Verifiably Encrypted Group Signatures

Zhen Wang[1,2], Xiling Luo[1,2], and Qianhong Wu[1(✉)]

[1] School of Electronic and Information Engineering,
Beihang University, Beijing, China
qianhong.wu@buaa.edu.cn
[2] Beijing Key Laboratory for Network-Based Cooperative Air Traffic Management,
Beijing, China

Abstract. Recently, verifiably encrypted signatures (VESs) have been widely used in fair exchange, however most of them do not provide a method to protect the anonymity of the signer, leading to privacy leakage in fair exchange. Verifiably Encrypted Group Signature (VEGS) overcomes drawbacks of VES, which allows a verifier to check its validity without decryption. And VEGS does not reveal the identity of the signer, thus protecting the privacy of the signer. In VEGS systems, a signer generates a group signature with his private key, then encrypts it with the adjudicator's public key and outputs a VEGS. A verifier can check whether a VEGS is valid. The group manager reveals the identity of the VEGS if necessary. The adjudicator can extract the original group signature from the VEGS with his private key. In this paper, we propose the first concrete VEGS scheme according to our model. We define several security properties which are essential to VEGS schemes and we prove that our scheme is secure in the standard model. Additionally, we discuss some relevant issues about our scheme.

Keywords: Verifiably Encrypted Group Signature · Verifiably Encrypted Signature · Group signature · Security properties

1 Introduction

With the development of the Internet, fair exchange has been applied to online transaction. In fair exchange, two involved parties exchange goods with each other fairly. However, most existing protocols can not protect the privacy of the exchange parties. Suppose one person wants to exchange a file with company B on behalf of company A, however, he may not want to expose his identity. This leads to a big challenge to achieve fair exchange protocols since most of them use verifiably encrypted signature (VES), which exposes the identities of the two parties in the transaction.

VES is an encrypted signature and its validity can be checked without decryption. As stated in [17,20], a VES scheme consists of a signature scheme and an encryption scheme. Boneh et al. first proposed a VES scheme [6], which is constructed by aggregate signatures. Lu et al. [17], Nishimaki and Xagawa [18] independently proposed their VES schemes, which are both secure in the standard

T. Okamoto et al. (Eds.): ProvSec 2017, LNCS 10592, pp. 107–126, 2017.
https://doi.org/10.1007/978-3-319-68637-0_7

model. However, the private key size is rather large. Besides, the scheme of Nishimaki and Xagawa is based on Waters' dual signature scheme, leading to a large size of the signature. Rückert and Schröder [20] proposed a VES scheme based on a short signature scheme [4], which is efficient due to the short verification key. However, most existing VES schemes are based on Public Key Infrastructure (PKI), leading to high cost in the authentication and management of the public keys. Using identity-based cryptosystems, the above problem can be solved. Gu et al. [16] proposed an identity-based VES scheme with random oracles. However, their scheme was proved to be insecure [21]. Then Zhang et al. [22] proposed an identity-based VES scheme in the standard model. However, their scheme is a weak version of identity-based VES [15]. Besides, all above schemes can not protect the anonymity of the signer, thus we have to use another technique called group signature.

Group signature was first introduced by Chaum and Heyst [11], which allows an authentic user generates a signature on behave of a group and hides his identity from others. In their paper, they gave the basic ideas about group signature and presented four group signature schemes. However, they did not give specific security definitions. Then several related works were presented [2,3,5,13]. However, all of them are too inefficient or provably secure with random oracles. Then Bellare et al. [9] first formalized the security definitions of the group signature and presented a group signature scheme which is secure in the standard model. Ateniese et al. [1] also proposed a group signature which is secure without random oracles. However, all above schemes use Zero-Knowledge (ZK) proof technique which is inefficient. Later, Boyen and Waters [10] constructed a group signature scheme without ZK proof technique and their scheme is provably secure in the standard model.

Motivated by above works, we first formalized a new concept called verifiably encrypted group signature (VEGS), which is derived from verifiably encrypted signature (VES) and group signature. As a consequence, VEGS has similar properties with both VES and group signature. VEGS can be checked without decryption and protect the signer's anonymity. Besides, if there exists dispute, a trusted parties can trace the identity of the signer. Thus VEGS can be used to construct fair exchange protocols which hide the identity of the parties in the transaction.

For example, if Alice wishes to exchange signature on a file with company B on behalf of company A and she does not want to expose her identity, she can use a VEGS to complete the exchange instead of an original signature. Alice first sends a VEGS to company B. Then a staff of company B (known as Bob) checks whether the VEGS is valid. If the VEGS is valid, Bob generates a group signature and sends it to company A. Then Alice checks whether the group signature is valid. If it is valid, Alice sends her group signature to company B. If Alice does not sends her group signature to B, B sends the VEGS together with Bob's group signature to the adjudicator. If both of them are valid, the adjudicator recovers the original group signature of Alice and returns it to company B. The exchange reveals nothing about identities of Alice and Bob due to the anonymity of the group signature and VEGS. If someone denies that he generates the VEGS or

group signature, the group manager can trace the identity of the signer. Besides, VEGS has useful applications such as online data exchange and online contact signing. And the special properties make it appealing to explore the potential in VEGS.

1.1 Our Contributions

We formalize a new concept of verifiably encrypted group signature (VEGS), which combines verifiably encrypted signature (VES) and group signature. VEGSs are encrypted group signatures which can be used to protect the anonymity of the signers. And VEGSs allow us to check their validity without decryption. In VEGS, the group master key and group tracing key are generated by the group master and the group manager keeps the group tracing key. A user generates a group signature with his private key, then encrypts it with the adjudicator's public key, and obtains a VEGS. A verifier checks whether the VEGS is valid. The group manager can open the VEGS and trace the identity of the signer if necessary. The adjudicator can extract the original group signature from the VEGS with his private key.

We define the security properties required in VEGS schemes, i.e., full-anonymity, full-traceability, unforgeability, opacity and extractability. Full-anonymity describes that no one can reveal the identity of the signer except the group manager. Full-traceability means that any valid VEGS can be traced to a valid identity by the group manager. Unforgeability guarantees that no one can forge a VEGS without a signing key. Opacity means that no one can extract a valid group signature from a VEGS without the adjudicator's private key. Extractability guarantees that if a VEGS is valid, then the original group signature can be extracted by the adjudicator.

We propose the first concrete VEGS scheme by employing Boyen-Waters group signature scheme [10] and the ElGamal encryption scheme [14]. Then we prove our VEGS scheme is secure in the standard model. Finally, we discuss the extensions of our VEGS scheme.

1.2 Outline

We organize the rest of the paper as follows. In Sect. 2 we give the relevant notions. In Sect. 3 we present definition of VEGS scheme and security definitions. In Sect. 4 we propose our concrete VEGS scheme, then we prove our scheme is secure in the standard model. In Sect. 5 we discuss the extensions of our VEGS scheme. Finally, we conclude in Sect. 6.

2 Preliminaries

In this section, we briefly review the bilinear maps and complexity assumptions that are essential in our construction.

2.1 Bilinear Maps

In our paper, we use composite order bilinear groups as stated in [7]. Let $\mathbb{G}$ and $\mathbb{G}_T$ be finite cyclic groups of order n, g be a generator of $\mathbb{G}$, and $n = pq$ has two large prime factors (p and q). A map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ can be called an efficient bilinear map if it satisfies the following properties:

- *Bilinear*: For $\forall a, b \in \mathbb{Z}_n$, we have $e(g^a, g^b) = e(g, g)^{ab}$. Clearly, the bilinearity implies that for $\forall g_1, g_2, g_3 \in \mathbb{G}$, we have $e(g_1, g_3)e(g_2, g_3) = e(g_1 g_2, g_3)$.
- *Non-degeneracy*: $e(g, g) \neq 1$. In other words, the element $e(g, g)$ is a generator of $\mathbb{G}_T$.
- e is efficiently computable.

2.2 Complexity Assumptions

The security of our VEGS scheme is based on subgroup decision assumption, CDH assumption and aggregate extraction assumption. The subgroup decision assumption is based on the hardness of factoring [7], and aggregate extraction assumption is a variant of CDH assumption, thus all assumptions employed in our scheme are basic assumptions. We briefly review them below.

