign** \((params,m,ID_a,SK_a,Cert_a,PK_a)\): The algorithm takes the system parameters \(params\), a message \(m\), a signer’s identity \(ID_a\) and his private key \(SK_a\), certificate \(Cert_a\), an adjudication’s public key \(PK_r\) as its input, outputs a verifiable encrypted signature \(\delta\) on the message \(m\).

• **CVES-Verify** \((params,m,\delta,ID_a,PK_a,Cert_a,PK_r)\): The algorithm takes the system parameters \(params\), a message/SES pair \((m,\delta)\), a signer’s identity \(ID_a\) and his public key \(PK_a\), certificate \(Cert_a\), an adjudication’s public key \(PK_r\) as its input, and returns true or false.

• **Adjudication** \((params,m,\delta,ID_a,SK_a)\): The algorithm takes the system parameters \(params\), a message/CVES pair \((m,\delta)\), a signer’s identity \(ID_a\), an adjudication’s private key \(SK_r\) as its input, outputs an ordinary signature \(\sigma\) on message \(m\).

In the formal definition described as above, the algorithms **UserKeyGen**, **CertGen**, **Sign** and **Verify** are the same as those of ordinary CB-PKS schemes.

### 3.2. Security Model

This section proposes a security model for **CVES**. We are concerned with three different types of attacks including signer’s attack, verifier’s attack and adjudicator’s attack. We want our **CVES** scheme to be secure against each of these attacks.

As defined in ordinary CBS schemes, we should consider two types of adversary for a **CVES** scheme.

- **Type I Adversary** \(A_I\): The adversary \(A_I\) simulates an uncertified user who holds the private key of the user and \(A_I\) can substitute for any user’s public key with his own values, but \(A_I\) is not allowed to possess the system master-key. \(A_I\) cannot obtain the certificate of the false public key from the certifier if he has replaced the user \(ID\) ‘s public key.

- **Type II Adversary** \(A_{II}\): The adversary \(A_{II}\) simulates the malicious-but-passive \(CA\) which is allowed to possess the system master-key, but he is not able to substitute for any user’s public key, and he doesn’t know anything about the user’s private key.

A secure **CVES** scheme required three security properties including validity, unforgeability and opacity.

- **Validity**: The validity of a **CVES** scheme can be verified by anyone, and the adjudicator can resume the valid ordinary CBS from a given **CVES**. Validity of a **CVES** scheme requires that the CBS which is generated by **Sign** algorithm must be able to pass the **Verify** algorithm, the **CVES** which is generated by **CVES-Sign** algorithm must be able to pass **CVES-Verify** algorithm, and the ordinary signature which is resumed from a given **CVES** by the adjudicator also must be able to be verified as an ordinary CBS. Namely, following equations should be satisfied.

\[
\text{Verify}(\text{params}, m, \sigma, \text{Cert}_a, \text{PK}_a) = \text{true}
\]

\[
\text{CVES-Verify}(\text{params}, m, \delta, \text{ID}_a, \text{PK}_a, \text{Cert}_a, \text{PK}_r) = \text{true}
\]

\[
\text{Verify}(\text{params}, m, \text{Adju}(\text{params}, m, \delta, \text{ID}_a, \text{SK}_r), \text{Cert}_a, \text{PK}_a) = \text{true}
\]

where **Verify**, **CVES-Verify** and **Adju** are the algorithms **Verify**, **CVES-Verify** and **Adjudication** in our **CVES** scheme, respectively.

- **Unforgeability**: The unforgeability requires that it is hard to forge a valid **CVES** by a malicious adversary. The unforgeability in our **CVES** be considered against both types of adversary \(A_I\) and \(A_{II}\). \(A_I\) may request query oracles for **UserKeyGen**, **Hash**, **CertGen**, **Corruption**, **ReplacePublicKey**, **CVES-Sign** and **Adjudication**. \(A_{II}\) may request query oracles for **UserKeyGen**, **Hash**, **Corruption**, **CVES-Sign** and **Adjudication**.

