Group-Oriented Fair Exchange of Signatures

Qiong Huang* Duncan S. Wong* Willy Susilo†

Abstract

In an Optimistic Fair Exchange (OFE) for digital signatures, two parties exchange their signatures fairly without requiring any online trusted third party. The third party is only involved when a dispute occurs. In all the previous work, OFE has been considered only in a setting where both of the communicating parties are individuals. There is little work discussing about the fair exchange between two *groups* of users, though we can see that this is actually a common scenario in actual OFE applications. In this paper, we introduce a new variant of OFE, called *Group-Oriented Optimistic Fair Exchange* (GOFE). A GOFE allows two users from two different groups to exchange signatures on behalf of their groups in a fair and anonymous manner. Although GOFE may be considered as a fair exchange for group signatures, it might be inefficient if it is constructed generically from a group signature scheme. Instead, we show that GOFE is *backward compatible* to the Ambiguous OFE (AOFE). Also, we propose an efficient and concrete construction of GOFE, and prove its security under the security models we propose in this model. The security of the scheme relies on the decision linear assumption and strong Diffie-Hellman assumption under the random oracle model.

Keywords. fair exchange, signature, ambiguity, fairness, group signature, random oracle model.

*Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong S.A.R., China. Emails: csqhuang@student.cityu.edu.hk, duncan@cityu.edu.hk.

†School of Computer Science and Software Engineering, University of Wollongong, Northfields Avenue, New South Wales 2522, Australia. Email: wsusilo@uow.edu.au.

Contents

1 Introduction

Optimistic fair exchange (OFE), introduced by Asokan, Schunter and Waidner [1] in 1997, aims to solve the problem of fairly exchanging items between two parties say, Alice and Bob. In OFE, there is an arbitrator semi-trusted by Alice and Bob, and involves into a transaction only when one party attempts to cheat the other or crashes. Later, Asokan, Shoup and Waidner proposed a protocol for optimistically fairly exchanging digital signatures [2], in which Alice, the signer, first sends a partial signature to Bob, the verifier, which then returns his full signature; in response, Alice then sends back her full signature. If everything goes well and they honestly follow the protocol, Alice and Bob both get their counterpart's full signature; however, if Alice refuses or fails to respond, Bob then resorts to the arbitrator for resolving her partial signature into a full one.

(Related Work). Most of the previous works on optimistic fair exchange are in the single-user setting, in which there are only one signer and one verifier, along with an arbitrator. Dodis, Lee and Yum [14] initiated the study of the security of OFE in the multi-user setting, in which there are multiple signers and multiple verifiers (along with one arbitrator), by showing that security of OFE in the single-user setting does not necessarily imply security in the multi-user setting. They proposed an OFE scheme secure in the multi-user setting and the certified-key model in which the adversary has to prove its knowledge of the secret key before using a public key. Huang, Yang, Wong and Susilo [18] further improved their results and proposed to consider the security of OFE in a more relaxed model, called chosen-key model, in which the adversary is free to use any public key without showing its knowledge of the corresponding secret key. They showed that security of OFE in the certified-key model does not imply security in the chosen-key model.

Traditional OFE has a potential weakness in fairness/abuseness. Specifically, after receiving Alice's partial signature, instead of returning his full signature, Bob turns to a third party and convinces it that the partial signature was indeed generated by Alice. This is possible because the validity of Alice's partial signature does show that she committed herself to something. In this way Bob may be able to benefit, while Alice gains nothing. Clearly it is unfair for Alice. Hence, Huang, Yang, Wong and Susilo [17] introduced the notion of *ambiguous optimistic fair exchange* (AOFE), which is similar to *abuse-free optimistic contract signing* introduced by Garay, Jakobsson and MacKenzie [15], aiming to solve the aforementioned problem. AOFE differs from the original OFE in that Alice's partial signature is *ambiguous* in the sense that Bob is also able to produce signatures on the same message, which are indistinguishable from those generated by Alice. Thus, the aforementioned problem does not occur again, as anyone else would not believe in Bob that a partial signature is binding to Alice.

(Our Work). To the best of our knowledge, until now there is no work on OFE discussing about a setting in which either party consists of a group of users. That is, either or both of Alice and Bob are groups of users. In this paper we study OFE in the latter setting, i.e. both parties are groups of users. We introduce the notion of *group-oriented optimistic fair exchange* (GOFE), which in essence is an AOFE of group signatures [11].

In GOFE, there are two groups of users, i.e. a signing group G_A and a verifying group G_B, along with an arbitrator. Each group has a group manager, which takes charge of the enrolment of users by issuing user secret keys to them. After joining a group and obtaining its user secret key, a user in a group then can sign documents on behalf of its group, and exchange signatures with (another user in) the other group. Anyone, *including the arbitrator but except the group manager*, cannot find out the identity of the real signer. While in AOFE, though the partial signature is ambiguous, any third party can tell from the validity of the signature that it must be either Alice or Bob who signed the document.

GOFE has practical applications. For example, consider the case in which two companies say,

Macrosoft and Doodle want to sign an electronic contract online. Any board member of Macrosoft or Doodle can sign the contract on behalf of their respective company. The chairman of the board acts as the group manager. After sending out the partial signature, a board member in Macrosoft certainly does not want company Doodle to gain any advantage from the signature; meanwhile, it does not want Doodle to find out who he/she really is either, for the sake of privacy. Doodle is ensured is that the partial signature is from someone in the board of Macrosoft, but it is prevented from convincing anyone else this fact due to the ambiguity of the partial signature. Besides, in order to restrain board members from abusing their privilege, the chairman of either company surely wants to find out who signed the contract on behalf of their company.

In this work we give the formal definition of GOFE, which combines the definition of group signatures and that of AOFE. Like group signatures, GOFE also has an '*open*' algorithm, which binds a *valid* full signature to someone in the group. We also propose formal security models for GOFE. We extend the model of *signer ambiguity* of AOFE to GOFE and define the property *group ambiguity*, which basically says that nobody but the arbitrator can tell a valid *partial* signature was generated by a user in which of the two groups. We also define the property *user anonymity*, which says that no one can tell which user in group G_A generated the (partial) signature. We show that if a GOFE scheme is group ambiguous, it is also user anonymous.

We propose an efficient construction of GOFE, based on Boneh, Boyen and Shacham's group signature scheme [6] and Kiltz' tag-based public key encryption scheme [19]. Suppose that the signer is in group G_A. In our GOFE scheme, the user secret key of the signer is encrypted under the public key of G_A, and G_A's public key is also encrypted under the arbitrator's public key. Let the two ciphertexts be C and D respectively. Then a signature of knowledge is attached to show that C contains the user secret key of a user in either group G_A or group G_B. To resolve a partial signature, the arbitrator decrypts D to obtain the group public key, and then determine the group accordingly. Based on the hardness of decision linear problem and strong Diffie-Hellman problem, we show that the proposed GOFE scheme is secure under the given security models in the random oracle model.

(GOFE v.s. **AOFE)**. In a nutshell, GOFE and AOFE are two different but closely related variants of the original definition of optimistic fair exchange [2]. They share all the security properties that an OFE scheme has, however, there is a major difference between them. That is, AOFE is a protocol for exchanging standard signatures, while GOFE is a protocol for exchanging group signatures. On the other hand, GOFE is an extension of AOFE in the group setting. We show in Sec. 6 that GOFE is *backward compatible* with AOFE. Specifically, we give a generic transform from GOFE to AOFE without any security loss.

(Discussion). A way of constructing a GOFE scheme is to transform a group signature to a verifiably encrypted one, an analogy to the verifiably encrypted signature [7]. Say, the partial signature of the signing group is a group signature verifiably encrypted under the arbitrator's public key. It is feasible, however, for the sake of an efficient construction, it is not trivial. As far as we know, almost all the recently proposed efficient group signature schemes, for example, [3, 4, 6, 8, 9, 10, 16], basically follow the following paradigm:

1. sign the message using user secret key to obtain σ;
2. encrypt σ under the group public key to obtain e; and
3. provide a non-interactive proof π showing that e contains a signature generated by a user in the group.

To construct a GOFE scheme from a group signature scheme, a direct idea is to encrypt σ under the arbitrator's public key, and then modify the proof π accordingly to show that e contains either group

G_A or G_B's signature. This method has the problem that the arbitrator might be able to reveal the identity of the real signer, which is what we want to avoid. Another possible way is to encrypt e under the arbitrator's public key to obtain a new ciphertext e', and provide another proof π' showing that the ciphertext e' contains an encryption of a signature generated by a user in either group G_A or G_B. The problem is that the proof π' becomes complicated, making the method not efficient enough for practical use. Therefore, it is interesting to find a practical and efficient construction of GOFE.

(Paper Organization). Next we give the formal definition of GOFE, and its security models. In Sec. 3 we review some number-theoretical assumptions that will be used in our construction of GOFE. In Sec. 4 we present a core tool for the construction, i.e. a zero-knowledge proof of knowledge for a specific problem related to strong Diffie-Hellman assumption. The efficient construction of GOFE is then given in Sec. 5, along with its security analysis under the proposed model. In Sec. 6 we give the generic transform from GOFE to AOFE. Finally the paper is concluded in Sec. 7.

2 Group-Oriented Optimistic Fair Exchange

Roughly speaking, Group-oriented Optimistic Fair Exchange (GOFE) is the optimistic fair exchange of group signatures. In GOFE, there are two groups, i.e. the signing group and the verifying group, and an arbitrator, which is involved only when there is any dispute between the two groups. Each group has a group manager equipped with a public/private key pair, and is in charge of issuing secret keys to users who join the group. Any member in a group can sign a message on behalf of the group which it belongs to, without revealing any information about its identity to anyone except the group manager, which is able to trace the identity of the real signer of a valid signature. Meanwhile, a partial signature should not confirm the identity of a group either, except the fact that the signature is from either the signing group or the verifying group. Below we give the formal definition of GOFE.