Subgroup Decision problem: Let $\mathbb{G}$ and $\mathbb{G}_T$ be finite cyclic groups of order $n = pq$, $\mathbb{G}_p$ and $\mathbb{G}_q$ be subgroups of $\mathbb{G}$ of order p and q, e be a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Choose $w \in \mathbb{G}$ randomly, decide whether $w \in \mathbb{G}_q$.

The subgroup decision assumption is as follows.

Definition 1. *The (t, ϵ)-subgroup decision assumption holds if no adversary runs at most t time and has at least ϵ advantage in solving the subgroup decision problem.*

CDH problem: Given g, g^a, g^b, compute g^{ab}.

If the probability that adversary $\mathcal{B}$ solves the CDH problem is at least ϵ, then we have

$$Pr[\mathcal{B}(g, g^a, g^b) = g^{ab}] \geq \epsilon,$$

Then CDH assumption is as follows.

Definition 2. *The (t, ϵ)-CDH assumption holds if no adversary runs at most t time and has at least ϵ advantage in solving the CDH problem on $\mathbb{G}$.*

The aggregate extraction problem: Given $\mathbb{G}$, $\mathbb{G}_p$, $\mathbb{G}_q$, $n = pq$, p, q, g, g^a, g^b, g^δ, g^ζ and $g^{ab+\delta\zeta}$, compute g^{ab}.

If the probability that adversary $\mathcal{B}$ solves the aggregate extraction problem is at least ϵ, then we have

$$Pr[\mathcal{B}(g, g^a, g^b, g^\delta, g^\zeta, g^{ab+\delta\zeta}) = g^{ab}] \geq \epsilon,$$

Then aggregate extraction assumption is as follows.

Definition 3. *The (t, ϵ)-aggregate extraction assumption holds if no adversary runs at most t time and has at least ϵ advantage in solving the aggregate extraction problem on $\mathbb{G}$.*

3 Modelling VEGS

3.1 Definition of VEGS Scheme

VEGS works as follows. A group master sets up the system and distributes the keys of users. A group manager keeps the group tracing key, which can be used to reveal a user's identity from the VEGS. Group members first register in the system with their identities and obtain their signing keys. Then they generate group signatures with the signing keys, encrypt them with the adjudicator's public key and finally obtain VEGSs. A verifier can check whether the VEGS is valid without decrypting it. The adjudicator can reveal the original group signature from the VEGS with his private key.

A VEGS scheme consists of following algorithms: Setup, AKG, Enroll, Sign, Verify, VESign, VEVerify, Open, Adj.

Setup: Setup takes as input security parameter 1^λ, and outputs public parameters $\mathtt{param}$ for verification, a master key MK for enrollment of users, and a tracing key TK for revealing the identity from the VEGS.

AKG: AKG takes as input security parameter 1^λ, and outputs a pair of keys (SK_T, PK_T) for the adjudicator.

Enroll: Enroll takes as input a user's identity $\mathtt{u}$, and the master key MK, outputs signing key $sk_\mathtt{u}$ for a group member.

Sign: Sign takes as input a message $\mathtt{m}$, the signing key $sk_\mathtt{u}$, and outputs a group signature σ.

Verify: Verify takes as input a message $\mathtt{m}$, a group signature σ and the public parameters $\mathtt{param}$, outputs a bit $b \in \{0,1\}$. If $b = 0$, the group signature is invalid. Otherwise, it is valid.

VESign: VESign takes as input a message $\mathtt{m}$, a signing key $sk_\mathtt{u}$ and the adjudicator's public key PK_T, outputs a VEGS ω.

VEVerify: VEVerify takes as input a message $\mathtt{m}$, a VEGS ω, and public parameters $\mathtt{param}$, outputs a bit $b \in \{0,1\}$. If $b = 0$, the VEGS is invalid. Otherwise, it is valid.

Open: Open takes as input the tracing key TK, a VEGS ω, and outputs the identity $\mathtt{u}$ of the signer.

Adj: Adj takes as input a VEGS ω, the adjudicator's private key SK_T, output the original group signature σ.

A VEGS scheme $\mathsf{VEGS} = (\mathsf{Setup}, \mathsf{AKG}, \mathsf{Enroll}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{VESign}, \mathsf{VEVerify}, \mathsf{Open}, \mathsf{Adj})$ is correct if for all $(\mathtt{param}, \mathsf{MK}, \mathsf{TK}) \leftarrow \mathsf{Setup}(1^\lambda)$, $(SK, PK) \leftarrow \mathsf{AKG}(1^\lambda)$, $\mathtt{u}$, $sk_\mathtt{u} \leftarrow \mathsf{Enroll}(\mathsf{MK}, \mathtt{u})$, $\mathtt{m}$, and $\omega \leftarrow \mathsf{VESign}(\mathtt{m}, sk_\mathtt{u}, PK_T)$, it always holds that $\mathsf{VEVerify}(\mathtt{m}, \mathsf{VESign}(\mathtt{m}, sk_\mathtt{u}, PK_T), PK_T, \mathtt{param}) = 1$ and $\mathsf{Verify}(\mathtt{m}, \mathsf{Adj}(\mathsf{VESign}(\mathtt{m}, sk_\mathtt{u}, PK_T), SK_T), \mathtt{param}, \mathtt{u}) = 1$.

3.2 Security Definitions

Security is significant for VEGS schemes. Informally, a VEGS scheme is secure if it satisfies the following properties, i.e., anonymity, traceability, unforgeability, opacity and extractability. Briefly, anonymity means that given a valid VEGS,

no one can extract the identity of the signer except the group manager who keeps the group tracing key. And traceability describes the property that the group manager can open any valid VEGS and reveal the identity of the signer. In our paper, we give stronger notions about anonymity and traceability called full-anonymity and full-traceability [9]. We define the new properties under stronger attack, which means that the adversary has the access to the private key oracle, the group signing oracle and VESign oracle. And for the attack of the full-traceability, we can even give the tracing key to the adversary. Unforgeability describes the property that no one can forge a VEGS without a signing key. And opacity means that no one can extract a valid group signature from a VEGS without the adjudicator's private key. Finally, extractability is also a necessary property and it guarantees that the valid group signature can be extracted from the valid VEGS. Formally, we define these properties by the following games.

Definition 4. *Full-anonymity is defined by the game* $\mathsf{Game}_{Anoy}(\lambda)$*. The involved parties in the game are a challenger and an adversary* $\mathcal{A}$*.*

- **Setup.** The challenger sets up the system, generates the system parameters and sends the public parameters to $\mathcal{A}$.
- **Query.** $\mathcal{A}$ submits an identity $\mathfrak{u}$ to the challenger and asks for a private key, the challenger runs Enroll and returns the signing key $sk_{\mathfrak{u}}$ to $\mathcal{A}$. $\mathcal{A}$ can query at most q_1 times for signing keys. $\mathcal{A}$ submits an identity $\mathfrak{u}$, a message $\mathfrak{m}$ to the challenger and asks for a group signature or VEGS, the challenger runs Sign or VESign and returns a group signature σ or a VEGS ω. $\mathcal{A}$ can query at most q_2 times for group signatures and q_3 times for VEGSs. If $\mathcal{A}$ submits a message $\mathfrak{m}$, a VEGS ω to the challenger, and asks for arbitration, the challenger first checks whether ω is valid, if it is not, then the challenger returns $\perp$. Otherwise, the challenger runs Adj and returns a group signature σ. $\mathcal{A}$ can query at most q_4 times for adjudication.
- **Challenge.** $\mathcal{A}$ randomly chooses two identities $\mathfrak{u}_1$, $\mathfrak{u}_2$ which have the same length, a message $\mathfrak{m}^*$ and sends them to the challenger. The challenger random picks a bit $b \in \{0, 1\}$, generates a private key of $\mathfrak{u}_b$, and returns a VEGS $\omega_b \leftarrow \mathsf{VESign}(\mathfrak{m}^*, sk_{\mathfrak{u}_b}, PK_T)$ to $\mathcal{A}$.
- **Guess.** Finally, $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ as a guess of b.