- **Opacity**: The opacity requires that it is hard to extract an ordinary CBS from a given **CVES** by a malicious adversary. In our **CVES** scheme, because \(CA\) is an adjudicator, it is trusted not to break the opacity, so the opacity in our **CVES** secure model can be considered against adversary \(A_I\) only. Adversary \(A_I\) may request query oracles for **UserKeyGen**, **Hash**, **Corruption**, Public key replacement and **Adjudication**.

### 4. PROPOSED CVES SCHEME

We propose an efficient **CVES** scheme in this section. Our **CVES** scheme consists of the following eight algo-
rithms, and we set $CA$ as an adjudicator in our CVES scheme.

- **Setup**: Sets $E/F_q$ to be an elliptic curve, $E$ over a prime finite field, $F_q$ as defined in Section 2. We assume that $k$ be a system security parameter, and the algorithm randomly selects $s_i \in \mathbb{Z}_q^*$ as the system master secret key $msk$, computes the system master public key $mpk = s_i P$, and selects two cryptographic hash functions: $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$. Then the system public parameters are:

$$params = (F_q, E/F_q, G, P, q, mpk, H_1, H_2)$$

- **UserKeyGen**: Given the system public parameters $params$, the signer $ID_A$ randomly selects $s_A \in \mathbb{Z}_q^*$, computes $SK_A = s_A$ and $PK_A = s_A P$. The algorithm outputs $(SK_A, PK_A)$ as $ID_A$’s key pair.

- **CertGen**: Given the system public parameters $params$ and master secret key $msk$, a signer’s identity $ID_A$ and his public key $PK_A$, $CA$ computes $Q_A = H_1(ID_A \| PK_A \| mpk)$, and outputs a certificate $Cert_A = s_A Q_A \mod q$ to signer. The signer verifies whether $Cert_A P = Q_A mpk$ holds with equality.

- **Sign**: Given the system public parameters $params$, a message $m$, a signer’s identity $ID_A$, and his private key, certificate $Cert_A$. The signer performs as follows:

a) Computes $S_A = Cert_A + s_A$ as his temporary signing key;

b) Picks $r \in \mathbb{Z}_q^*$ at random and computes $U = rP$;

c) Computes $h = H_2(m, U)$, $V = h S_A + r$.

Outputs an ordinary CB-PKS $\sigma = (V, U)$.

- **Verify**: Given the system public parameters $params$, an ordinary CBS $\sigma = (V, U)$ for the identity $ID_A$ on the message $m$, the verifier performs as follows:

a) Computes $h = H_2(m, U)$, $Q_A = H_1(ID_A \| PK_A \| mpk)$;

b) Verifies $VP = (Q_A mpk + PK_A)_h + U$, if the equation holds, the CB-PKS $\sigma = (V, U)$ is valid.

- **CVES-Sign**: Given the system parameters $params$, a message $m$, a signer’s identity $ID_A$, his private key $SK_A$ and certificate $Cert_A$. The signer works as follows:

a) Computes $S_A = Cert_A + s_A$ as his temporary signing key;

b) Picks $r \in \mathbb{Z}_q^*$ at random and computes $U = rP$;

c) Computes $h = H_2(m, U)$, $V = h S_A + r$;

d) Computes $W = V + rhmpk$.

Outputs a CVES $\delta = (W, U)$.

- **CVES-Verify**: Given the system parameters $params$, a message/CVES pair $(m, \delta)$, a signer’s identity $ID_A$ and his public key $PK_A$, an adjudicator’s public key $mpk$. The verifier works as follows:

a) Computes $h = H_2(m, U)$,

$b) Verifies whether the following equation holds, if so, the CVES $\delta$ is valid.

$$WP = (Q_A mpk + PK_A + Umhk)h + U$$

- **Adjudication**: Given the system public parameters $params$, a signer’s identity $ID_A$, a message/CVES pair $(m, \delta)$, and an adjudicator’s private key $msk$. The adjudicator computes $V = W - hs_A U$, and outputs an ordinary CBS $\sigma = (V, U)$ for the message $m$.