Definition 2.1 (Group-oriented Optimistic Fair Exchange (GOFE))**.** *A* Group-oriented optimistic fair exchange *scheme consists of the following (probabilistic) polynomial-time algorithms:*

- PMGen*: On input* 1^k *where* k *is a security parameter, it outputs a system parameter* PM.
- $\mathsf{Setup}^{\mathsf{TTP}}$*: On input* PM*, the algorithm generates a public arbitration key* $\mathtt{apk}$ *and a secret arbitration key* $\mathtt{ask}$ *for the arbitrator.*
- $\mathsf{Setup}^{\mathsf{Group}}$*: On input* PM *and (optionally)* $\mathtt{apk}$*, the algorithm outputs a public/secret key pair* $(\mathtt{gpk}, \mathtt{gsk})$ *for a group. For group* G_i*, we use* $(\mathtt{gpk}_i, \mathtt{gsk}_i)$ *to denote its key pair.*
- Join*: On input a group secret key* $\mathtt{gsk}$ *and (optionally) a user's identity* $\mathtt{ID}_t$*, the algorithm outputs a user secret key* $\mathtt{usk}_t$ *for the user.*
- GSig *and* GVer*:* $\mathsf{GSig}(M, \mathtt{usk}_t, \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk})$ *outputs a (full) signature* σ_F *on* M *of group* G_i *with the verifying group* G_j*, where message* M *is chosen from the message space* $\mathcal{M}$ *and* $\mathtt{usk}_t$ *is the user secret key of the user with identity* $\mathtt{ID}_t$ *in group* G_i*, while* $\mathsf{GVer}(M, \sigma_F, \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk})$ *outputs* 1 *or* 0*, indicating* σ_F *is a valid full signature on* M *of* G_i *with verifying group* G_j *or not.*
- GPSig *and* GPVer*: They are partial signing and verification algorithms respectively.* $\mathsf{PSig}(M, \mathtt{usk}_t, \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk})$ *outputs a partial signature* σ_P*, while* $\mathsf{PVer}(M, \sigma_P, \mathbf{gpk}, \mathtt{apk})$ *outputs* 1 *or* 0*, indicating* σ_P *is a valid partial signature on* M *of group* G_i *or* G_j*, where* $\mathbf{gpk} := \{\mathtt{gpk}_i, \mathtt{gpk}_j\}$.
- Res*: This is the resolution algorithm.* $\mathsf{Res}(M, \sigma_P, \mathtt{ask}, \mathbf{gpk})$*, where* $\mathbf{gpk} = \{\mathtt{gpk}_i, \mathtt{gpk}_j\}$*, outputs a full signature* σ_F*, or* $\perp$ *indicating the failure of resolving a partial signature.*
- Trace*: This is the trace algorithm.* $\mathsf{Trace}(M, \sigma_F, \mathtt{gsk}_i, \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk})$*, where* σ_F *is a* valid *full signature of group* G_i *with respect to* $\mathtt{gpk}_i, \mathtt{gpk}_j$ *and* $\mathtt{apk}$*, outputs either* $\perp$ *indicating failure, or an identity* $\mathtt{ID}$.

The *correctness* can be defined in a natural way. Namely, we require that for any $k \in \mathbb{N}$, $\mathsf{PM} \leftarrow \mathsf{PMGen}(1^k)$, $(\mathtt{apk}, \mathtt{ask}) \leftarrow \mathsf{Setup}^{\mathsf{TTP}}(\mathsf{PM})$, $(\mathtt{gpk}_i, \mathtt{gsk}_i) \leftarrow \mathsf{Setup}^{\mathsf{Group}}(\mathsf{PM}, \mathtt{apk})$, $(\mathtt{gpk}_j, \mathtt{gsk}_j) \leftarrow \mathsf{Setup}^{\mathsf{Group}}(\mathsf{PM}, \mathtt{apk})$, $\mathtt{usk} \leftarrow \mathsf{Join}(\mathtt{gsk}_i, \mathtt{ID})$, and $M \leftarrow \mathcal{M}$, let $\mathbf{gpk} := \{\mathtt{gpk}_i, \mathtt{gpk}_j\}$, we have that

$$\begin{aligned} \mathsf{GPVer}(M, \mathsf{GPSig}(M, \mathtt{usk}, \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk}), \mathbf{gpk}, \mathtt{apk}) &= 1, \\ \mathsf{GVer}(M, \mathsf{GSig}(M, \mathtt{usk}, \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk}), \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk}) &= 1, \quad \text{and} \\ \mathsf{GVer}(M, \mathsf{Res}(M, \mathsf{GPSig}(M, \mathtt{usk}, \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk}), \mathtt{ask}, \mathbf{gpk}), \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk}) &= 1. \end{aligned}$$

The *resolution ambiguity* requires that the 'resolved' signatures output by the arbitrator, i.e. $\mathsf{Res}(M, \mathsf{GPSig}(M, \mathtt{usk}, \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk}), \mathtt{ask}, \mathbf{gpk})$, are *computationally indistinguishable* from the 'actual' signatures output by the real signer, i.e. $\mathsf{GSig}(M, \mathtt{usk}, \mathtt{gpk}_i, \mathtt{gpk}_j, \mathtt{apk})$. A secure GOFE scheme should satisfy group ambiguity, user anonymity, security against signing groups, security against verifying groups and traceability. We will define them in the following subsection, respectively.

2.1 Security Properties

(Group Ambiguity). Roughly, *group ambiguity* requires that it is infeasible for an adversary to tell if a given (valid) partial signature was generated by the signing group or the verifying group, even it knows the secret keys of both group managers. This is for protecting the fairness of the signing group, i.e. preventing the verifying group from using the signature for other purposes. To achieve it, we endow members of the verifying group the ability of producing indistinguishable partial signatures on the same message. Formally, we consider the game G_{ga} depicted in Fig. 1 (page 6), in which $\mathcal{D}$ is a probabilistic polynomial-time distinguisher. The advantage of $\mathcal{D}$, denoted by $\mathtt{Adv}^{\mathrm{GA}}_{\mathcal{D}}(k)$, is defined as the absolute value of the difference between $\mathcal{D}$'s success probability in the game above and $1/2$.

Definition 2.2 (Group Ambiguity)**.** *A Group-oriented OFE scheme is said to be* (t, q_r, ϵ)-group ambiguous *if there is no algorithm* $\mathcal{D}$ *which runs in time at most* t*, makes at most* q_r *queries to* $\mathcal{O}_{\mathsf{R}}$ *and wins the game above with advantage at least* ϵ*.*

Remark 1 : In game G_{ga} the adversary $\mathcal{D}$ chooses both of the two group key pairs. Therefore, we do not give the tracing oracle to $\mathcal{D}$, as it already knows the group secret keys. Also note that in the game above we do not require that $\mathtt{ID}_0$ and $\mathtt{ID}_1$ should be different. The group ambiguity here considers that partial signatures of two signers in different groups on the same message are indistinguishable. Similarly, we can define '*User Anonymity*', which considers that partial signatures of two signers from the same group on the same message are indistinguishable. The game G_{ua} is depicted in Fig. 1. The advantage of $\mathcal{D}$ in the distinguishing game above is defined to be $\mathtt{Adv}^{\mathrm{UA}}_{\mathcal{D}}(k) := |\Pr[b' = b] - 1/2|$.

Definition 2.3 (User Anonymity)**.** *A Group-oriented OFE scheme is said to be* (t, q_r, ϵ)-user anonymous *if there is no algorithm* $\mathcal{D}$ *which runs in time at most* t*, makes at most* q_r *queries to* $\mathcal{O}_{\mathsf{R}}$ *and wins the game above with advantage at least* ϵ*.*

Remark 2 : We stress that the *user anonymity* is different from the *signer ambiguity* in [17]. The signer ambiguity is closely related to group ambiguity defined above. It is defined in the setting in which individual users instead of groups of users are involved in a transaction. It requires that the partial signatures of the two individual users are indistinguishable, while the user anonymity requires that partial signatures of two users in the same group are indistinguishable.

(Security Against Signing Groups). We require that no PPT adversary $\mathcal{A}$ should be able to produce a partial signature with non-negligible probability, which looks good to a verifying group but cannot be resolved to a full signature by the honest arbitrator. This ensures the fairness for the

verifying group, that is, if a signer in the signing group has committed to a message with respect to a verifying group, the verifying group should always be able to obtain the full commitment of the signing group. Formally, we consider the game G_{sasg} depicted in Fig. 1. Note that the adversary is not allowed to corrupt $\mathtt{gpk}_B$, otherwise it can easily succeed in the game by simply outputting a partial signature generated under group public keys $\mathtt{gpk}_A, \mathtt{gpk}_B$ using any user secret key derived from $\mathtt{gsk}_B$. The advantage of $\mathcal{A}$ in the game, $\mathtt{Adv}^{\mathrm{SASG}}_{\mathcal{A}}(k)$, is defined to be $\mathcal{A}$'s success probability.

Definition 2.4 (Security Against Signing Groups)**.** *A GOFE scheme is said to be* $(t, q_p, q_j, q_t, q_r, \epsilon)$-secure against signing groups *if there is no adversary* $\mathcal{A}$ *which runs in time at most* t*, makes at most* q_p *queries to* $\mathcal{O}_{\mathsf{P}}$*,* q_j *queries to* $\mathcal{O}_{\mathsf{J}}$*,* q_t *queries to* $\mathcal{O}_{\mathsf{T}}$ *and* q_r *queries to* $\mathcal{O}_{\mathsf{R}}$*, and wins the game above with advantage at least* ϵ*.*

(Security Against Verifying Groups). This security notion requires that any user in a verifying group $\mathcal{B}$, even the manager of the group, should not be able to transform a partial signature into a full one with non-negligible probability if no help has been obtained from the signer in the signing group or the arbitrator. Formally, we consider the game G_{savg} depicted in Fig. 1. The advantage of $\mathcal{B}$ in the game, $\mathtt{Adv}^{\mathrm{SAVG}}_{\mathcal{B}}(k)$, is defined to be $\mathcal{B}$'s success probability in the game above.

Definition 2.5 (Security Against Verifying Groups)**.** *A GOFE scheme is said to be* $(t, q_p, q_j, q_t, q_r, \epsilon)$-secure against verifying groups *if there is no adversary which runs in time at most* t*, makes at most* q_p *queries to* $\mathcal{O}_{\mathsf{P}}$*,* q_j *queries to* $\mathcal{O}_{\mathsf{J}}$*,* q_t *queries to* $\mathcal{O}_{\mathsf{T}}$ *and* q_r *queries to* $\mathcal{O}_{\mathsf{R}}$*, and wins the game above with advantage at least* ϵ*.*

(Traceability). A GOFE scheme is treaceable if all valid full signature of a group, even those created by the collusion of multiple users and the group manager, trace to a member of the forging coalition. Formally, we consider the game G_{tr} depicted in Fig. 1. The advantage of $\mathcal{C}$ in this game, $\mathtt{Adv}^{\mathrm{Tr}}_{\mathcal{C}}(k)$, is defined to be its success probability.