Define the probability that $\mathcal{A}$ wins in the above game as

$$Adv_{\mathcal{A}}^{Anon} = |Pr[b' = b] - \frac{1}{2}|.$$

A VEGS scheme is fully-anonymous if for every probability polynomial time (PPT) adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the above game is negligible. In fact, since we use the "selective-identity,adaptive-message" attack [10] in the above game, we call it CPA (chosen-plaintext attack)-ID model.

Definition 5. *Full-traceability is defined by the game* $\mathsf{Game}_{Trac}(\lambda)$ *which is played by a challenger and an adversary* $\mathcal{A}$*.*

- **Setup.** The challenger sets up the system, generates the system parameters and sends the public parameters to $\mathcal{A}$. In this step, $\mathcal{A}$ can also get the group tracing key.
- **Query.** In this step, $\mathcal{A}$ does the same thing as he does in $\mathsf{Game}_{Anoy}(\lambda)$.
- **Forge.** Finally, $\mathcal{A}$ outputs a pair $(\mathtt{m}^*, \omega^*)$. The challenger first checks whether the VEGS is valid. If it is invalid, the challenger returns $\perp$. Otherwise, the challenger runs Open and obtains an identity $\mathtt{u}^*$. If $\mathtt{u}^* \in \mathcal{U}$ (we assume $\mathcal{U}$ is a set of all queried identities), then the challenger returns $\perp$. If $\mathtt{u}^* \notin \mathcal{U}$ and $\mathcal{A}$ has not queried a private key of identity $\mathtt{u}^*$, a group signature or VEGS on $(\mathtt{u}^*, \mathtt{m}^*)$, then $\mathcal{A}$ wins in the game.

A VEGS scheme is said to be fully-traceable if for every PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the above game is negligible.

One may find that our definition of full-traceability simply implies unforgeability, thus we do not give more details about unforgeability. And we deduce that a fully-traceable VEGS scheme must be unforgetable.

Definition 6. *Opacity is defined by the game* $\mathsf{Game}_{Opac}(\lambda)$ *which is played by a challenger and an adversary* $\mathcal{A}$*.*

- **Setup.** The challenger sets up the system, generates the system parameters and sends the public parameters to $\mathcal{A}$.
- **Query.** In this step, $\mathcal{A}$ does the same thing as he does in $\mathsf{Game}_{Anoy}(\lambda)$.
- **Forge.** Finally, $\mathcal{A}$ outputs a pair $(\mathtt{m}^*, \sigma^*)$. The challenger first checks whether the group signature σ^* is valid. If it is invalid, then the challenger returns $\perp$. If σ^* is valid and $\mathcal{A}$ has not queried a private key of identity $\mathtt{u}^*$, a group signature on $(\mathtt{u}^*, \mathtt{m}^*)$, then $\mathcal{A}$ wins in the game.

A VEGS scheme is said to be opaque if for every PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the above game is negligible.

Definition 7. *Extractability is defined by the game* $\mathsf{Game}_{Extr}(\lambda)$ *which is played by a challenger and an adversary* $\mathcal{A}$*.*

- **Setup.** The challenger sets up the system, generates the system parameters and sends the public parameters to $\mathcal{A}$.
- **Query.** In this step, $\mathcal{A}$ does the same thing as he does in $\mathsf{Game}_{Anoy}(\lambda)$.
- **Forge.** Finally, $\mathcal{A}$ submits a tuple $(\mathtt{m}^*, \omega^*, \mathtt{param}^*)$ to the challenger.
- **Extract.** The challenger runs Adj and gets a group signature σ^*. If $\mathsf{VEVerify}(\mathtt{m}^*, \omega^*, PK_T, \mathtt{param}^*) = 1$ and $\mathsf{Verify}(\mathtt{m}^*, \sigma^*, \mathtt{param}^*) = 0$, then $\mathcal{A}$ wins in the game.

A VEGS scheme is extractable if for every PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the above game is negligible.

4 VEGS Scheme

In this section, we present our VEGS scheme, which is based on Boyen-Waters group signature scheme [10] and ElGamal encryption scheme [14]. The VEGS scheme consists of following algorithms, Setup, Enroll, Sign, Verify, VESign, $\mathsf{VEVerify}$, Open, Adj.

4.1 Construction of VEGS Scheme

Setup: Take as input a security parameter 1^λ, and setup the system as follows. Let $\mathbb{G}$ and $\mathbb{G}_T$ be finite cyclic groups of order $n = pq$, $\mathbb{G}_p$ and $\mathbb{G}_q$ be subgroups of $\mathbb{G}$ of order p and q, e be a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Choose generators $g \in \mathbb{G}$ and $h \in \mathbb{G}_q$, a secret value $\alpha_1 \in \mathbb{Z}_n$ at random. Besides, choose random $g_2, u', u_1, ..., u_{n_u}, m', m_1, ..., m_{n_m} \in \mathbb{G}$, and set $g_1 = g^{\alpha_1}$, the master key $\mathsf{MK} = g_2^{\alpha_1}$, the group tracing key $\mathsf{TK} = q$. And the public parameters are $\mathtt{param} = (g, h, g_1, g_2, u', u_1, ...u_{n_u}, m', m_1, ..., m_{n_m})$.

AKG: Choose a secret value $\alpha_T \in \mathbb{Z}_n$, and set the adjudicator's keys as $(SK_T, PK_T) = (\alpha_T, g^{\alpha_T})$.

Enroll: Let $\mathfrak{u} = (k_1^u \cdots k_{n_u}^u)$ $(k_i^u \in \{0, 1\})$ be an identity of a group member, then his signing key is generated as follows. Choose $r_u \in \mathbb{Z}_n$ randomly, and compute,

$$sk_{\mathfrak{u}} = d_{\mathfrak{u}} = (d_1, d_2, d_3) = \left(g_2^{\alpha_1} \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u} \right)^{r_u}, g^{r_u}, h^{r_u} \right).$$

Sign: Suppose a user of identity $\mathfrak{u} = (k_1^u \cdots k_{n_u}^u)$ wishes to generate a group signature on message $\mathfrak{m} = (k_1^m \cdots k_{n_m}^m)$, then he does as follows. First, choose $r'_u, r_m, t_1, ..., t_{n_u} \in \mathbb{Z}_n$, and set $t = \sum_{i=1}^{n_u} t_i$. Then compute,

$$\begin{aligned}
\sigma_1 &= g_2^{\alpha_1} \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u} \right)^{r_u + r'_u} \left(m' \prod_{j=1}^{n_m} m_j^{k_j^m} \right)^{r_m} h^{(r_u + r'_u)t}, \\
\sigma_2 &= g^{r_u + r'_u}, \\
\sigma_3 &= g^{r_m}, \\
\sigma_4 &= h^t, \\
\sigma_5 &= \sigma_2^t = g^{(r_u + r'_u)t}, \\
c_i &= u_i^{k_i^u} \cdot h^{t_i}, \\
\pi_i &= (u_i^{2k_i^u - 1} \cdot h^{t_i})^{t_i}, \\
\sigma &= (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, c_1, ..., c_{n_u}, \pi_1, ..., \pi_{n_u}).
\end{aligned}$$

For simplicity, let $c = u' \prod_{i=1}^{n_u} c_i$ and $M = m' \prod_{j=1}^{n_m} m_j^{k_j^m}$, then we have $\sigma_1 = g_2^{\alpha_1} c^{r_u + r'_u} M^{r_m}$.

Verify: If a verifier wishes to check whether a group signature σ is valid, he first computes $c = u' \prod_{i=1}^{n_u} c_i$, then checks whether the following equations hold.

$$\forall i = 1, ..., k : e(c_i, u_i^{-1} c_i) \stackrel{?}{=} e(h, \pi_i).$$

If all of them hold, then check whether the following equations hold.

$$e(\sigma_1, g) \stackrel{?}{=} e(g_2, g_1) e(c, \sigma_2) e(M, \sigma_3).$$

$$e(\sigma_2, \sigma_4) \stackrel{?}{=} e(\sigma_5, h).$$

If the equations hold, then the group signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, c_1, ..., c_{n_u}, \pi_1, ..., \pi_{n_u})$ is valid.