5. SECURITY ANALYSIS

5.1. Validity

**Theorem 1.** The proposed CVES scheme is valid.

**Proof:** We shall demonstrate the validity of our CVES scheme with three aspects as follows:

- If $\delta = (W, U)$ is a valid CVES, it should meet the verification equation of the CVES-Verify algorithm. The verification is as follows:

$$WP = (V + rhmpk)P$$

$$= (h S_A + r) P + humpk$$

$$= (Q_A mpk + PK_A + Umhk)h + U$$

- If $\sigma = (V, U)$ is a valid ordinary CBS, then it must meet the verification equation of the Verify algorithm. The verification is as follows:

$$WP = (Q_A mpk + PK_A)_h + U$$

$$= (Q_A s_A P + s_A P) h + rP$$

$$= (S_A h + r) P$$

$$= VP$$

- If $\delta = (W, U)$ is a valid CVES, then the adjudicator can resume the ordinary signature $\sigma = (V, U)$ from a given CVES $\delta = (W, U)$ with adjudicator’s private key $s_k$, and the verification is as follows: because $\delta = (W, U)$ is
a valid CVES, then the adjudicator can compute \( V = W - hS_U \) from \( S = (W, U) \), and the \((V, U)\) meets the original verification equation of the Verify algorithm, the verification is as follows:

\[
V P = (W - hS_U) P \\
= W P - hU mpk \\
= (Q, mpk + PK_P) h + U
\]

Combining the above analysis, we can get that our CVES scheme meets the validity.

### 5.2. Unforgeability

**Lemma 1.** Our CVES scheme is existential unforgeable against the Type I adversary \( A_i \) under the hardness of ECDLP in polynomial time.

**Proof.** We denote by \( A_i \) a type I Adversary who could attack our CVES scheme with non-negligible advantage, then, the adjudicator would successfully construct an algorithm \( B \) by interacting with the adversary \( A_i \) to solve the elliptic curve discrete logarithm problem. Let, \( P \) be a generator of a multiplicative group \( G \), whose order is a prime \( q \).

Algorithm \( B \) is given a group element \( Q \in G \). Its goal is to find \( x \in \mathbb{Z}_q^* \) such that \( Q = xP \).

1. **Setup:** The algorithm \( B \) sets \( mpk = Q \), \( PID_i = ID_j \| PK_{ID_j} \| mpk \), and the hash functions \( H_1 \) and \( H_2 \) are considered as random oracles, and algorithm \( B \) maintains four lists, those are \( UK - List\left(ID_j, PK_{ID_j}, SK_{ID_j}\right) \), \( H_1 - List\left(PID_i, q_i\right) \), \( H_2 - List\left((m_j, U_j), h_j\right) \) and \( Cert - List\left(ID_j, Cert_{ID_j}\right) \), which are empty at first. The system parameters are:

   \[
   params = (F_q, E / F_q, G, P, q, mpk, H_1, H_2)
   \]

2. **Queries:** The adversary \( A_i \) can issue additional queries to random oracles as follows.

   - **UserKeyGenQueries:** On input a new query \((ID_j, P)\), \( B \) first scans \( UK - List \) to check whether \( UK - List \) contained \((ID_j, *, *, *)\), if so, \( B \) returns \((PK_{ID_j}, SK_{ID_j})\) to \( A_i \), and \( ID_j \) is said to be created. Otherwise, \( B \) picks \( x_{ID_j} \in \mathbb{Z}_q^* \) at random and sets \( PK_{ID_j} = x_{ID_j}P \), \( SK_{ID_j} = x_{ID_j} \), returns \((PK_{ID_j}, SK_{ID_j})\) to \( A_i \) and adds \((ID_j, PK_{ID_j}, SK_{ID_j})\) into \( UK - List \).

   - **H_1Queries:** On input a new query \((PID_i, q_i)\), \( B \) first scans \( H_1 - List \) to check whether \( H_1 - List \) contained \((PID_i, *, *, *)\). If so, \( q_i \) is returned, otherwise \( B \) performs as follows:

     - If \( PID_i \neq PID_i' \), then \( B \) randomly picks \( q_i \in \mathbb{Z}_q^* \), sets \( H_1(PID_i) = q_i \).
     - If \( PID_i = PID_i' \), then \( B \) randomly picks \( \lambda \in \mathbb{Z}_q^* \), sets \( q_i = \lambda P \).