Definition 2.6 (Traceability)**.** *A GOFE scheme is said to be* (t, q_p, q_j, ϵ)-traceable *if there is no adversary* $\mathcal{C}$ *which runs in time at most* t*, makes at most* q_p *queries to* $\mathcal{O}_{\mathsf{P}}$ *and* q_j *queries to* $\mathcal{O}_{\mathsf{J}}$*, and wins the game above with advantage at least* ϵ*.*

Remark 3 : Readers may notice that we do not define the security against the arbitrator explicitly, as traceability already covers it. On the other hand, as AOFE [17], both the signing group G_A and verifying group G_B are equipped with public/secret key pairs, and any user in them can generate indistinguishable partial signatures on the same message using the user secret keys issued by their respective group managers. If the traceability holds for group G_A (as described in game G_{tr}), it should also hold for group G_B.

Definition 2.7 (Secure GOFE)**.** *A GOFE scheme is said to be secure in the multi-user setting and chosen-key model, if it is group ambiguous, secure against signing groups, secure against verifying groups and traceable.*

Remark 4 : If a GOFE scheme is provably secure in the random oracle model, the adversaries in the games above have access to the random oracles. Also, we stress that all the definitions above are in the multi-user setting and chosen-key model. Namely, we allow the adversary to collude with other users in the system and use a public key without proving its knowledge of the secret key.

2.2 Relations Among The Properties

As mentioned above, group ambiguity considers that two users in two different groups can produce indistinguishable partial signatures on the same message, while user anonymity considers that two

Group Ambiguity, G_{ga}

$\mathsf{PM} \leftarrow \mathsf{PMGen}(1^k)$
$(\mathtt{apk}, \mathtt{ask}) \leftarrow \mathsf{Setup}^{\mathsf{TTP}}(\mathsf{PM})$
$(M, \{\mathtt{gpk}_d, \mathtt{gsk}_d, \mathtt{ID}_d\}_{d=0}^{1}, \Upsilon) \leftarrow \mathcal{D}^{\mathcal{O}_{\mathsf{R}}}(\mathsf{PM}, \mathtt{apk})$
$b \leftarrow \{0,1\}$
$\mathtt{usk}_b \leftarrow \mathsf{Join}(\mathtt{gsk}_b, \mathtt{ID}_b)$
$\sigma_P \leftarrow \mathsf{GPSig}(M, \mathtt{usk}_b, \mathtt{gpk}_b, \mathtt{gpk}_{1-b}, \mathtt{apk})$
$b' \leftarrow \mathcal{D}^{\mathcal{O}_{\mathsf{R}}}(\Upsilon, \sigma_P)$
success of $\mathcal{D} := \big[b' = b \wedge (M, \sigma_P, \{\mathtt{gpk}_0, \mathtt{gpk}_1\}) \notin \mathit{Query}(\mathcal{D}, \mathcal{O}_{\mathsf{R}})\big]$

User Anonymity, G_{ua}

$\mathsf{PM} \leftarrow \mathsf{PMGen}(1^k)$
$(\mathtt{apk}, \mathtt{ask}) \leftarrow \mathsf{Setup}^{\mathsf{TTP}}(\mathsf{PM})$
$(M, \{\mathtt{gpk}_d, \mathtt{gsk}_d, \mathtt{ID}_d\}_{d=0}^{1}, \Upsilon) \leftarrow \mathcal{D}^{\mathcal{O}_{\mathsf{R}}}(\mathsf{PM}, \mathtt{apk})$
$b \leftarrow \{0,1\}$
$\mathtt{usk}_b \leftarrow \mathsf{Join}(\mathtt{gsk}_0, \mathtt{ID}_b)$
$\sigma_P \leftarrow \mathsf{GPSig}(M, \mathtt{usk}_b, \mathtt{gpk}_0, \mathtt{gpk}_1, \mathtt{apk})$
$b' \leftarrow \mathcal{D}^{\mathcal{O}_{\mathsf{R}}}(\Upsilon, \sigma_P)$
success of $\mathcal{D} := [b' = b]$

Security Against Signing Groups, G_{sasg}

$\mathsf{PM} \leftarrow \mathsf{PMGen}(1^k)$
$(\mathtt{apk}, \mathtt{ask}) \leftarrow \mathsf{Setup}^{\mathsf{TTP}}(\mathsf{PM})$
$(\mathtt{gpk}_B, \mathtt{gsk}_B) \leftarrow \mathsf{Setup}^{\mathsf{Group}}(\mathsf{PM}, \mathtt{apk})$
$(M, \sigma_P, \mathtt{gpk}_A) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{P}}(B), \mathcal{O}_{\mathsf{J}}(B), \mathcal{O}_{\mathsf{T}}(B), \mathcal{O}_{\mathsf{R}}}(\mathsf{PM}, \mathtt{apk}, \mathtt{gpk}_B)$
$\sigma_F \leftarrow \mathsf{Res}(M, \sigma_P, \mathtt{ask}, \{\mathtt{gpk}_A, \mathtt{gpk}_B\})$
success of $\mathcal{A} := \big[\mathsf{GPVer}(M, \sigma_P, \{\mathtt{gpk}_A, \mathtt{gpk}_B\}, \mathtt{apk}) = 1 \wedge \mathsf{GVer}(M, \sigma_F, \mathtt{gpk}_A, \mathtt{gpk}_B, \mathtt{apk}) = 0 \wedge (M, \cdot, \mathtt{gpk}_A) \notin \mathit{Query}(\mathcal{A}, \mathcal{O}_{\mathsf{P}}(B))\big]$

Treaceability, G_{tr}

$\mathsf{PM} \leftarrow \mathsf{PMGen}(1^k)$
$(\mathtt{apk}, \Upsilon) \leftarrow \mathcal{C}(\mathsf{PM})$
$(\mathtt{gpk}_A, \mathtt{gsk}_A) \leftarrow \mathsf{Setup}^{\mathsf{Group}}(\mathsf{PM}, \mathtt{apk})$
$(M, \mathtt{gpk}_B, \sigma_F) \leftarrow \mathcal{C}^{\mathcal{O}_{\mathsf{P}}(A), \mathcal{O}_{\mathsf{J}}(A)}(\Upsilon, \mathtt{gpk}_A)$
$\mathtt{ID} \leftarrow \mathsf{Trace}(M, \sigma_P, \mathtt{gsk}_A, \mathtt{gpk}_A, \mathtt{gpk}_B, \mathtt{apk})$
success of $\mathcal{C} := \big[\mathsf{GVer}(M, \sigma_F, \mathtt{gpk}_A, \mathtt{gpk}_B, \mathtt{apk}) = 1 \wedge \mathtt{ID} \neq \perp \wedge \mathtt{ID} \notin \mathit{Query}(\mathcal{C}, \mathcal{O}_{\mathsf{J}}(A)) \wedge (M, \mathtt{ID}, \mathtt{gpk}_B) \notin \mathit{Query}(\mathcal{C}, \mathcal{O}_{\mathsf{P}}(A))\big]$

Security Against Verifying Groups, G_{savg}

$\mathsf{PM} \leftarrow \mathsf{PMGen}(1^k)$
$(\mathtt{apk}, \mathtt{ask}) \leftarrow \mathsf{Setup}^{\mathsf{TTP}}(\mathsf{PM})$
$(\mathtt{gpk}_A, \mathtt{gpk}_A) \leftarrow \mathsf{Setup}^{\mathsf{Group}}(\mathsf{PM}, \mathtt{apk})$
$(M, \mathtt{gpk}_B, \sigma_F) \leftarrow \mathcal{B}^{\mathcal{O}_{\mathsf{P}}(A), \mathcal{O}_{\mathsf{J}}(A), \mathcal{O}_{\mathsf{T}}(A), \mathcal{O}_{\mathsf{R}}}(\mathsf{PM}, \mathtt{gpk}_A, \mathtt{apk})$
success of $\mathcal{B} := \big[\mathsf{GVer}(M, \sigma_F, \mathtt{gpk}_A, \mathtt{gpk}_B, \mathtt{apk}) = 1 \wedge (M, \cdot, \{\mathtt{gpk}_A, \mathtt{gpk}_B\}) \notin \mathit{Query}(\mathcal{B}, \mathcal{O}_{\mathsf{R}})\big]$

Notations: Let I, $\mathcal{F}$ take value from $\{A, B\}$, $\{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}\}$ respectively, and $\mathcal{O}$ from $\{\mathcal{O}_{\mathsf{P}}(I), \mathcal{O}_{\mathsf{J}}(I), \mathcal{O}_{\mathsf{T}}(I), \mathcal{O}_{\mathsf{R}}\}$.
$\mathcal{O}_{\mathsf{P}}(I)$ takes as input $(M, \mathtt{ID}, \mathtt{gpk}_i)$ and outputs a partial signature on M under $\{\mathtt{gpk}_I, \mathtt{gpk}_i\}$ generated using the secret key of user $\mathtt{ID}$ derived from $\mathtt{gsk}_I$.
$\mathcal{O}_{\mathsf{J}}(I)$ takes as input $\mathtt{ID}$ and outputs the secret key of user $\mathtt{ID}$ derived from $\mathtt{gsk}_I$.
$\mathcal{O}_{\mathsf{T}}(I)$ takes as input $(M, \sigma_F, \mathtt{gpk}_i)$ and outputs nothing if $0 \leftarrow \mathsf{GVer}(M, \sigma_F, \mathtt{gpk}_I, \mathtt{gpk}_i, \mathtt{apk})$, $\perp$ for failure, or $\mathtt{ID}$.
$\mathcal{O}_{\mathsf{R}}$ takes as input $(M, \sigma_P, \{\mathtt{gpk}_i, \mathtt{gpk}_j\})$ and outputs $\perp$ indicating failure, or σ_F otherwise.
$\mathit{Query}(\mathcal{F}, \mathcal{O})$ is the set of queries submitted by the adversary $\mathcal{F}$ to the oracle $\mathcal{O}$. Υ is the adversary's state information.

Figure 1: Security Models of GOFE

users in the same group produce indistinguishable signatures. We have the following lemma, the proof of which is deferred to Appendix A.

Lemma 2.8. *If a GOFE scheme is (t, q_r, ϵ)-group ambiguous, it is also (t', q_r, ϵ')-user anonymous, where $t' \approx t$ and $\epsilon' \leq \epsilon$.*

On the other hand, the security against the verifying groups indicates that a verifying group (manager) cannot convert the signing group's partial signature into a full one without help from the signing group and the arbitrator. Intuitively, if someone (in the verifying group) is able to convert a partial signature from the signing group, it can also convince others before the end of a transaction that someone in the signing group committed to some message. This contradicts the intuition behind the definition of group ambiguity.