VESign: To create a VEGS of identity $\mathfrak{u} = (k_1^u \cdots k_{n_u}^u)$ on message $\mathfrak{m} = (k_1^m \cdots k_{n_m}^m)$, the signer first generates a group signature σ, then chooses a random $s \in \mathbb{Z}_n$, and computes,

$$\omega_1 = (PK_T)^s \cdot \sigma_1 = (PK_T)^s g_2^{\alpha_1} \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u} \right)^{r_u + r'_u} \left(m' \prod_{j=1}^{n_m} m_j^{k_j^m} \right)^{r_m} h^{(r_u + r'_u)t},$$

$$\omega_2 = g^s,$$

$$\omega_3 = \sigma_2 = g^{r_u + r'_u},$$

$$\omega_4 = \sigma_3 = g^{r_m},$$

$$\omega_5 = \sigma_4 = h^t,$$

$$\omega_6 = \sigma_5 = g^{(r_u + r'_u)t},$$

$$\omega = (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, c_1, ..., c_{n_u}, \pi_1, ..., \pi_{n_u}).$$

In fact, we only encrypt σ_1, because the other part of the group signature is independent with the message $\mathfrak{m}$ and identity $\mathfrak{u}$.

VEVerify: To verify if a VEGS is valid, a verifier checks whether the following equations hold.

$$\forall i = 1, ..., k : e(c_i, u_i^{-1} c_i) \stackrel{?}{=} e(h, \pi_i),$$

$$e(\omega_1, g) \stackrel{?}{=} e(PK_T, \omega_2) e(g_2, g_1) e(c, \omega_3) e(M, \omega_4),$$

$$e(\omega_3, \omega_5) \stackrel{?}{=} e(\omega_6, h).$$

where $c = u' \prod_{i=1}^{n_u} c_i$ and $M = m' \prod_{j=1}^{n_m} m_j^{k_j^m}$. If all equations hold, then the VEGS is valid. Otherwise, it is invalid.

Open: The group manager recovers the signer's identity from the VEGS as follows if necessary. For each $i = 1, ..., n_u$, if $(c_i)^q = g^0$, the group manager sets $k_i^u = 0$. Otherwise, he sets $k_i^u = 1$. Finally, the group manager outputs the signer's identity, $\mathfrak{u} = (k_1^u \cdots k_{n_u}^u)$.

Adj: Take as input a VEGS ω, the adjudicator's private key SK_T, output the original group signature as follows.

$$\sigma_1 = \frac{\omega_1}{\omega_2^{\alpha_T}} = g_2^{\alpha_1} \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u} \right)^{r_u + r'_u} \left(m' \prod_{j=1}^{n_m} m_j^{k_j^m} \right)^{r_m} h^{(r_u + r'_u)t},$$

$$\sigma_2 = \omega_3 = g^{r_u + r'_u},$$

$$\sigma_3 = \omega_4 = g^{r_m},$$

$$\sigma_4 = \omega_5 = h^t,$$

$$\sigma_5 = \omega_6 = g^{(r_u + r'_u)t},$$

$$\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, c_1, ..., c_{n_u}, \pi_1, ..., \pi_{n_u}).$$

The correctness of our scheme is quite explicit and we will not prove it.

4.2 Security

Our VEGS scheme is secure in the standard model, which means that our scheme satisfies all properties described in Subsect. 3.2, we now prove it.

Theorem 1. *Our VEGS scheme is fully-anonymous (under CPA-ID attack) if the subgroup decision assumption holds.*

We do not prove it because the similar proof is given in [10].

Theorem 2. *Our VEGS scheme is fully-traceable if the underlying signature scheme is unforgeable.*

Proof. Suppose an adversary $\mathcal{A}$ breaks full-traceability of our VEGS scheme with advantage at least ϵ, then there exists an adversary $\mathcal{B}$ which can break the unforgeability of the underlying identity-based signature scheme [10] (also called two-level signature scheme) with the same advantage. $\mathcal{B}$ and $\mathcal{A}$ play the game $\mathsf{Game}_{Trac}(\lambda)$, $\mathcal{B}$ interacts with $\mathcal{A}$ and acts as a simulator. At the same time, $\mathcal{B}$ also plays a signature game called unforgeable game and tries to break the unforgeability of the underlying signature scheme. To complete the simulation, we assume that $\mathcal{B}$ plays the unforgeable game in $\mathbb{G}_p$, while he plays game $\mathsf{Game}_{Trac}(\lambda)$ in $\mathbb{G}$. We show how to construct $\mathcal{B}$.

– **Setup.** $\mathcal{B}$ gets the parameters of the signature scheme from his challenger, $\mathtt{param}_{\mathbb{G}_p} = (\tilde{g}, \tilde{g}_1 = \tilde{g}^{\alpha}, \tilde{g}_2, \tilde{u}', \tilde{u}_1, ..., \tilde{u}_{n_u}, \tilde{m}', \tilde{m}_1, ..., \tilde{m}_{n_m}) \in \mathbb{G}_p^{n_u + n_m + 3}$. Then $\mathcal{B}$ chooses $(\hat{g}, \hat{g}_1 = \hat{g}^{\beta}, \hat{g}_2, h, \hat{u}', \hat{u}_1, ..., \hat{u}_{n_u}, \hat{m}', \hat{m}_1, ..., \hat{m}_{n_m}) \in \mathbb{G}_q^{n_u + n_m + 4}$ randomly, and sets the public parameters as,

$$\mathtt{param}_{\mathbb{G}} = (g = \tilde{g}\hat{g}, g_1 = \tilde{g}_1\hat{g}_1, g_2 = \tilde{g}_2\hat{g}_2, h, u' = \tilde{u}'\hat{u}', u_1 = \tilde{u}_1\hat{u}_1, ..., u_{n_u} = \tilde{u}_{n_u}\hat{u}_{n_u},$$
$$m' = \tilde{m}'\hat{m}', m_1 = \tilde{m}_1\hat{m}_1, ..., m_{n_m} = \tilde{m}_{n_m}\hat{m}_{n_m}).$$

Besides, $\mathcal{B}$ chooses a random value $\alpha_T \in \mathbb{Z}_n$ and sets the adjudicator's private key as $(SK_T, PK_T) = (\alpha_T, g^{\alpha_T})$. Then $\mathcal{B}$ sends $\mathtt{param}_{\mathbb{G}}$, PK_T and the tracing key $\mathsf{TK} = q$ to $\mathcal{A}$. The parameters are distributed identically to what $\mathcal{A}$ expects.

– **Query.** In this step, $\mathcal{A}$ can make queries for private keys, group signatures and VEGSs. When $\mathcal{A}$ asks for a signing key of identity $\mathfrak{u} = (k_1^u \cdots k_{n_u}^u)$, $\mathcal{B}$ also asks his challenger for the user's (with the identity $\mathfrak{u}$) signing key. Then $\mathcal{B}$ receives the signing key of the underlying signature scheme, $\tilde{sk}_{\mathfrak{u}} = \tilde{d}_{\mathfrak{u}} = (\tilde{d}_1, \tilde{d}_2) = (\tilde{g}_2^{\alpha}(\tilde{u}' \prod_{i=1}^{n_u} \tilde{u}_i^{k_i^u})^{\tilde{r}_u}, \tilde{g}^{\tilde{r}_u})$. Then $\mathcal{B}$ chooses $\hat{r}_u \in \mathbb{Z}_q$ and computes,

$$sk_{\mathfrak{u}} = d_{\mathfrak{u}} = (d_1, d_2, d_3) = \left(\tilde{d}_1 \hat{g}_2^{\beta} \left(\hat{u}' \prod_{i=1}^{n_u} \hat{u}_i^{k_i^u} \right)^{\hat{r}_u}, \tilde{d}_2 \hat{g}^{\hat{r}_u}, h^{\hat{r}_u} \right).$$