       In both cases, \( B \) sets \( H_1(PID_i) = q_i \). Finally, \( B \) returns \( H_1(PID_i) \) to \( A_i \), then adds an element \((PID_i, q_i)\) to \( H_1 - List \).

   - **H_2Queries:** On input a new query \((m_j, U_j)\), \( B \) first scans \( H_2 - List \) to check whether \( H_2 - List \) contained \((m_j, U_j, *)\). If so, \( h_j \) is returned. Otherwise, \( B \) picks \( \zeta_j \in \mathbb{Z}_q^* \), and sets \( h_j = \zeta_j P \), \( H_2(m_j, U_j) = h_j \), returns \( H_2(m_j, U_j) \) to \( A_i \) and adds \((m_j, U_j, h_j)\) into \( H_2 - List \).

   - **CertGenQueries:** On input a new query \((ID_j, P)\), \( B \) first scans \( Cert - List \) to check whether \( Cert - List \) contained \((ID_j, *, *)\). If so, \( Cert_{ID_j} \) is returned. Otherwise, \( B \) performs as follows:

     - If \( PID_i \neq PID_i' \), then \( B \) sets \( Cert_{ID_j} = \lambda Q \), returns \( Cert_{ID_j} \) to \( A_i \) and adds \((ID_i, Cert_{ID_j})\) into \( Cert - List \).
     - If \( PID_i = PID_i' \), then \( B \) output “failure” and halts.

   - **ReplacePublicKeyQueries:** On inputting a new query \((ID_j, PK'_{ID_j})\), \( B \) first scans \( UK - List \) to check whether \( UK - List \) contained an item \((ID_j, *, *, *)\). If so, \( B \) sets \( PK_{ID_j} = PK'_{ID_j} \), \( SK_{ID_j} = SK'_{ID_j} \) and saves \((ID_j, PK_{ID_j}, SK_{ID_j})\) to \( UK - List \), otherwise \( B \) adds an element \((ID_j, PK'_{ID_j}, SK'_{ID_j})\) to \( UK - List \), where, we assume that \( B \) can communicate with \( A_i \) to get private key \( SK_{ID_j} \) corresponding to \( PK_{ID_j} \).

   - **CVES-SignQueries:** On inputting a new query \((ID_j, m_j)\), \( B \) first scans \( UK - List \) to check whether \( ID_j \) has already been created. If not, then \( B \) issues a UserKeyGenQuery to obtain \((ID_j, PK_{ID_j}, SK_{ID_j})\), otherwise \( B \) checks \( H_1 - List \) to obtain \((PID_i, q_i)\), then randomly chooses \( r_j \in \mathbb{Z}_q^* \), computes \( U_j = r_j P \) and makes a \( H_2Query \) to obtain \((m_j, U_j, h_j)\). By assumption, \((PID_i, q_i)\) has been in \( H_1 - List \). Then, \( B \) computes \( W_j = h_j (\lambda, mpk + x_{ID_j} + r_j, mpk) + r_j \). Returns \( \delta_j = (W_j, U_j) \) as a CVES on \( m_j \). We can easily verify that \( \delta_j = (W_j, U_j) \) is a valid CVES with the following:
W_j P

= (h_j(\lambda, mpk + x_{id}, + r_j mpk) + r_j)P
= h_j(Q_{id} mpk + PK_{id} + U_{mpk}) + U_j

- AdjudicationQueries: On input a new adjudication query for CVES \( \delta_j = (W_j, U_j) \) on \((ID_j, m_j)\), B first checks whether \( \delta_j = (W_j, U_j) \) is valid, then computes \( V_j = W_j - \zeta_j U_{mpk} \) and returns \( \sigma_j = (V_j, U_j) \) as an ordinary CBS on \((ID_j, m_j)\).

- Setup: The algorithm B picks \( s_x \in \mathbb{Z}_p^* \) at random as the system master secret key, sets \( mpk = s_x P \), \( PID_i = ID_i \parallel PK_{id} \parallel mpk \), and we regard the hash functions \( H_1, H_2 \) as the random oracles. Returns the system public parameters \( params = (F_q, E(F_q), G, P, q, mpk, H_1, H_2) \).