As [17] we define a weak version of security against verifying groups, in which the game challenger chooses $(\mathtt{gpk}_A, \mathtt{gsk}_A)$ and $(\mathtt{gpk}_B, \mathtt{gsk}_B)$ and gives $(\mathtt{gpk}_A, \mathtt{gpk}_B, \mathtt{gsk}_B)$ together with $\mathtt{apk}$ to the adversary $\mathcal{B}$, and the full signature output by $\mathcal{B}$ should be valid with respect to the given $\mathtt{gpk}_A$ and $\mathtt{gpk}_B$. We call this version of security as '*weak security against verifying groups*'. Similarly, we have the following lemma. The proof of the lemma is almost the same as that in [17] except some minor modifications, so we omit it here.

Lemma 2.9. *If a GOFE scheme is group ambiguous and traceable, it is also weakly secure against verifying groups. Specifically, if a GOFE scheme is (t_1, q_r, ϵ_1)-group ambiguous and $(t_2, q_p, q_j, \epsilon_2)$-traceable, it is also $(t', q'_p, q'_j, q'_t, q'_r, \epsilon')$-weakly secure against verifying groups, where $t' \approx t_1$, $q'_p \leq q_p$, $q'_j \leq q_j$, $q'_r \leq q_r$ and $\epsilon' \leq 2\epsilon_2 + 2q_p^2\epsilon_1$.*

3 Mathematical Background

(Admissible Pairings). Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups of large prime order p. The mapping $\mathtt{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is said to be an *admissible pairing*, if

Bilinearity $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$, $\mathtt{e}(u^a, v^b) = \mathtt{e}(u, v)^{ab}$;
Non-degeneracy $\exists u, v \in \mathbb{G}$ such that $\mathtt{e}(u, v) \neq 1_T$, where 1_T is the identity element of $\mathbb{G}_T$; and
Computability there exists an efficient algorithm for computing $\mathtt{e}(u, v)$ for any $u, v \in \mathbb{G}$.

(**Decision Linear Assumption (DLN)** [6]): Let $\mathbb{G}$ be a cyclic group of large prime order p. The linear problem is defined as follows: given $u, v, h, u^\alpha, v^\beta \in \mathbb{G}$, where $\alpha, \beta \leftarrow \mathbb{Z}_p$, output $h^{\alpha+\beta}$. The DLN assumption (t, ϵ) holds in $\mathbb{G}$, if there is no adversary $\mathcal{A}$ which runs in time at most t, and

$$\left| \Pr[u, v, h \leftarrow \mathbb{G}; \alpha, \beta \leftarrow \mathbb{Z}_p; Z_0 \leftarrow h^{\alpha+\beta}; Z_1 \xleftarrow{\$} \mathbb{G}; b \leftarrow \{0,1\}; b' \leftarrow \mathcal{A}(u, v, h, u^\alpha, v^\beta, Z_b) : b' = b] - \frac{1}{2} \right| > \epsilon$$

where the probability is taken over the random choices of $u, v, h, Z_1 \in \mathbb{G}$, $\alpha, \beta \in \mathbb{Z}_p$, and the random coins consumed by $\mathcal{A}$.

(q-**Strong Diffie-Hellman Assumption (q-SDH)** [5]): The q-SDH problem in $\mathbb{G}$ is defined as follows: given a $(q+1)$-tuple $(g, g^\gamma, g^{\gamma^2}, \cdots, g^{\gamma^q})$, output a pair $(g^{1/(\gamma+x)}, x)$ where $x \in \mathbb{Z}_p^*$. The q-SDH assumption (t, ϵ) holds in $\mathbb{G}$, if there is no adversary $\mathcal{A}$ which runs in time at most t and

$$\Pr\left[\gamma \leftarrow \mathbb{Z}_p^* : \mathcal{A}\left(g, g^\gamma, \cdots, g^{\gamma^q}\right) = \left(g^{\frac{1}{\gamma+x}}, x\right)\right] > \epsilon$$

where the probability is taken over the random choices of $g \in \mathbb{G}$, $\gamma \in \mathbb{Z}_p^*$, and the random coins consumed by $\mathcal{A}$.

4 A Zero-Knowledge Protocol for A Variant of SDH Problem

In this section we present a zero-knowledge proof of knowledge system for a special problem related to the SDH assumption, called *Double-SDH* problem.

> **Double-SDH Problem**. Given $p, \mathbb{G}, \mathbb{G}_T, g, \mathsf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and $\Gamma_0, \Gamma_1 \in \mathbb{G}$ $(\Gamma_0 \neq \Gamma_1)$, find a pair $(A, x) \in \mathbb{G} \times \mathbb{Z}_p^*$ such that either $\mathsf{e}(A, \Gamma_0 g^x) = \mathsf{e}(g, g)$ or $\mathsf{e}(A, \Gamma_1 g^x) = \mathsf{e}(g, g)$ holds.

Below we present a zero-knowledge proof of knowledge protocol for the Double-SDH problem, which does not reveal the values of A and x, and which equation is satisfied by them. In the protocol the prover proves its knowledge of a solution to the Double-SDH problem.

4.1 The Basic Zero-Knowledge Protocol

The public values are $g, u, v, h, U, V, H, \Gamma_0, \Gamma_1$, where g is a random generator of $\mathbb{G}$, u, v, h, U, V, H are randomly chosen from $\mathbb{G}$, and Γ_b equals g^{γ_b} for some secret $\gamma_b \in \mathbb{Z}_p$ for $b = 0, 1$. The protocol is a generalization of Schnorr's protocol for proving knowledge of discrete logarithm in a group of known order [21].

Assume that Alice, the prover, holds a pair (A, x) such that $\mathsf{e}(A, \Gamma_b \cdot g^x) = \mathsf{e}(g, g)$ for some bit $b \in \{0, 1\}$. She randomly selects exponents $\alpha, \beta, \alpha', \beta' \leftarrow \mathbb{Z}_p$ and computes the following.

$$T_1 \leftarrow u^\alpha \quad T_2 \leftarrow v^\beta \quad T_3 \leftarrow A \cdot h^{\alpha+\beta} \quad S_1 \leftarrow U^{\alpha'} \quad S_2 \leftarrow V^{\beta'} \quad S_3 \leftarrow \Gamma_b \cdot H^{\alpha'+\beta'} \tag{1}$$

Alice sends $(T_1, T_2, T_3, S_1, S_2, S_3)$ to Bob, the verifier, and then interacts with him to carry out a proof of knowledge of values $(x, \alpha, \beta, \alpha', \beta')$ satisfying the following statement:

$$\begin{aligned} u^\alpha = T_1 \ \wedge\ v^\beta = T_2 \ \wedge\ U^{\alpha'} = S_1 \ \wedge\ V^{\beta'} = S_2 \ \wedge\ \mathsf{e}(T_3 \cdot h^{-\alpha-\beta}, S_3 \cdot H^{-\alpha'-\beta'} \cdot g^x) = \mathsf{e}(g, g) \\ \wedge\ \left(H^{\alpha'+\beta'} = S_3/\Gamma_0 \ \vee\ H^{\alpha'+\beta'} = S_3/\Gamma_1\right) \end{aligned} \tag{2}$$

For brevity, let C be the first line of the statement above, and D_0, D_1 be the two components of the OR relation respectively. Then statement (2) can be expressed as $C \wedge (D_0 \vee D_1)$. To prove it, it is equivalent to prove $(C \wedge D_0) \vee (C \wedge D_1)$. Below we first give a protocol for proving $C \wedge D_b$, and then show how to extend it to prove the whole statement, i.e. $(C \wedge D_b) \vee (C \wedge D_{1-b})$.

Protocol 1. Without loss of generality, we assume that $b = 0$. The proof of knowledge proceeds as follows. Alice first computes the following helper values:

$$\delta_1 \leftarrow x\alpha, \quad \delta_2 \leftarrow x\beta, \quad \delta_3 \leftarrow \alpha\alpha', \quad \delta_4 \leftarrow \alpha\beta', \quad \delta_5 \leftarrow \beta\alpha', \quad \delta_6 \leftarrow \beta\beta'$$

She then selects $r_x, r_\alpha, r_\beta, r_{\alpha'}, r_{\beta'}, r_1, r_2, r_3, r_4, r_5, r_6$ at random from $\mathbb{Z}_p$, and computes the following values:

$$\begin{gathered} R_1 \leftarrow u^{r_\alpha}, \quad R_2 \leftarrow v^{r_\beta}, \quad R_3 \leftarrow U^{r_{\alpha'}}, \quad R_4 \leftarrow V^{r_{\beta'}}, \quad R_5 \leftarrow u^{-r_1} T_1^{r_x}, \quad R_6 \leftarrow v^{-r_2} T_2^{r_x}, \\ R_7 \leftarrow U^{-r_3} S_1^{r_\alpha}, \quad R_8 \leftarrow V^{-r_4} S_2^{r_\alpha}, \quad R_9 \leftarrow U^{-r_5} S_1^{r_\beta}, \quad R_{10} \leftarrow V^{-r_6} S_2^{r_\beta}, \quad R_{11} \leftarrow H^{r_{\alpha'}+r_{\beta'}}, \\ R_{12} \leftarrow \mathsf{e}(T_3, H)^{-r_{\alpha'}-r_{\beta'}}\ \mathsf{e}(T_3, g)^{r_x}\ \mathsf{e}(h, S_3)^{-r_\alpha-r_\beta}\ \mathsf{e}(h, H)^{r_3+r_4+r_5+r_6}\ \mathsf{e}(h, g)^{-r_1-r_2}. \end{gathered}$$

Alice sends all these values, i.e. $(R_1, R_2, \cdots, R_{12})$, to Bob, which then returns a challenge value c chosen uniformly at random from $\mathbb{Z}_p$. Alice then computes the following response values:

$$\begin{gathered} s_x \leftarrow r_x + cx, \quad s_\alpha \leftarrow r_\alpha + c\alpha, \quad s_\beta \leftarrow r_\beta + c\beta, \quad s_{\alpha'} \leftarrow r_{\alpha'} + c\alpha', \quad s_{\beta'} \leftarrow r_{\beta'} + c\beta', \\ s_i \leftarrow r_i + c\delta_i \qquad \text{for} \quad i = 1, 2, \cdots, 6 \end{gathered}$$

and sends them back. Bob accepts if all the equations below hold, and rejects otherwise.

$$u^{s_\alpha} = T_1^c \cdot R_1 \quad (3) \qquad V^{s_{\beta'}} = S_2^c \cdot R_4 \quad (6) \qquad u^{-s_1} T_1^{s_x} = R_5 \quad (7) \qquad V^{-s_4} S_2^{s_\alpha} = R_8 \quad (10)$$

$$v^{s_\beta} = T_2^c \cdot R_2 \quad (4) \qquad v^{-s_2} T_2^{s_x} = R_6 \quad (8) \qquad U^{-s_5} S_1^{s_\beta} = R_9 \quad (11)$$

$$U^{s_{\alpha'}} = S_1^c \cdot R_3 \quad (5) \qquad U^{-s_3} S_1^{s_\alpha} = R_7 \quad (9) \qquad V^{-s_6} S_2^{s_\beta} = R_{10} \quad (12)$$

$$H^{s_{\alpha'}+s_{\beta'}} = (S_3/\Gamma_0)^c \cdot R_{11} \quad (13)$$

$$\mathsf{e}(T_3, H)^{-s_{\alpha'}-s_{\beta'}} \mathsf{e}(T_3, g)^{s_x} \mathsf{e}(h, S_3)^{-s_\alpha - s_\beta} \mathsf{e}(h, H)^{s_3+s_4+s_5+s_6} \mathsf{e}(h, g)^{-s_1-s_2} = \left(\frac{\mathsf{e}(g,g)}{\mathsf{e}(T_3, S_3)}\right)^c R_{12} \quad (14)$$

Regarding the security of Protocol 1, we have the following theorem. The proof is given in Appendix B.