It is obvious that the private keys generated by $\mathcal{B}$ have the same distribution with the real ones. If $\mathcal{A}$ asks for a group signature of identity $\mathfrak{u} = (k_1^u \cdots k_{n_u}^u)$ on message $\mathfrak{m} = (k_1^m \cdots k_{n_m}^m)$, $\mathcal{B}$ also submits the same identity $\mathfrak{u}$, the same message $\mathfrak{m}$ to his challenger and asks for an identity-based signature. Then $\mathcal{B}$ will obtain a signature,

$$\tilde{\sigma} = (\tilde{\sigma}_1, \tilde{\sigma}_2, \tilde{\sigma}_3) = \left(\tilde{g}_2^{\alpha} \left(\tilde{u}' \prod_{i=1}^{n_u} \tilde{u}_i^{k_i^u} \right)^{\tilde{r}_u + \tilde{r}'_u} \left(\tilde{m}' \prod_{j=1}^{n_m} \tilde{m}_j^{k_j^m} \right)^{\tilde{r}_m}, \tilde{g}^{\tilde{r}_u + \tilde{r}'_u}, \tilde{g}^{\tilde{r}_m} \right).$$

Next $\mathcal{B}$ chooses $t_1, ..., t_{n_u} \in \mathbb{Z}_n$, $r_u, r_m \in \mathbb{Z}_q$ at random and computes,

$$\begin{aligned}
t &= \sum_{i=1}^{n_u} t_i, c_i = u_i^{k_i^u} h^{t_i}, \pi_i = (u_i^{2k_i^u - 1} h^{t_i})^{t_i}, \\
\sigma_1 &= \tilde{\sigma}_1 \hat{g}_2^{\beta} \left(\hat{u}' \prod_{i=1}^{n_u} \hat{u}_i^{k_i^u} \right)^{r_u} \left(\hat{m}' \prod_{j=1}^{n_m} \hat{m}_j^{k_j^m} \right)^{r_m} h^{r_u t}, \\
\sigma_2 &= \tilde{\sigma}_2 \hat{g}^{r_u}, \sigma_3 = \tilde{\sigma}_3 \hat{g}^{r_m}, \sigma_4 = h^t, \sigma_5 = (\tilde{\sigma}_2 \hat{g}^{r_u})^t \\
\sigma &= (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, c_1, ..., c_{n_u}, \pi_1, ..., \pi_{n_u}).
\end{aligned}$$

The distribution of the group signature is the same as the real one. If $\mathcal{A}$ submits an identity $\mathfrak{u} = (k_1^u \cdots k_{n_u}^u)$, a message $\mathfrak{m} = (k_1^m \cdots k_{n_m}^m)$ and asks for a VEGS. $\mathcal{B}$ first generates a group signature according to the above steps. Then $\mathcal{B}$ chooses $s \in \mathbb{Z}_n$ and computes,

$$\begin{aligned}
\omega &= (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, c_1, ..., c_{n_u}, \pi_1, ..., \pi_{n_u}) \\
&= ((PK_T)^s \sigma_1, g^s, \sigma_2, \sigma_3, \sigma_4, \sigma_5, c_1, ..., c_{n_u}, \pi_1, ..., \pi_{n_u}).
\end{aligned}$$

The distribution of the VEGS is the same as the real one. Besides, $\mathcal{A}$ can also submit a VEGS ω, and a message $\mathfrak{m}$ to $\mathcal{B}$ for adjudication. $\mathcal{B}$ first checks whether the VEGS is valid. If it is invalid, then $\mathcal{B}$ responses with an empty symbol $\perp$. Otherwise, $\mathcal{B}$ runs Adj and returns the valid group signature σ to $\mathcal{A}$.

– **Forge.** Finally, $\mathcal{A}$ outputs a pair $(\mathfrak{m}^*, \omega^*)$. $\mathcal{B}$ first checks whether the VEGS is valid. If it is invalid, then the challenger returns 0. Otherwise, $\mathcal{B}$ runs Open

and obtains an identity $\mathfrak{u}^*$. If $\mathcal{A}$ has not queried a private key of identity $\mathfrak{u}^*$, a group signature or VEGS on $(\mathfrak{u}^*, \mathfrak{m}^*)$, then $\mathcal{A}$ successfully forges a valid VEGS.

And $\mathcal{B}$ can also forge a valid identity-based signature. $\mathcal{B}$ first decrypts the VEGS, and obtains a valid group signature,

$$\sigma_1^* = \frac{\omega_1^*}{\omega_2^{*\alpha_T}} = g_2^{\alpha_1} \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u} \right)^{r_u + r'_u} \left(m' \prod_{j=1}^{n_m} m_j^{k_j^m} \right)^{r_m} h^{(r_u + r'_u)t},$$

$$\sigma_2^* = \omega_3^* = g^{r_u + r'_u},$$

$$\sigma_3^* = \omega_4^* = g^{r_m},$$

$$\sigma_4^* = \omega_5^* = h^t,$$

$$\sigma_5^* = \omega_6^* = g^{(r_u + r'_u)t},$$

$$\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*, c_1, ..., c_{n_u}, \pi_1, ..., \pi_{n_u}).$$

Let $\gamma \in \mathbb{Z}_n$ be an integer and $\gamma \equiv 0 \pmod{q}$, $\gamma \equiv 1 \pmod{p}$ hold, then we have

$$e(\sigma_1^{*\gamma}, \tilde{g}) = e(\tilde{g}_2, \tilde{g}_1) e \left(\tilde{u}' \prod_{j=1}^{n_u} \tilde{u}_j^{k_i^u}, \sigma_2^{*\gamma} \right) e \left(\tilde{m}' \prod_{j=1}^{n_m} \tilde{m}_j^{k_j^m}, \sigma_3^{*\gamma} \right).$$

Thus $\mathcal{B}$ submits a tuple $(\mathfrak{u}^*, \mathfrak{m}^*, (\sigma_1^{*\gamma}, \sigma_2^{*\gamma}, \sigma_3^{*\gamma}))$ to his challenger. Since the signature has not been queried, $\mathcal{B}$ forges a valid identity-based signature $\sigma = (\sigma_1^{*\gamma}, \sigma_2^{*\gamma}, \sigma_3^{*\gamma})$. Therefore, if $\mathcal{A}$ breaks full-traceability of our VEGS scheme, then $\mathcal{B}$ also breaks the underlying identity-based signature scheme with the same advantage. Since the underlying identity-based signature scheme is unforgeable [10], our VEGS scheme satisfies full-traceability.

Theorem 3. *Our VEGS scheme is opaque if the aggregate extraction assumption holds on* $\mathbb{G}$.

Proof. Suppose an adversary $\mathcal{A}$ breaks opacity of our VEGS scheme with advantage at least ϵ, then there exists an adversary $\mathcal{B}$ that solves the aggregate extraction problem with a non-negligible probability. $\mathcal{B}$ and $\mathcal{A}$ play the game $\mathsf{Game}_{Opac}(\lambda)$, $\mathcal{B}$ simulates a challenger for $\mathcal{A}$ and tries to solve the given aggregate extraction problem on $\mathbb{G}$ (Given $\mathbb{G}, \mathbb{G}_p, \mathbb{G}_q, n = pq, p, q, g, g^a, g^b, g^\delta, g^\zeta, g^{ab+\delta\zeta}$, compute g^{ab}). We show how to construct $\mathcal{B}$.