2 Queries: The adversary \( A_U \) which can submit additional \( q_h \) queries to random oracles. The algorithm B maintains three lists, those are \( UK \rightarrow List(\{ID, PK_{id}, SK_{id}\}) \), \( H_1 \rightarrow List(\{PID_i, q_i\}) \) and \( H_2 \rightarrow List(\{(m, U_i), h_j\}) \), which are empty at first.

- UserKeyGenQueries: On input a new query \( ID_j \), B first scans \( UK \rightarrow List \) to check whether \( UK \rightarrow List \) contained \((ID_j,*)*\), if so, B returns \( (PK_{id}, SK_{id}) \) to \( A_U \), and \( ID_j \) is said to be created. Otherwise, B picks \( x_{id} \in \mathbb{Z}_q^* \) at random, and performs as follows:

- If \( PID_i \neq PID^* \), then B sets \( SK_{id} = x_{id}, PK_{id} = x_{id} P \);

- If \( PID_i = PID^* \), then B sets \( SK_{id} = x_{id}, PK_{id} = Q \).

In both case, B returns \( PK_{id} \) to \( A_U \) and adds \((ID_j, SK_{id}, PK_{id})\) into \( UK \rightarrow List \).

- H1Queries: On input a new query \( PID_i \), B first scans \( H_1 \rightarrow List \) to check whether \( H_1 \rightarrow List \) contained \((PID_i,*)*\). If so, \( q_i \) is returned, otherwise B picks \( q_i \in \mathbb{Z}_q^* \) at random, and sets \( H_1(PID_i) = q_i \). B return \( H_1(PID_i) \rightarrow A_U \), then adds an element \((PID_i,q_i)\) to \( H_1 \rightarrow List \).

- H2Queries: On input a new query \((m, U_i)\), B first scans \( H_2 \rightarrow List \) to check whether \( H_2 \rightarrow List \) contained \(((m, U_i), h_j)*\). If so, \( h_j \) is returned. Otherwise, B picks \( \zeta_j \in \mathbb{Z}_q^* \) at random, and sets \( h_j = \zeta_j P \), and \( H_2(m, U_i) = h_j \), returns \( H_2(m, U_i) \rightarrow A_U \) and adds \(((m, U_i), h_j)\) into \( H_2 \rightarrow List \).

- CorruptionQueries: On input a new query \( ID_j \), B will check the \( UK \rightarrow List \) and returns \( SK_{id} \) to \( A_U \). If \( SK_{id} = \bot \), B fails to solve this problem.

CVES-Sign Queries: On inputs a new query \((ID_i, PK_{id}, m_j)\), B first scans \( UK \rightarrow List \) to check whether \( ID_j \) has already been created, if so, B checks \( H_1 \rightarrow List \) to obtain \((PID_i,q_i)\) and picks \( r_j \in \mathbb{Z}_q^* \) at random, computes \( U_j = r_j P \), then scans \( H_2 \rightarrow List \) to get \(((m, U_i), h_j)\), and computes:
\[ Cert_{id} = s_x H_1(PID_i) = s_x q_i \]
\( W_j = (q_\text{mpk} + PK_{\text{id}_j} + U_j \text{mpk})\xi_j + r_j \)

The CVES \( \delta_j = (W_j, U_j, h_j) \) is returned. The validity can be easily verified with the following:

\[
W_j P = (q_\text{mpk} + PK_{\text{id}_j} + U_j \text{mpk})\xi_j + r_j P = (Q_{\text{id}_j}\text{mpk} + PK_{\text{id}_j} + U_j \text{mpk})h_j + U_j
\]

- **Adjudication Queries**: On input a new adjudication query for CVES \( \delta_j = (W_j, U_j, h_j) \) on \((ID_j, m_j)\), B first checks whether \( \delta_j = (W_j, U_j, h_j) \) is valid, then computes \( V_j = W_j - \xi_j U_j \text{mpk} \) and returns \( \sigma = (V_j, U_j) \) as the ordinary CBS on \((ID_j, m_j)\).