Theorem 4.1. *Protocol 1 satisfies completeness, perfect special honest-verifier zero knowledge, and special soundness.*

Protocol 2. Protocol 1 proves $C \wedge D_0$. The same protocol applies to $C \wedge D_1$ when $b = 1$ (i.e. $\mathsf{e}(A, \Gamma_1 g^x) = \mathsf{e}(g, g)$) by replacing Γ_0 with Γ_1 in the proof. Now we show how to extend it to the full protocol that proves the whole statement (2).

Alice, the prover, is given a pair (A, x) such that $\mathsf{e}(A, \Gamma_b g^x) = \mathsf{e}(g, g)$ for some bit b. She first picks at random $\alpha, \beta, \alpha', \beta' \in \mathbb{Z}_p$, and generates $(T_1, T_2, T_3, S_1, S_2, S_3, R_1^b, \cdots, R_{12}^b)$ as in Protocol 1, except that S_3 is computed as $S_3 \leftarrow \Gamma_b H^{\alpha'+\beta'}$. She also randomly picks a challenge value $c^{1-b} \in \mathbb{Z}_p$, and calls the simulator of Protocol 1 to produce a transcript for the statement $\mathsf{e}(A, \Gamma_{1-b} g^x) = \mathsf{e}(g, g)$. Let the transcript be $(R_1^{1-b}, \cdots, R_{12}^{1-b}, c^{1-b}, s_x^{1-b}, s_\alpha^{1-b}, s_\beta^{1-b}, s_{\alpha'}^{1-b}, s_{\beta'}^{1-b}, s_1^{1-b}, \cdots, s_6^{1-b})$. Alice sends $(T_1, T_2, T_3, S_1, S_2, S_3, R_1^0, \cdots, R_{12}^0, R_1^1, \cdots, R_{12}^1)$ to Bob, the verifier, which then returns a challenge value $c \in \mathbb{Z}_p$. Alice first computes $c^b \leftarrow c - c^{1-b}$, and then generates response values $s_x^b, s_\alpha^b, s_\beta^b, s_{\alpha'}^b, s_{\beta'}^b, s_1^b, \cdots, s_6^b$ as in Protocol 1. She responds back $(c^0, s_x^0, s_\alpha^0, s_\beta^0, s_{\alpha'}^0, s_{\beta'}^0, s_1^0, \cdots, s_6^0, c^1, s_x^1, s_\alpha^1, s_\beta^1, s_{\alpha'}^1, s_{\beta'}^1, s_1^1, \cdots, s_6^1)$. After receiving them, Bob first checks if $c \stackrel{?}{=} c^0 + c^1$. If not, he rejects; otherwise, he does as in Protocol 1 to check the validity of $(R_1^0, \cdots, R_{12}^0, c^0, s_x^0, s_\alpha^0, s_\beta^0, s_{\alpha'}^0, s_{\beta'}^0, s_1^0, \cdots, s_6^0)$ and $(R_1^1, \cdots, R_{12}^1, c^1, s_x^1, s_\alpha^1, s_\beta^1, s_{\alpha'}^1, s_{\beta'}^1, s_1^1, \cdots, s_6^1)$ with respect to $(T_1, T_2, T_3, S_1, S_2, S_3)$ respectively. If either fails, he rejects; otherwise he accepts.

Theorem 4.2. *Protocol 2 is an honest-verifier zero-knowledge proof of knowledge system for the Double-SDH problem.*

The proof is deferred to Appendix B, where we show that Protocol 2 is simulatable, and has special soundness, i.e. there is an extractor which can extract (A, x) and a bit b such that statement (2) is satisfied. We stress that besides (A, x) and the bit b, the extractor can also output the randomness used in the two encryptions, i.e. $(\alpha, \beta, \alpha', \beta')$. Implied by the special soundness, the probability that the pair (A, x) contained in (T_1, T_2, T_3) does not satisfy statement (2) for any bit b is then upper bounded by $1/p$ (see [13]).

4.2 Signature of Knowledge for Double-SDH

By applying a variant of the standard Fiat-Shamir heuristic to Protocol 2 we can obtain a signature of knowledge for statement (2). Namely, after generating all the R values, the signer applies a (collision-resistant) hash function $\mathtt{H}_1 : \{0,1\}^* \times \mathbb{G}^{13} \times \mathbb{G}_T \times \mathbb{G}^{11} \times \mathbb{G}_T \rightarrow \mathbb{Z}_p^*$ to the message M to be signed, Γ_0, Γ_1, and these R values, and obtains a challenge value c. It then computes the response values with respect to c using its secret key. A signature θ then consists of

$(c^0, s_x^0, s_\alpha^0, s_\beta^0, s_{\alpha'}^0, s_{\beta'}^0, s_1^0, \cdots, s_6^0, c^1, s_x^1, s_\alpha^1, s_\beta^1, s_{\alpha'}^1, s_{\beta'}^1, s_1^1, \cdots, s_6^1) \in \mathbb{Z}_p^{24}$. To verify a signature, one computes the following for $b = 0, 1$:

$$R_1^b \leftarrow u^{s_\alpha^b}/T_1^{c^d}, \quad R_2^b \leftarrow v^{s_\beta^b}/T_2^{c^b}, \quad R_3^b \leftarrow U^{s_{\alpha'}^b}/S_1^{c^b}, \quad R_4^b \leftarrow V^{s_{\beta'}^b}/S_2^{c^b}, \quad R_5^b \leftarrow u^{-s_1^b} T_1^{s_x^b},$$
$$R_6^b \leftarrow v^{-s_2^b} T_2^{s_x^b}, \quad R_7^b \leftarrow U^{-s_3^b} S_1^{s_\alpha^b}, \quad R_8^b \leftarrow V^{-s_4^b} S_2^{s_\alpha^b}, \quad R_9^b \leftarrow U^{-s_5^b} S_1^{s_\beta^b}, \quad R_{10}^b \leftarrow V^{-s_6^b} S_2^{s_\beta^b},$$
$$R_{11}^b \leftarrow \frac{H^{s_{\alpha'}^b + s_{\beta'}^b}}{(S_3/\Gamma_b)^{c^b}}, \quad R_{12}^b \leftarrow \frac{\mathsf{e}(T_3, H)^{-s_{\alpha'}^b - s_{\beta'}^b} \mathsf{e}(T_3, g)^{s_x^b} \mathsf{e}(h, S_3)^{-s_\alpha^b - s_\beta^b} \mathsf{e}(h, H)^{s_3^b + s_4^b + s_5^b + s_6^b} \mathsf{e}(h, g)^{-s_1^b - s_2^b}}{(\mathsf{e}(g, g)/\mathsf{e}(T_3, S_3))^{c^b}},$$

and then checks if $c^0 + c^1 \stackrel{?}{=} \mathtt{H}_1(\bar{M}, R_1^0, R_2^0, \cdots R_{12}^0, R_1^1, R_2^1, \cdots, R_{12}^1)$, where $\bar{M} = M \| \Gamma_0 \| \Gamma_1$. If not, the verifier rejects; otherwise it accepts.

5 The Group-oriented OFE scheme

In this section we propose our efficient construction of GOFE scheme based on a zero-knowledge protocol slightly modified from Protocol 2 given in Sec. 4. The zero-knowledge protocol uses two more parameters, i.e. $K, L \in \mathbb{G}$, and a collision-resistant hash function $\mathtt{H}_3 : \mathbb{G}^4 \to \mathbb{Z}_p$. The proof remains the same, except that the encryption of Γ_b is computed as

$$S_1 \leftarrow U^{\alpha'}, \quad S_2 \leftarrow V^{\beta'}, \quad S_3 \leftarrow \Gamma_b \cdot H^{\alpha' + \beta'}, \quad S_4 \leftarrow (H^\chi K)^{\alpha'}, \quad S_5 \leftarrow (H^\chi L)^{\beta'} \tag{15}$$

where $\chi \leftarrow \mathtt{H}_3(S_1, S_2, \Gamma_0, \Gamma_1)$. Given a proof, the verifier checks not only the validity of θ with respect $(T_1, T_2, T_3, S_1, S_2, S_3)$, but also checks if $\mathsf{e}(S_1, H^\chi K) \stackrel{?}{=} \mathsf{e}(U, S_4)$ and $\mathsf{e}(S_2, H^\chi L) \stackrel{?}{=} \mathsf{e}(V, S_5)$. If either check fails, it rejects the proof; otherwise it accepts. The modified encryption of Γ_b is from Kiltz' (selective-tag CCA-secure) tag-based public key encryption scheme [19]. The introduction of the two extra parameters enables us to simulate the resolution oracle in the CCA type. See the security proofs for more details. Here we note that the signer does not need to prove the knowledge of the discrete logarithms of S_4, S_5 with respect to the bases $H^\chi K$ and $H^\chi L$ respectively, as the two equations $\mathsf{e}(S_1, H^\chi K) = \mathsf{e}(U, S_4)$ and $\mathsf{e}(S_2, H^\chi L) = \mathsf{e}(V, S_5)$ already implies that the signer knows their discrete logarithms, which are the same as $\log_U S_1$ and $\log_V S_2$ respectively.