– **Setup.** Let $\mathbb{G}$ and $\mathbb{G}_T$ be finite cyclic groups of order $n = pq$, $\mathbb{G}_p$ and $\mathbb{G}_q$ be subgroups of $\mathbb{G}$ of order p and q, g be a generator of $\mathbb{G}$, e be a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. $\mathcal{B}$ generates the system parameters as follows. $\mathcal{B}$ chooses $u', u_1, ..., u_{n_u}, m', m_1, ..., m_{n_m} \in \mathbb{G}$ at random, and sets $g_1 = g^a$, $g_2 = g^b$, $PK_T = g^\delta$. Besides, choose a generator $h \in \mathbb{G}_q$. We assume $h = g^\eta$ and η is known to $\mathcal{B}$. Then $\mathcal{B}$ sends the public parameters $\mathtt{param} = (g, h, g_1, g_2, u', u_1, ...u_{n_u}, m', m_1, ..., m_{n_m})$ and the adjudicator's public key $PK_T = g^\delta$ to $\mathcal{A}$. The distribution of the parameters are the same as

the real ones. Although $\mathcal{B}$ does not know $\mathsf{MK} = g_2^a$ and $SK_T = \delta$, we can still complete the simulation by playing some tricks. $\mathcal{B}$ first sets $l_u = 2(q_1+q_2+q_3)$ and $l_m = 2(q_2+q_3)$ ($\mathcal{A}$ can query at most q_1 times for private keys, q_2 times for group signatures and q_3 times for VEGSs), and chooses x', z', n_u-length vector $X = (x_i)$ and n_m-length vector $Z = (z_j)$ at random, where x' and x_i are random values in $\{0, ..., l_u\}$, z' and z_j are random values in $\{0, ..., l_m\}$. Besides, $\mathcal{B}$ picks y', w', n_u-length vector $Y = (y_i)$ and n_m-length vector $W = (w_j)$, where y', y_i, w' and w_j are random elements in $\mathbb{Z}_n$. Next $\mathcal{B}$ sets $u' = g_2^{-l_u k_1 + x'} g^{y'}$, $u_i = g_2^{x_i} g^{y_i}$, $m' = g_2^{-l_m k_2 + z'} g^{w'}$, $m_j = g_2^{z_j} g^{w_j}$, where $0 \le k_1 \le n_u$ and $0 \le k_2 \le n_m$. Then define the following functions,

$$F_1(\mathfrak{u}) = -l_u k_1 + x' + \textstyle\sum_{i=1}^{n_u} x_i k_i^u,\ K_1(\mathfrak{u}) = y' + \sum_{i=1}^{n_u} y_i k_i^u$$

$$F_2(\mathfrak{m}) = -l_m k_2 + z' + \textstyle\sum_{j=1}^{n_m} z_j k_j^m,\ K_2(\mathfrak{m}) = w' + \sum_{j=1}^{n_m} w_j k_j^m$$

And we have

$$u' \prod_{i=1}^{n_u} u_i^{k_i^u} = g_2^{F_1(\mathfrak{u})} g^{K_1(\mathfrak{u})}$$

$$m' \prod_{j=1}^{n_m} m_j^{k_j^m} = g_2^{F_2(\mathfrak{m})} g^{K_2(\mathfrak{m})}$$

– **Query.** Private key queries: If $\mathcal{A}$ submits an identity $\mathfrak{u} = (k_1^u \cdots k_{n_u}^u)$ to $\mathcal{B}$ and asks for a signing key, $\mathcal{B}$ randomly chooses $r_u \in \mathbb{Z}_n$, and computes,

$$d_{\mathfrak{u}} = (d_1, d_2) = \left(g_1^{\frac{-K_1(\mathfrak{u})}{F_1(\mathfrak{u})}} \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u} \right)^{r_u}, g_1^{\frac{-1}{F_1(\mathfrak{u})}} g^{r_u} \right).$$

Writing $\bar{r}_u = r_u - \frac{a}{F_1(\mathfrak{u})}$, then we have

$$\begin{aligned}
d_1 &= g_1^{\frac{-K_1(\mathfrak{u})}{F_1(\mathfrak{u})}} \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u} \right)^{r_u} \\
&= g_1^{\frac{-K_1(\mathfrak{u})}{F_1(\mathfrak{u})}} \left(g_2^{F_1(\mathfrak{u}} g^{K_1(\mathfrak{u})} \right)^{r_u} \\
&= g_2^a (g_2^{F_1(\mathfrak{u})} g^{K_1(\mathfrak{u})})^{-\frac{a}{F_1(\mathfrak{u})}} (g_2^{F_1(\mathfrak{u})} g^{K_1(\mathfrak{u})})^{r_u} \\
&= g_2^a \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u} \right)^{r_u - \frac{a}{F_1(\mathfrak{u})}} \\
&= g_2^a \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u} \right)^{\bar{r}_u}, \\
d_2 &= g_1^{-\frac{1}{F_1(\mathfrak{u})}} g^{r_u} = g^{r_u - \frac{a}{F_1(\mathfrak{u})}} \\
&= g^{\bar{r}_u} \\
d_3 &= g_1^{-\frac{\eta}{F_1(\mathfrak{u})}} h^{r_u} \\
&= h^{\bar{r}_u}
\end{aligned}$$

Therefore the private keys generated by $\mathcal{B}$ are indistinguishable from the real ones. Then $\mathcal{B}$ sends $d_{\mathfrak{u}} = (d_1, d_2, d_3)$ to $\mathcal{A}$.

Group signature queries: If $\mathcal{A}$ submits an identity $\mathfrak{u} = (k_1^u \cdots k_{n_u}^u)$, a message $\mathfrak{m} = (k_1^m \cdots k_{n_m}^m)$ to $\mathcal{B}$ and requests a group signature, $\mathcal{B}$ answers as follows. $\mathcal{B}$ chooses $r_u, r'_u, r_m, t_1, ..., t_{n_u} \in \mathbb{Z}_n$ at random, sets $t = \sum_{i=1}^{n_u} t_i$, $c_i = u_i^{k_i^u} h^{t_i}$, $\pi_i = (u_i^{2k_i^u - 1} h^{t_i})^{t_i}$, and computes,

$$\begin{aligned}
\sigma_1 &= g_1^{\frac{-K_2(\mathfrak{m}_\ell)}{F_2(\mathfrak{m}_\ell)}} \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u}\right)^{r_u} \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u}\right)^{r'_u} \left(m' \prod_{j=1}^{n_m} m_j^{k_j^m}\right)^{r_m} h^{(r_u + r'_u)t} \\
&= g_2^a \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u}\right)^{r_u + r'_u} \left(m' \prod_{j=1}^{n_m} m_j^{k_j^m}\right)^{\bar{r}_m} h^{(r_u + r'_u)t}, \\
\sigma_2 &= g^{r_u} g^{r'_u} = g^{r_u + r'_u}, \\
\sigma_3 &= g_1^{\frac{-1}{F_2(\mathfrak{m})}} g^{r_m} = g^{\bar{r}_m}, \\
\sigma_4 &= h^t, \\
\sigma_5 &= g^{(r_u + r'_u)t}, \\
\sigma &= (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, c_1, ...c_{n_u}, \pi_1, ..., \pi_{n_u}),
\end{aligned}$$

where $\bar{r}_m = r_m - \frac{a}{F_2(\mathfrak{m}_\ell)}$. The distribution of the group signature is the same as the real one.

VEGS queries: When $\mathcal{A}$ submits an identity $\mathfrak{u} = (k_1^u \cdots k_{n_u}^u)$, a message $\mathfrak{m} = (k_1^m \cdots k_{n_m}^m)$ to $\mathcal{B}$ and requests a VEGS, $\mathcal{B}$ can use a list $QueryList$ to response. $\mathcal{B}$ initializes list $QueryList := \emptyset$ and chooses random index $\ell^* \in \{1, ..., q_3\}$ to guess from which VEGS $\mathcal{A}$ selects and outputs the extraction. And $\mathcal{A}$ has not queried for a signing key at $\mathfrak{u}_{\ell^*}$ or group signature at $(\mathfrak{u}_{\ell^*}, \mathfrak{m}_{\ell^*})$. If $\ell \neq \ell^*$, $\mathcal{B}$ first generates a group signature $\sigma_\ell = (\sigma_{1,\ell}, \sigma_{2,\ell}, \sigma_{3,\ell}, \sigma_{4,\ell}, \sigma_{5,\ell}, c_1, ...c_{n_u}, \pi_1, ..., \pi_{n_u})$ as he does in group signature queries. Next $\mathcal{B}$ chooses $s \in \mathbb{Z}_n$ at random and computes,

$$\omega_{1,\ell} = (PK_T)^s \sigma_{1,\ell} = (PK_T)^s g_2^a \left(u' \prod_{i=1}^{n_u} u_i^{k_i^u}\right)^{r_u + r'_u} \left(m' \prod_{j=1}^{n_m} m_j^{k_j^m}\right)^{\tilde{r}_m} h^{(r_u + r'_u)t},$$

$$\omega_{2,\ell} = g^s, \omega_{3,\ell} = \sigma_{2,\ell} = g^{r_u + r'_u}, \omega_{4,\ell} = \sigma_{3,\ell} = g^{\bar{r}_m}$$

$$\omega_{5,\ell} = \sigma_{4,\ell} = h^t, \omega_{6,\ell} = \sigma_{5,\ell} = g^{(r_u + r'_u)t}$$

$$\omega_\ell = (\omega_{1,\ell}, \omega_{2,\ell}, \omega_{3,\ell}, \omega_{4,\ell}, c_1, ...c_{n_u}, \pi_1, ..., \pi_{n_u})$$