(3) **Output**: At last, adversary \( A_{\text{ID}_j} \) outputs a valid signature forgery \( \delta^* = (W^*, U^*, h^*) \) for \( ID^* \) with public key \( PK_{\text{id}_j} \), and the following equation holds:

\[
W^* P = (Q_{\text{id}_j}\text{mpk} + PK_{\text{id}_j} + U^* \text{mpk})h^* + U^*
\]

Applying the forking technique, B can obtains another forged signatures \( \delta^*_j = (W^*_j, U^*_j, h^*_j) \) on the same message \( m^* \), and the following equation holds:

\[
W^*_j P = (Q_{\text{id}_j}\text{mpk} + PK_{\text{id}_j} + U^* \text{mpk})h^*_j + U^*
\]

From the above, B can get:

\[
Q = (W^* - W^*_j)(h^* - h^*_j)^{-1} - q^* s_e - s_e U^*) P
\]

Thereby, B has successfully computed \( x \) as the solution of the elliptic curve discrete logarithm problem. Hence, we obtain the contradiction.

**Theorem 2**. The proposed CVES scheme is existentially unforgeable under adaptively chosen message attacks and the hardness of the elliptic curve discrete logarithm problem.

**Proof.** It is available from Lemma 1 and Lemma 2 easily.

### 5.3. Opacity

**Theorem 3**. The proposed CVES scheme meets the opacity.

**Proof.** Sets \( A_{\text{ID}_j} \) denotes a malicious adversary who could extract an ordinary signature \( \sigma = (V, U) \) from a given message/CVES pair \((m, \delta)\) with the non-negligible probability, where, \( \delta = (W, U) \), then the adjudicator would successfully construct an algorithm \( B \) by interacting with the adversary \( A_{\text{ID}_j} \) to solve the elliptic curve discrete logarithm problem. Let \( P \) be the generator of a multiplicative group \( G \), whose order is a prime \( q \). Algorithm \( B \) is given a group element \( Q \in G \). Its goal is to find \( x \in Z_q^* \) such that, \( Q = xP \).

(1) **Setup**: Algorithm \( B \) sets \( mpk = Q \), and we regard the hash functions \( H_1, H_2 \) as the random oracles. The system parameters are:

\[
\text{params} = (F_q, E / F_q, G, P, mpk, H_1, H_2)
\]

(2) **Queries**: The adversary \( A_{\text{ID}_j} \) which can submit additional \( q_b \) queries to random oracles. Algorithm \( B \) sets \( PID_{\text{ID}_j} = ID_{\text{ID}_j} || PK_{\text{id}_j} || mpk \), and maintains three lists, those are \( UK - \text{List}(ID_{\text{ID}_j}, PK_{\text{id}_j}, SK_{\text{id}_j}) \), \( H_1 - \text{List}(PID_{\text{ID}_j}, q_i) \) and \( H_2 - \text{List}((m_i, U_i), h_i) \), which are empty at first.

- **User KeyGen Queries**: On input a new query \( ID_{\text{ID}_j}, B \) first scans \( UK - \text{List} \) to check whether \( UK - \text{List} \) contains \((ID_{\text{ID}_j}, *, *)\), if so, \( B \) returns \((PK_{\text{id}_j}, SK_{\text{id}_j})\) to \( A_{\text{ID}_j} \), and \( ID_{\text{ID}_j} \) is said to be created. Otherwise, \( B \) picks \( \gamma_{\text{id}_j} \in r Z_q^* \) at random, and sets \( SK_{\text{id}_j} = x_{\text{id}_j} \), \( PK_{\text{id}_j} = x_{\text{id}_j} P \). \( B \) returns \( PK_{\text{id}_j} \) to \( A_{\text{ID}_j} \) and adds \((ID_{\text{ID}_j}, SK_{\text{id}_j}, PK_{\text{id}_j})\) into \( UK - \text{List} \).

- **H1 Queries**: On input a new query \( PID_{\text{ID}_j} \), \( B \) first scans \( H_1 - \text{List} \) to check whether \( H_1 - \text{List} \) contains \((PID_{\text{ID}_j}, q_i)\). If so, \( q_i \) is returned. Otherwise, \( B \) picks \( q_i \in Z_q^* \) at random, and sets \( H_1(PID_{\text{ID}_j}) = q_i \), returns \( H_1(PID_{\text{ID}_j}) \) to \( A_{\text{ID}_j} \) and adds \((PID_{\text{ID}_j}, q_i)\) into \( H_1 - \text{List} \).