It's not hard to see that the modified protocol is still honest-verifier zero-knowledge and $(A, x, \alpha, \beta, \alpha', \beta', \Gamma_b)$ can be extracted from two protocol transcripts sharing the same R's. The signature of knowledge version of the modified protocol can be obtained in the same way as in Sec. 4.2. Below we propose our construction of GOFE.

PMGen : The algorithm takes as input 1^k and outputs two multiplicative groups of prime order p, $\mathbb{G}$ and $\mathbb{G}_T$, a random generator g of $\mathbb{G}$, and an admissible bilinear pairing $\mathsf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. It also selects three collision-resistant hash functions, i.e. $\mathtt{H}_1 : \{0,1\}^* \times \mathbb{G}^{19} \times \mathbb{G}_T \times \mathbb{G}^{11} \times \mathbb{G}_T \to \mathbb{Z}_p$, $\mathtt{H}_2 : \{0,1\}^* \times \mathbb{G}^{16} \times \mathbb{Z}_p^{24} \times \mathbb{G}^6 \to \mathbb{Z}_p$, and $\mathtt{H}_3 : \mathbb{G}^{10} \to \mathbb{Z}_p$, which will be modeled as random oracles in the security proofs.

$\mathsf{Setup}^{\mathsf{TTP}}$: The arbitrator randomly selects U, V and H such that $U^{\xi_1} = V^{\xi_2} = H$ for some $(\xi_1, \xi_2) \in \mathbb{Z}_p^2$. It also picks at random $K, L \in \mathbb{G}$. The public/private key pair is then set to be $(\mathtt{apk}, \mathtt{ask}) := ((U, V, H, K, L), (\xi_1, \xi_2))$.

$\mathsf{Setup}^{\mathsf{Group}}$: Each group manager picks at random $\gamma \leftarrow \mathbb{Z}_p^*$, and sets $\Gamma \leftarrow g^\gamma$. It also chooses $u, v, h \in \mathbb{G}$ at random satisfying that $u^{\nu_1} = v^{\nu_2} = h$ for some known $(\nu_1, \nu_2) \in \mathbb{Z}_p^2$. The group public/private key pair is set to be $(\mathtt{gpk}, \mathtt{gsk}) := ((\Gamma, u, v, h), (\gamma, \nu_1, \nu_2))$.

Join : When a user (with identity $\mathtt{ID}$) joins a group G (with public key $\mathtt{gpk} = \Gamma$), the group manager selects $x \leftarrow \mathbb{Z}_p^*$ at random and sets $A \leftarrow g^{1/(\gamma + x)}$. It returns the user secret key $\mathtt{usk} := (A, x)$ to the user, and stores $(\mathtt{ID}, A, x)$ in its database.

GPSig : Let M be the message to be partially signed, and the signer be in group G_i (with public key $\mathtt{gpk}_i = (\Gamma_i, u_i, v_i, h_i)$) and be with private key $\mathtt{usk} = (A, x)$, and the verifying group be group G_j (with public key $\mathtt{gpk}_j = (\Gamma_j, u_j, v_j, h_j)$). The signer generates a partial signature σ_P on M as below.

1. Select $\alpha, \beta, \alpha', \beta' \in \mathbb{Z}_p$ at random, and computes T_1, T_2, T_3 as in (1) with u, v, h replaced with u_i, v_i, h_i, and S_1, S_2, S_3, S_4, S_5 as in (15). Note that (T_1, T_2, T_3) is the linear encryption of A under group G_i's public key, and $(S_1, S_2, S_3, S_4, S_5)$ is the encryption of Γ_i under the arbitrator's public key.
2. Use $(x, \alpha, \beta, \alpha', \beta')$ to produce the signature θ on the message $\bar{M} = M \| \mathtt{gpk}_i \| \mathtt{gpk}_j$ as specified in Sec. 4.2, i.e. $\theta = (c^0, s_x^0, s_\alpha^0, s_\beta^0, s_{\alpha'}^0, s_{\beta'}^0, s_1^0, \cdots, s_6^0, c^1, s_x^1, s_\alpha^1, s_\beta^1, s_{\alpha'}^1, s_{\beta'}^1, s_1^1, \cdots, s_6^1)$. Note that in the generation of σ_P, hash function $\mathtt{H}_1$ is used for converting Protocol 2 to a signature, and that the signature of knowledge above is a little bit different from the one in Sec. 4, as we include $\mathtt{gpk}_i, \mathtt{gpk}_j$ and $\mathtt{apk}$ additionally into the message to be signed. The partial signature is then set to be $\sigma_P := (T_1, T_2, T_3, S_1, S_2, S_3, S_4, S_5, \theta)$. The signer stores (α', β') for later use, i.e. for converting σ_P to a full signature.

GPVer : Given a partial signature $\sigma_P = (T_1, T_2, T_3, S_1, S_2, S_3, S_4, S_5, \theta)$, a message M, group public keys $\mathtt{gpk}_i$, $\mathtt{gpk}_j$, and the arbitrator's public key $\mathtt{apk}$, the verifier calls the verification algorithm of the signature of knowledge at the beginning of Sec. 5 to check the validity of σ_P. Namely, the verifier checks both the validity of θ with respect to $(T_1, T_2, T_3, S_1, S_2, S_3)$ and if $\mathtt{e}(S_1, H^\chi K) = \mathtt{e}(U, S_4)$ and $\mathtt{e}(S_2, H^\chi L) = \mathtt{e}(V, S_5)$ both hold, where $\chi \leftarrow \mathtt{H}_3(S_1, S_2, \mathtt{gpk}_i, \mathtt{gpk}_j)$, and rejects if either check fails.

GSig : To (fully) sign a message M with the verifying group G_j, the signer in group G_i does as below:

1. use its secret key (A, x) to compute a partial signature $\sigma_P = (T_1, T_2, T_3, S_1, S_2, S_3, S_4, S_5, \theta)$ by running the GPSig algorithm;
2. provide the following signature of knowledge ϑ, showing that $(S_1, S_2, S_3, S_4, S_5)$ is indeed an encryption of Γ_i, or it can be decrypted to Γ_i by the arbitrator.

$$\vartheta \leftarrow SPK\big\{(\eta_1, \eta_2) : \big(U^{\eta_1} = S_1 \ \wedge \ V^{\eta_2} = S_2 \ \wedge \ H^{\eta_1+\eta_2} = S_3/\Gamma_i\big) \ \vee \\ \big(U^{\eta_1} = H \ \wedge \ V^{\eta_2} = H \ \wedge \ S_1^{\eta_1} \cdot S_2^{\eta_2} = S_3/\Gamma_i\big)\big\}(M \| \mathtt{gpk}_i \| \mathtt{gpk}_j \| \sigma_P) \quad (16)$$

Note that in the generation of ϑ hash function $\mathtt{H}_2$ is used, and the witness used by the signer is the randomness used in generation of the encryption, i.e. (α', β'). The details can be found in Appendix C. The full signature on M is then set to be $\sigma_F := (\sigma_P, \Gamma_i, \vartheta)$.

GVer : Given a full signature $\sigma = (\sigma_P, \Gamma, \vartheta)$, the verifier checks both the validity of σ_P by calling the GPVer algorithm and that of ϑ. It rejects if either check fails, and accepts otherwise.

Res : Given a partial signature $\sigma_P = (T_1, T_2, T_3, S_1, S_2, S_3, S_4, S_5, \theta)$, a message M and public keys Γ_i, Γ_j and $\mathtt{apk}$, the arbitrator first checks the validity of σ_P by invoking the GPVer algorithm. If it's invalid, the arbitrator returns $\perp$; otherwise, it decrypts $(S_1, S_2, S_3, S_4, S_5)$ using its secret key $\mathtt{ask} = (\xi_1, \xi_2)$. Namely, it checks if $\mathtt{e}(S_1, H^\chi K) = \mathtt{e}(U, S_4)$ and $\mathtt{e}(S_2, H^\chi L) = \mathtt{e}(V, S_5)$, where $\chi \leftarrow \mathtt{H}_3(S_1, S_2, \Gamma_i, \Gamma_j)$. If not, it returns $\perp$; otherwise, it computes $\Gamma \leftarrow S_3/(S_1^{\xi_1} \cdot S_2^{\xi_2})$. If Γ is equal to neither Γ_i nor Γ_j, the arbitrator returns $\perp$; otherwise, it computes a signature of knowledge ϑ of (16) using its knowledge of (ξ_1, ξ_2). It returns $(\sigma_P, \Gamma, \vartheta)$.

Trace : Given a *valid* group G_i's full signature $\sigma_F = ((T_1, T_2, T_3, S_1, S_2, S_3, S_4, S_5, \theta), \Gamma_i, \vartheta)$, the manager of G_i who holds the group secret key $\mathtt{gsk}_i = (\gamma_i, \nu_{i1}, \nu_{i2})$ computes $A \leftarrow T_3/T_1^{\nu_1} T_2^{\nu_2}$ and searches its database for a tuple $(\mathtt{ID}, A, x)$. If found, it outputs $\mathtt{ID}$; otherwise, it outputs $\perp$.

Remark 5 : The signature of knowledge ϑ is obtained by applying a variant of Fiat-Shamir heuristic to the corresponding three-move zero-knowledge proof of knowledge protocol which is given in Appendix C. We stress that implied by the special soundness of the protocol, the probability that the witness used in the generation of the proof does not satisfy (16) is upper bounded by $1/p$. (see [13]).

We also remark that though the trace algorithm takes the whole group secret key as input, it only uses ν_1, ν_2. Therefore, in practice, the group secret key can be given to two person. One holds γ and is responsible for user enrollment, while the other holds ν_1, ν_2 and is responsible for tracing.

Efficiency. The following table shows the comparison of our GOFE with AOFE of [17] in the size of arbitrator's public key, group/user public key, partial signature and full signature. In Fig. 2, pk is

	$\lvert\mathtt{apk}\rvert$	$\lvert\mathtt{ask}\rvert$	$\lvert pk\rvert$	$\lvert sk\rvert$	$\lvert\mathtt{usk}\rvert$	$\lvert\sigma_P\rvert$	$\lvert\sigma_F\rvert$
GOFE	$5\mathbb{G}$	$2\mathbb{Z}_p$	$4\mathbb{G}$	$3\mathbb{Z}_p$	$1\mathbb{G} + 1\mathbb{Z}_p$	$8\mathbb{G} + 24\mathbb{Z}_p$	$9\mathbb{G} + 30\mathbb{Z}_p$
AOFE	$10\mathbb{G}$	$2\mathbb{Z}_p$	$1\mathbb{G}$	$1\mathbb{Z}_p$	n/a	$41\mathbb{G} + \lvert otvk\rvert + \lvert\sigma_{ot}\rvert$	$42\mathbb{G} + \lvert otvk\rvert + \lvert\sigma_{ot}\rvert$

Figure 2: Comparison with AOFE [17]

the group public key in GOFE, and is the public key of a user in AOFE; 'n/a' means that it is not applicable; $otvk$ is a one-time verification key and σ_{ot} is a one-time signature.