$\mathcal{B}$ sends ω_ℓ to $\mathcal{A}$ and stores the tuple $(\mathfrak{u}_\ell, \mathfrak{m}_\ell, \sigma_\ell, \omega_\ell)$ in $QueryList$. If $\ell = \ell^*$, then $\mathcal{B}$ will embed the instance. $\mathcal{B}$ randomly chooses $r_u, r'_u, r_m, t_1, ..., t_{n_u} \in \mathbb{Z}_n$ and sets,

$$\omega_{1,\ell^*} = g^{ab+\delta\zeta}\left(u'\prod_{i=1}^{n_u} u_i^{k_i^u}\right)^{r_u+r'_u}\left(m'\prod_{j=1}^{n_m} m_j^{k_j^m}\right)^{r_m} h^{(r_u+r'_u)t}$$

$$= (g^{\delta})^{\zeta} g_2^a\left(u'\prod_{i=1}^{n_u} u_i^{k_i^u}\right)^{r_u+r'_u}\left(m'\prod_{j=1}^{n_m} m_j^{k_j^m}\right)^{r_m} h^{(r_u+r'_u)t}$$

$$\omega_{2,\ell^*} = g^{\zeta}, \omega_{3,\ell^*} = g^{r_u+r'_u}, \omega_{4,\ell^*} = g^{r_m}, \omega_{5,\ell^*} = h^t, \omega_{6,\ell^*} = g^{(r_u+r'_u)t}$$

$$c_i = u_i^{k_i^u} h^{t_i}, \pi_i = (u_i^{2k_i^u-1} h^{t_i})^{t_i}$$

$$\omega_{\ell^*} = (\omega_{1,\ell^*}, \omega_{2,\ell^*}, \omega_{3,\ell^*}, \omega_{4,\ell^*}, \omega_{5,\ell^*}, \omega_{6,\ell^*}, c_1, ...c_{n_u}, \pi_1, ..., \pi_{n_u}).$$

$\mathcal{B}$ sends ω_{ℓ^*} to $\mathcal{A}$ and stores the tuple $(\mathfrak{u}_{\ell^*}, \mathfrak{m}_{\ell^*}, \sigma_{\ell^*}, \omega_{\ell^*})$ in *QueryList*. If one of $F_1(\mathfrak{u}_\ell) = 0$, $F_2(\mathfrak{m}_\ell) = 0$, $F_1(\mathfrak{u}_{\ell^*}) \neq 0$, $F_2(\mathfrak{m}_{\ell^*}) \neq 0$ holds, $\mathcal{B}$ stops the game (If one of them holds, $\mathcal{B}$ can not solve his problem). We denote this event by S_1. Otherwise, the distribution of the VEGSs are the same with the real ones.

Adjudication queries: In this phase, $\mathcal{A}$ is not allowed to make a query on $(\mathfrak{m}_{\ell^*}, \omega_{\ell^*})$. When $\mathcal{A}$ submits a message $\mathfrak{m}_{\ell'} = (k_1^m \cdots k_{n_m}^m)$, a VEGS $\omega_{\ell'}$ to $\mathcal{B}$ and requests a group signature, $\mathcal{B}$ does as follows. $\mathcal{B}$ first checks whether the VEGS is valid. If it is invalid, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ checks whether the pair $(\mathfrak{m}_{\ell'}, \omega_{\ell'})$ exists in the list *QueryList*, if it is not in *QueryList*, then the VEGS is invalid and $\mathcal{B}$ returns $\perp$ (If the VEGS is valid, then $\mathcal{A}$ forges a valid VEGS, and this contradicts the unforgeability of our VEGS scheme).

If the tuple is in the list *QueryList*, and $\ell' = \ell$, then $\mathcal{B}$ finds out the tuple $(\mathfrak{u}_\ell, \mathfrak{m}_\ell, \sigma_\ell, \omega_\ell)$, and returns σ_ℓ to $\mathcal{A}$. The above simulation is perfect if $\mathcal{B}$ has not aborted.

– **Forge.** Finally, $\mathcal{A}$ outputs a valid signature $\sigma_{\ell^*} = (\sigma_{1,\ell^*}, \sigma_{2,\ell^*}, \sigma_{3,\ell^*}, \sigma_{4,\ell^*}, \sigma_{5,\ell^*}, c_1, ..., c_{n_u}, \pi_1, ..., \pi_{n_u})$ (on message $\mathfrak{m}_{\ell^*}$) of identity $\mathfrak{u}_{\ell^*}$ with a non-negligible probability ϵ. That means $\mathcal{B}$ correctly guesses from which VEGS $\mathcal{A}$ extracts the group signature, and we denote this event by S_2.

Then $\mathcal{B}$ solves his problem by computing,

$$g^{ab} = \frac{\sigma_{1,\ell^*}}{(\sigma_{2,\ell^*})^{K_1(\mathfrak{u}_{\ell^*})}(\sigma_{3,\ell^*})^{K_2(\mathfrak{m}_{\ell^*})}\sigma_{5,\ell^*}^{\eta}}$$

The probability that $\mathcal{B}$ wins in the above game is as follows.

$$Pr[S_1 \wedge S_2] = Pr[S_1]Pr[S_2].$$

The probability that $\mathcal{B}$ correctly guesses the index ℓ^* is $1/q_3$. Since we use the proof techniques in [19], we deduce that $Pr[S_1] \geq 1/(16(q_1+q_2+q_3)(q_2+q_3)(n_u+1)(n_m+1))$. Thus we have

$$Pr[S_1 \wedge S_2] \geq \frac{1}{16(q_1+q_2+q_3)(q_2+q_3)(n_u+1)(n_m+1)} \cdot \frac{1}{q_3}$$

$$= \frac{1}{16q_3(q_1+q_2+q_3)(q_2+q_3)(n_u+1)(n_m+1)}$$

and the probability that $\mathcal{B}$ solves the aggregate extraction problem is at least $\epsilon/(16q_3(q_1+q_2+q_3)(q_2+q_3)(n_u+1)(n_m+1))$, which is non-negligible.

Theorem 4. *Our VEGS scheme is extractable.*

Proof. The challenger plays the game $\mathsf{Game}_{Extr}(\lambda)$ with an adversary $\mathcal{A}$ as follows.

- **Setup.** $\mathcal{B}$ runs Setup and AKG and generates system parameters of VEGS scheme, and sends the public parameters $(\mathtt{param}, PK_T)$ to $\mathcal{A}$.
- **Query.** In this phase, the challenger runs algorithms Enroll, Sign, VESign and Adj to response $\mathcal{A}$.
- **Forge.** Finally, $\mathcal{A}$ submits a tuple $(\mathtt{m}^*, \omega^*, \mathtt{param}^*)$ to the challenger.
- **Extract.** If the given VEGS $\omega^* = (\omega_1^*, \omega_2^*, \omega_3^*, \omega_4^*, \omega_5^*, \omega_6^*, c_1, ...c_{n_u}, \pi_1, ..., \pi_{n_u})$ passes the check, then we can obtain a valid identity $\mathtt{u} = (k_1^u \cdots k_{n_u}^u)$. Besides, we have

$$e(\omega_1^*, g) = e(PK_T, \omega_2^*)e(g_2, g_1)e\left(c, \omega_3^*\right)e\left(M, \omega_4^*\right),$$
$$e(\omega_3^*, \omega_5^*) = e(\omega_6^*, h).$$

Then a group signature can be extracted by computing,

$$\sigma_1^* = \frac{\omega_1^*}{\omega_2^{*\alpha_T}}, \sigma_2^* = \omega_3^*, \sigma_3^* = \omega_4^*, \sigma_4^* = \omega_5^*, \sigma_5^* = \omega_6^*$$

$$\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*, c_1, ...c_{n_u}, \pi_1, ..., \pi_{n_u}).$$

and we have

$$\begin{aligned} e(\sigma_0^*, g) &= e\left(\frac{\omega_1^*}{\omega_2^{*\alpha_T}}, g\right) \\ &= e(\omega_1^*, g)e(\omega_2^{*\alpha_T}, g)^{-1} \\ &= e(PK_T, \omega_2^*)e(g_2, g_1)e\left(c, \omega_3^*\right)e\left(M, \omega_4^*\right)e(\omega_2^*, g^{\alpha_T})^{-1} \\ &= e(g_2, g_1)e\left(c, \sigma_2^*\right)e\left(M, \sigma_3^*\right). \\ e(\omega_3^*, \omega_5^*) &= e(\sigma_2^*, \sigma_4^*) = e(\sigma_5^*, h) = e(\omega_6^*, h). \end{aligned}$$

It implies that if $\mathsf{VEVerify}(\mathtt{m}^*, \omega^*, PK_T, \mathtt{param}^*) = 1$ holds, then Verify $(\mathtt{m}^*, \sigma^*, \mathtt{param}^*) = 1$ always holds as well. Thus our VEGS scheme is extractable.