- **H2 Queries**: On input a new query \((m_j, U_j)\), \( B \) first scans \( H_2 - \text{List} \) to check whether \( H_2 - \text{List} \) contains \((m_j, U_j)\). If so, \( h_j \) is returned. Otherwise, \( B \) picks \( \xi_j \in Z_q^* \) at random, and sets \( h_j = \xi_j P \) and \( H_2(m_j, U_j) = h_j \), returns \( H_2(m_j, U_j) \) to \( A_{\text{ID}_j} \) and adds \((m_j, U_j, h_j)\) into \( H_2 - \text{List} \).

- **Corruption Queries**: On input a new query \( ID_{\text{ID}_j} \), \( B \) will check \( UK - \text{List} \) and returns \( SK_{\text{id}_j} \) to \( A_{\text{ID}_j} \). If \( SK_{\text{id}_j} = \perp \), \( B \) fails to solve this problem.

- **Replace PublicKey Queries**: On inputting new queries \((ID_{\text{ID}_j}, PK_{\text{id}_j}^{'}), B \) first scans \( UK - \text{List} \) to check whether \( UK - \text{List} \) contained an item \((ID_{\text{ID}_j}, *, *)\). If so, \( B \) sets \( PK_{\text{id}_j}^{'} = PK_{\text{id}_j}^{'} \), \( SK_{\text{id}_j}^{'} = SK_{\text{id}_j}^{'} \) and saved \((ID_{\text{ID}_j}, PK_{\text{id}_j}^{'}\), \( SK_{\text{id}_j}^{'} \)) to \( UK - \text{List} \), otherwise \( B \) adds an element \((ID_{\text{ID}_j}, PK_{\text{id}_j}^{'}\), \( SK_{\text{id}_j}^{'} \)) to \( UK - \text{List} \), where we assume that \( B \) can communicate with \( A_{\text{ID}_j} \) to get private key \( SK_{\text{id}_j}^{'} \) corresponding to \( PK_{\text{id}_j}^{'} \).
The elliptic curve discrete logarithm problem and the random oracle model from the above theorem 1, theorem 2 and theorem 3.

6. EFFICIENCY ANALYSIS AND COMPARISON

We analyze and compare the performance of our CVES scheme with other existing VES schemes [22-25], and summarize the result in Table 1 as below. We denote $P$ the pairing operation, $M$ the scalar multiplication, $E$ the modular exponentiation, $SM$ the simultaneous scalar multiplication.

As shown in the above Table 1, the existing VES scheme [22-24] all require pairing operations in the algorithm $\text{Sign}$, $\text{Verify}$, $\text{VES-Sign}$, $\text{VES-Verify}$ and $\text{Adjudication}$, scheme [25] does not require any pairing operation in the algorithm $\text{Sign}$, $\text{VES-Sign}$ and $\text{Adjudication}$, but both requires pairing operations in the algorithm $\text{Verify}$ and $\text{VES-Verify}$, while our scheme does not require any pairing operation throughout the protocol. Therefore there are more advantages in our scheme with less running time and operation cost. In addition, the proposed VES scheme simplifies the certificate management process and solves the private key escrow problem. Thus, our scheme has high performance than the other existing VES schemes.

7. CONCLUSION

The paper proposes an efficient certificate-based verifiable encrypted signature scheme based on the elliptic curve group. The scheme does not have the key escrow problem and simplifies management process of the certificate, and does not use any bilinear pairing operations. Security analysis shows that it meets security properties including validity, opacity, and unforgeability under the elliptic curve discrete logarithm problem over a finite field. The proposed CVES scheme, due to ability to have no pairings, is more efficient than the other previous VES schemes.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

The authors thank the anonymous referees and reviewers for their very valuable comments on this paper. This work is supported by Natural Science Foundation of China (No.
An Efficient Certificate-based Verifiable Encrypted Signature Scheme


REFERENCES