If we instantiate the one-time signature scheme with Boneh-Boyen short signature [5], the size of a partial signature of AOFE is $45\mathbb{G} + 1\mathbb{Z}_p$, and that of a full signature is $46\mathbb{G} + 1\mathbb{Z}_p$. As suggested by Cheon [12], we use 220-bit prime p for 80-bit symmetric security level. If we use a pairing with embedding degree 6, the partial signature size is $8 \times 221 + 24 \times 220 = 7048$ bits, and the size of a full signature is $9 \times 221 + 30 \times 220 = 8588$ bits, while the size of a partial signature and that of a full signature of AOFE are 10165 bits and 10386 bits, respectively. Though the (partial and full) signing algorithms involve dozens of (multi-)exponentiation operations, their costs are still constant, i.e. the number of total operations are independent of the numbers of users in the signing group and users in the verifying group, thus appropriate for practical use. We note that the disadvantage of our work when compared with [17] is that our scheme is provably secure in the random oracle model (as shown below), while their AOFE scheme is secure in the standard model.

Security. For the security, we have the following theorem. The proof is deferred to Appendix D.

Theorem 5.1. *Under the DLN assumption and SDH assumption, our construction above is a secure GOFE scheme in the multi-user setting and chosen-key model (Def. 2.7) in the random oracle model.*

6 Generic Construction of AOFE from GOFE

As mentioned before, the difference between AOFE and GOFE is that the former focuses on the setting in which the two parties (except the arbitrator) involved in a transaction are individual users, while the latter focuses on the setting in which each party (again, except the arbitrator) consists of a group of users. Therefore, GOFE is *backward compatible* with AOFE. That is, we can use GOFE to build an AOFE scheme in a very natural way. Let GOFE be a GOFE scheme, the generic construction of AOFE, denoted AOFE, is depicted in Fig. 3.

Theorem 6.1. *If* GOFE *is a secure GOFE scheme (see Def. 2.7),* AOFE *constructed above is also a secure AOFE scheme (see Def. 6 of* [17]*).*

The proof of the theorem above follows from the definitions of the security properties naturally, so we omit it here. Using the efficient construction of GOFE presented in Sec. 5 to instantiate the generic transform above, we thus obtain an AOFE scheme that is even more efficient than the one

$\mathsf{PMGen}(1^k)$: return $\mathsf{PM} \leftarrow \mathsf{GOFE.PMGen}(1^k)$	$\mathsf{Setup}^{\mathsf{TTP}}(\mathsf{PM})$: return $(\mathtt{apk}, \mathtt{ask}) \leftarrow \mathsf{GOFE.Setup}^{\mathsf{TTP}}(\mathsf{PM})$
$\mathsf{Res}(M, \sigma_P, \mathtt{ask}, \{\mathtt{pk}_i, \mathtt{pk}_j\})$: return $\mathsf{GOFE.Res}(M, \sigma_P, \mathtt{ask}, \{\mathtt{pk}_i, \mathtt{pk}_j\})$	$\mathsf{Setup}^{\mathsf{User}}(\mathsf{PM}, \mathtt{apk})$: $(\mathtt{gpk}, \mathtt{gsk}) \leftarrow \mathsf{GOFE.Setup}^{\mathsf{Group}}(\mathsf{PM}, \mathtt{apk})$ return $(\mathtt{pk}, \mathtt{sk}) := (\mathtt{gpk}, \mathtt{gsk})$
$\mathsf{PSig}(M, \mathtt{sk}_i, \{\mathtt{pk}_i, \mathtt{pk}_j\}, \mathtt{apk})$: $\mathtt{ID} \xleftarrow{\$} \{0,1\}^*$, $\mathtt{usk} \leftarrow \mathsf{GOFE.Join}(\mathtt{sk}_i, \mathtt{ID})$ return $\sigma_P \leftarrow \mathsf{GOFE.GPSig}(M, \mathtt{usk}, \{\mathtt{pk}_i, \mathtt{pk}_j\}, \mathtt{apk})$	$\mathsf{Sig}(M, \mathtt{sk}_i, \mathtt{pk}_i, \mathtt{pk}_j, \mathtt{apk})$: $\mathtt{ID} \xleftarrow{\$} \{0,1\}^*$, $\mathtt{usk} \leftarrow \mathsf{GOFE.Join}(\mathtt{sk}_i, \mathtt{ID})$ return $\sigma_P \leftarrow \mathsf{GOFE.GSig}(M, \mathtt{usk}, \{\mathtt{pk}_i, \mathtt{pk}_j\}, \mathtt{apk})$
$\mathsf{PVer}(M, \sigma_P, \{\mathtt{pk}_i, \mathtt{pk}_j\}, \mathtt{apk})$: return $\mathsf{GOFE.GPVer}(M, \sigma_P, \{\mathtt{pk}_i, \mathtt{pk}_j\}, \mathtt{apk})$	$\mathsf{Ver}(M, \sigma_F, \mathtt{pk}_i, \mathtt{pk}_j, \mathtt{apk})$: return $\mathsf{GOFE.GVer}(M, \sigma_F, \mathtt{pk}_i, \mathtt{pk}_j, \mathtt{apk})$

Figure 3: A Generic Transform from GOFE to AOFE

proposed in [17] in terms of the signature size. However, the efficiency gain is expected, because the security of the new AOFE scheme is only provable in the random oracle model.

7 Conclusions and Future Work

In this paper we introduced the notion of group-oriented optimistic fair exchange (GOFE) and gave formal security models for it. We then proposed an efficient construction of GOFE, and analyzed its security under the given models.

Since our GOFE scheme is only provably secure in the random oracle model, and security in this model does not imply real-life security, one of our future work would be to look for a new construction of GOFE that could be proved to be secure in the standard model, and has efficiency comparable with the scheme proposed in this paper.

References

[1] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *CCS*, pages 7–17. ACM, 1997.

[2] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures (extended abstract). In *EUROCRYPT98*, volume 1403 of *LNCS*, pages 591–606. Springer, 1998.

[3] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO00*, volume 1880 of *LNCS*, pages 255–270. Springer, 2000.

[4] G. Ateniese and B. de Medeiros. Efficient group signatures without trapdoors. In *ASIACRYPT03*, volume 2894 of *LNCS*, pages 246–268. Springer, 2003.

[5] D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT04*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[6] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO04*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[7] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT03*, volume 2656 of *LNCS*, pages 416–432. Springer, 2003.

[8] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *ACM CCS*, pages 168–177. ACM, 2004.

[9] X. Boyen and B. Waters. Compact group signatures without random oracles. In *EUROCRYPT06*, volume 4004 of *LNCS*, pages 427–444. Springer, 2006.

[10] X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC07*, volume 4450 of *LNCS*, pages 1–15. Springer, 2007.

[11] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT91*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.

[12] J. H. Cheon. Security analysis of the strong diffie-hellman problem. In *EUROCRYPT06*, volume 4004 of *LNCS*, pages 1–11. Springer, 2006.

[13] I. Damgård. On Σ-protocols. Course on Cryptologic Protocol Theory, Aarhus University, 2009. `http://www.daimi.au.dk/~ivan/Sigma.pdf`.

[14] Y. Dodis, P. J. Lee, and D. H. Yum. Optimistic fair exchange in a multi-user setting. In *PKC07*, volume 4450 of *LNCS*, pages 118–133. Springer, 2007. Also at Cryptology ePrint Archive, Report 2007/182.

[15] J. A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *CRYPTO99*, volume 1666 of *LNCS*, pages 449–466. Springer, 1999.

[16] J. Groth. Fully anonymous group signatures without random oracles. In K. Kurosawa, editor, *ASIACRYPT07*, volume 4833 of *LNCS*, pages 164–180. Springer, 2007. Also at Cryptology ePrint Archive, Report 2007/186.

[17] Q. Huang, G. Yang, D. S. Wong, and W. Susilo. Ambiguous optimistic fair exchange. In *ASIACRYPT08*, volume 5350 of *LNCS*, pages 74–89. Springer, 2008.

[18] Q. Huang, G. Yang, D. S. Wong, and W. Susilo. Efficient optimistic fair exchange secure in the multi-user setting and chosen-key model without random oracles. In *CT-RSA08*, volume 4964 of *LNCS*, pages 106–120. Springer, 2008.

[19] E. Kiltz. Chosen-ciphertext security from tag-based encryption. In S. Halevi and T. Rabin, editors, *TCC06*, volume 3876 of *LNCS*, pages 581–600. Springer, 2006.

[20] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

[21] C. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

A Proof of Lemma 2.8

Proof. We modify the definition of user anonymity to a two-game version. In the first game G_0, the user secret key of user with identity $\mathtt{ID}_0$ is derived from $\mathtt{gsk}_0$, and is used to sign the message M, while in the second game G_1, the user secret key of user with identity $\mathtt{ID}_1$ is derived from $\mathtt{gsk}_0$, and is used to sign M. Let X_i be the event that $\mathcal{D}$ outputs 1 in game G_i. It's not hard to see that this definition of or user anonymity is equivalent to the original one given in Sec. 2.1, and that the advantage of $\mathcal{D}$ in breaking the user anonymity is $\mathtt{Adv}^{\mathrm{UA}}_{\mathcal{D}}(k) = |\Pr[b' = b] - 1/2| = \frac{1}{2}|\Pr[X_1] - \Pr[X_0]|$. Note that the user secret keys of $\mathtt{ID}_0$, $\mathtt{ID}_1$ are derived from $\mathtt{gsk}_0$.