5 Extensions

In this section, we will discuss some extensions about our scheme.

5.1 Other Properties

In above paper, we discussed main properties of VEGS according to the security requirements of VES and group signature. In fact, there are other crucial properties for VEGS, we will give more details in this subsection.

Exculpability, first proposed by Chaum and Heystis [11], is also significant to group signature schemes. And we extend it to VEGS schemes. A VEGS scheme satisfies exculpability if on one can create VEGSs on behalf of other honest group members. Consider a malicious user who wishes to forge a VEGS on behalf other users. If he is not the group master, then he will not succeed to generate a valid VEGS if the VEGS scheme satisfies unforgeability. Then we consider the case that the malicious user is the group master. Ateniese et al. [1] pointed that Boyen-Waters group signature scheme does not satisfy (strong) exculpability because the group master generates and distributes users' secret keys, however their scheme can achieve exculpability by changing some settings to the group master. In their scheme, the group master is an ephemeral entity and the master key is destroyed once the group is set up. To achieve the exculpability of our VEGS scheme, we can construct the group master in the same way. Therefore, no one can create a valid VEGS on behalf of other users.

Coalition resistance means that if a group of signers collude together to generate a valid VEGS, then it must be traceable. Coalition resistance emphasizes the fact that it allows attacks by a coalition of group members. However, coalition resistance can still be obtained from full-traceability [9]. Therefore, we deduce that fully-traceable VEGS schemes are also coalition resistant.

Unlinkability requires that on one can determine whether two different VEGS are generated by the same group member except the group manager. Given two different VEGSs, if one (except the group manager) wishes to check whether they are created by the same user, he has to recover the identity of the signer. Then he breaks the anonymity of the VEGS if he succeeds with a non-negligible probability. It implies the anonymity immediately. Thus we can deduce that fully-anonymous VEGS schemes also satisfy unlinkability.

5.2 Batch Verification

To improve efficiency of our VEGS scheme, some measures can be taken. One method is to perform fast batch verification [10,12]. The generic definition of batch verification was given by Bellare et al. [8], then Camenisch et al. [12] instantiates it to the case of signatures from many signers and aggregate signatures. We can also use their method to simplify the verification of our VEGS scheme. Suppose a verifier wish to check if a VEGS is valid, and the different things he need to do is that he chooses $\theta_1, ..., \theta_{n_u} \in \mathbb{Z}_n$, then tests,

$$\prod_{i=1}^{n_u} e(c_i^{\theta_i}, u_i^{-1} c_i) e(h^{-\theta_i}, \pi_i) \stackrel{?}{=} 1.$$

Since we batch the pairs into a multi-pairing, which is similar to multi-exponentiation algorithm, we can reduce the cost of the pairing computations.

As stated in [10], to get better efficiency, some pre-computations and extra storage are also required.

5.3 Dynamic Groups

The above VEGS scheme is called a static VEGS scheme since we do not consider the case where users join and leave after the group is set up. To achieve dynamic groups where users can both join and leave the group, we need to add some modifications to the VEGS scheme. When a user is allowed to join the group, the group master distributes the user's private key with the group master key. When a user leaves the group, it is very different for the discussion of leave operation. The group master needs to publish the recovered signing keys, then he generates a new group master key and distributes each user's private key. And what calls for attention is that the revocation information is published on public channel while the signing keys are transferred in secret channel. The above method can also be used in the cases where multiple users are revoked.

Besides, someone may find that the group master in the dynamic VEGS is not an ephemeral entity, it involves in the scheme when users join or leave the group. Therefore the weakness of our VEGS scheme is that it can not achieve dynamic groups and exculpability simultaneously. However we believe it will be solved in the future works.

6 Conclusion

In this paper, we formalized the concept of VEGS which is derived from VES and group signature. Then we presented the first VEGS scheme based on Boyen-Waters group signature scheme and ElGamal encryption scheme. We defined the security properties which are necessary for VEGS schemes, i.e., anonymity, traceability, unforgeability, opacity and extractability. Then we proved our VEGS scheme is secure in the standard model according to the definitions. Additionally, we discussed the extentions of our VEGS scheme. The results showed that our VEGS scheme has many applications. However, there still exists a few open problems (e.g., achieving dynamic groups and exculpability simultaneously, using prime order groups), which will motivate more works on VEGS.

Acknowledgment. This paper is supported by Collaborative Innovation Center Of Geospatial Technology (No. ZF102T1701), by Beijing Municipal Science and Technology Project (No. D161100005816001), and by the Natural Science Foundation of China through projects 61672083.

References

1. Ateniese, G., Camenisch, J., Hohenberger, S., Medeiros, B.D.: Practical group signatures without random oracles. In: Theory and Application of Cryptographic Techniques (2005)
2. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000). doi:10.1007/3-540-44598-6_16
3. Ateniese, G., Song, D., Tsudik, G.: Quasi-efficient revocation of group signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003). doi:10.1007/3-540-36504-4_14
4. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptol. **21**(2), 149–177 (2008)
5. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28628-8_3
6. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_26
7. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30576-7_18
8. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998). doi:10.1007/BFb0054130
9. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_38
10. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006). doi:10.1007/11761679_26
11. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991). doi:10.1007/3-540-46416-6_22
12. Camenisch, J., Hohenberger, S., Pedersen, M.Ø.: Batch verification of short signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 246–263. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72540-4_14
13. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002). doi:10.1007/3-540-45708-9_5
14. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985). doi:10.1007/3-540-39568-7_2
15. Galindo, D., Herranz, J., Kiltz, E.: On the generic construction of identity-based signatures with additional properties. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 178–193. Springer, Heidelberg (2006). doi:10.1007/11935230_12

16. Gu, C., Zhu, Y.F.: An ID-based verifiable encrypted signature scheme based on Hess's scheme. In: Feng, D., Lin, D., Yung, M. (eds.) CISC 2005. LNCS, vol. 3822, pp. 42–52. Springer, Heidelberg (2005). doi:10.1007/11599548_4
17. Lu, S., Lynn, B., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures, multisignatures, and verifiably encrypted signatures without random oracles. J. Cryptol. **26**(2), 340–373 (2013)
18. Nishimaki, R., Xagawa, K.: Verifiably encrypted signatures with short keys based on the decisional linear problem and obfuscation for encrypted VES. Des. Codes Crypt. **77**(1), 61–98 (2015)
19. Paterson, K.G., Schuldt, J.C.N.: Efficient identity-based signatures secure in the standard model. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 207–222. Springer, Heidelberg (2006). doi:10.1007/11780656_18
20. Rückert, M., Schröder, D.: Security of verifiably encrypted signatures and a construction without random oracles. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 17–34. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03298-1_2
21. Zhang, Z.F.: Cryptanalysis of an identity-based verifiably encrypted signature scheme. Chin. J. Comput. **29**(9), 1688–1693 (2006)
22. Zhang, L., Wu, Q.H., Qin, B.: Identity-based verifiably encrypted signatures without random oracles. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 76–89. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04642-1_8