We then construct another game G' which is the same as game G_0, except that the user secret key of identity $\mathtt{ID}_0$ is derived from $\mathtt{gsk}_1$. As the only difference between games G_0 and G' is in the group secret key used to derive the user secret key of $\mathtt{ID}_0$, guaranteed by the group ambiguity, we have that

$$|\Pr[X_0] - \Pr[X']| \leq \mathtt{Adv}^{\mathrm{GA}}_{\mathcal{D}}(k)$$

On the other hand, it is also readily seen that the gap between the probabilities that $\mathcal{D}$ outputs $b' = 1$ in G' and G_1 is bounded by the advantage of it in the game of group ambiguity. That is,

$$|\Pr[X'] - \Pr[X_1]| \leq \mathtt{Adv}^{\mathrm{GA}}_{\mathcal{D}}(k)$$

Therefore, we obtain that

$$\mathtt{Adv}^{\mathrm{UA}}_{\mathcal{D}}(k) = |\Pr[X_0] - \Pr[X_1]|/2 \leq \left(|\Pr[X_0] - \Pr[X']| + |\Pr[X'] - \Pr[X_1]|\right)/2 \leq \mathtt{Adv}^{\mathrm{GA}}_{\mathcal{D}}(k)$$

□

B Security Proofs of the Zero-Knowledge Protocols

Theorem 4.1 is proved by the following lemmas.

Lemma B.1. *Protocol 1 is complete.*

Proof. Suppose that both Alice and Bob follow the protocol, then we have that

$$\begin{aligned}
u^{s_\alpha} &= u^{r_\alpha + c\alpha} = u^{r_\alpha} \cdot (u^\alpha)^c = R_1 \cdot T_1^c, \\
v^{s_\beta} &= v^{r_\beta + c\beta} = v^{r_\beta} \cdot (v^\beta)^c = R_2 \cdot T_2^c, \\
U^{s_{\alpha'}} &= U^{r_{\alpha'} + c\alpha'} = U^{r_{\alpha'}} \cdot (U^{\alpha'})^c = R_3 \cdot S_1^c, \\
V^{s_{\beta'}} &= V^{r_{\beta'} + c\beta'} = V^{r_{\beta'}} \cdot (V^{\beta'})^c = R_4 \cdot S_2^c, \\
u^{-s_1} T_1^{s_x} &= u^{-r_1 - c\delta_1} T_1^{r_x + cx} = u^{-r_1} T_1^{r_x} \cdot (u^{-\delta_1} T_1^x)^c = u^{-r_1} T_1^{r_x} \cdot (u^{-\alpha} T_1)^{cx} = R_5, \\
v^{-s_2} T_2^{s_x} &= v^{-r_2 - c\delta_2} T_2^{r_x + cx} = v^{-r_2} T_2^{r_x} \cdot (v^{-\delta_2} T_2^x)^c = v^{-r_2} T_2^{r_x} \cdot (v^{-\beta} T_2)^{cx} = R_6, \\
U^{-s_3} S_1^{s_\alpha} &= U^{-r_3 - c\delta_3} S_1^{r_\alpha + c\alpha} = U^{-r_3} S_1^{r_\alpha} \cdot (U^{-\delta_3} S_1^\alpha)^c = R_7 \cdot (U^{-\alpha'} S_1)^{c\alpha} = R_7, \\
V^{-s_4} S_2^{s_\alpha} &= V^{-r_4 - c\delta_4} S_2^{r_\alpha + c\alpha} = V^{-r_4} S_2^{r_\alpha} \cdot (V^{-\delta_4} S_2^\alpha)^c = R_8 \cdot (V^{-\beta'} S_2)^{c\alpha} = R_8, \\
U^{-s_5} S_1^{s_\beta} &= U^{-r_5 - c\delta_5} S_1^{r_\beta + c\beta} = U^{-r_5} S_1^{r_\beta} \cdot (U^{-\delta_5} S_1^\beta)^c = R_9 \cdot (U^{-\alpha'} S_1)^{c\beta} = R_9, \\
V^{-s_6} S_2^{s_\beta} &= V^{-r_6 - c\delta_6} S_2^{r_\beta + c\beta} = V^{-r_6} S_2^{r_\beta} \cdot (V^{-\delta_6} S_2^\beta)^c = R_{10} \cdot (V^{-\beta'} S_2)^{c\beta} = R_{10}, \\
H^{s_{\alpha'+\beta'}} &= H^{r_{\alpha'} + c\alpha' + r_{\beta'} + c\beta'} = H^{r_{\alpha'} + r_{\beta'}} \cdot (H^{\alpha'+\beta'})^c = R_{11} \cdot (S_3/\Gamma_0)^c, \quad \text{and} \\
&\quad \mathsf{e}(T_3, H)^{-s_{\alpha'} - s_{\beta'}} \mathsf{e}(T_3, g)^{s_x} \mathsf{e}(h, S_3)^{-s_\alpha - s_\beta} \mathsf{e}(h, H)^{s_3 + s_4 + s_5 + s_6} \mathsf{e}(h, g)^{-s_1 - s_2} \\
&= \mathsf{e}(T_3, H)^{-r_{\alpha'} - r_{\beta'}} \mathsf{e}(T_3, g)^{r_x} \mathsf{e}(h, S_3)^{-r_\alpha - r_\beta} \mathsf{e}(h, H)^{r_3 + r_4 + r_5 + r_6} \mathsf{e}(h, g)^{-r_1 - r_2} \cdot \\
&\qquad \Big(\mathsf{e}(T_3, H)^{-\alpha' - \beta'} \mathsf{e}(T_3, g)^x \mathsf{e}(h, S_3)^{-\alpha - \beta} \mathsf{e}(h, H)^{\delta_3 + \delta_4 + \delta_5 + \delta_6} \mathsf{e}(h, g)^{-\delta_1 - \delta_2} \Big)^c \\
&= R_{12} \cdot (\mathsf{e}(g, g)/\mathsf{e}(T_3, S_3))^c.
\end{aligned}$$

□

Lemma B.2. *For any $(g, u, v, h, U, V, H, \Gamma_0, \Gamma_1)$, Protocol 1 is honest-verifier zero-knowledge under the Decision Linear Assumption.*

Proof. The crux of the proof is the simulator $\mathcal{S}$. It begins by randomly picking $\alpha, \beta, \alpha', \beta' \in \mathbb{Z}_p$ and $A \in \mathbb{G}$, and computing $T_1 \leftarrow u^\alpha$, $T_2 \leftarrow v^\beta$, $T_3 \leftarrow A \cdot h^{\alpha+\beta}$,, $S_1 \leftarrow U^{\alpha'}$, $S_2 \leftarrow V^{\beta'}$ and $S_3 \leftarrow \Gamma_0 \cdot H^{\alpha'+\beta'}$. Under the decision linear assumption, $(T_1, T_2, T_3, S_1, S_2, S_3)$ is indistinguishable from that generated by a real prover.

Note that the rest of the simulation is *independent* of the choice of $(T_1, T_2, T_3, S_1, S_2, S_3) \in \mathbb{G}^6$. Below we describe how $\mathcal{S}$ works. It randomly selects challenge $c \in \mathbb{Z}_p$, and picks at random $s_x, s_\alpha, s_\beta, s_{\alpha'}, s_{\beta'}, s_1, s_2, s_3, s_4, s_5, s_6 \in \mathbb{Z}_p$. It computes the following values:

$$\begin{gathered}
R_1 \leftarrow u^{s_\alpha}/T_1^c, \qquad R_2 \leftarrow v^{s_\beta}/T_2^c, \qquad R_3 \leftarrow U^{s_{\alpha'}}/S_1^c, \qquad R_4 \leftarrow V^{s_{\beta'}}/S_2^c, \\
R_5 \leftarrow u^{-s_1} T_1^{s_x}, \qquad R_6 \leftarrow v^{-s_2} T_2^{s_x}, \qquad R_7 \leftarrow U^{-s_3} S_1^{s_\alpha}, \qquad R_8 \leftarrow V^{-s_4} S_2^{s_\alpha}, \\
R_9 \leftarrow U^{-s_5} S_1^{s_\beta}, \qquad R_{10} \leftarrow V^{-s_6} S_2^{s_\beta}, \qquad R_{11} \leftarrow H^{s_{\alpha'} + s_{\beta'}}/(S_3/\Gamma_0)^c, \\
R_{12} \leftarrow \mathsf{e}(T_3, H)^{-s_{\alpha'} - s_{\beta'}} \mathsf{e}(T_3, g)^{s_x} \mathsf{e}(h, S_3)^{-s_\alpha - s_\beta} \mathsf{e}(h, H)^{s_3 + s_4 + s_5 + s_6} \mathsf{e}(h, g)^{-s_1 - s_2} / (\mathsf{e}(g, g)/\mathsf{e}(T_3, S_3))^c.
\end{gathered}$$

It's readily seen that these values satisfy equations (3) to (14). Besides, $R_1, R_2, \cdots, R_{12}$ are distributed identically to those in a real transcript. The simulator outputs $(T_1, T_2, T_3, S_1, S_2, S_3, R_1, R_2, \cdots, R_{12}, c, s_x, s_\alpha, s_\beta, s_{\alpha'}, s_{\beta'}, s_1, s_2, \cdots, s_6)$. As discussed above, the transcript is indistinguishable from any transcript of Protocol 1, assuming the hardness of decision linear problem. □

Lemma B.3. *Protocol 1 has special soundness, i.e. there is an extractor for Protocol 1 which can extract an SDH tuple.*

Proof. Suppose that an extractor can rewind a prover in Protocol 1 to the status just before the prover is given a challenge c. At the first step of the protocol, the prover sends $T_1, T_2, T_3, S_1, S_2, S_3$ and $R_1, \cdots, R_{12}$. Then, to challenge value c, the prover responds with $s_x, s_\alpha, s_\beta, s_{\alpha'}, s_{\beta'}, s_1, \cdots, s_6$. To challenge value $c' \neq c$, the prover responds with $s'_x, s'_\alpha, s'_\beta, s'_{\alpha'}, s'_{\beta'}, s'_1, \cdots, s'_6$

Let $\Delta c = c - c'$, $\Delta s_\alpha = s_\alpha - s'_\alpha$, and similarly for Δs_β, $\Delta s_{\alpha'}$, $\Delta s_{\beta'}$ and Δs_i for $i = 1, 2, \cdots, 6$. Consider (3) first. Dividing the two instances of this equation (one instance using c and the other using c'), we obtain $u^{\Delta s_\alpha} = T_1^{\Delta c}$. Note that the exponents are in a group of known prime order, so we can take roots. Let $\tilde{\alpha} = \Delta s_\alpha / \Delta c$. Then $u^{\tilde{\alpha}} = T_1$. Similarly, from equations (4), (5) and (6) we can get $\tilde{\beta} = \Delta s_\beta / \Delta c$, $\tilde{\alpha'} = \Delta s_{\alpha'} / \Delta c$ and $\tilde{\beta'} = \Delta s_{\beta'} / \Delta c$ such that $v^{\tilde{\beta}} = T_2$, $U^{\tilde{\alpha'}} = S_1$ and $V^{\tilde{\beta'}} = S_2$ respectively.

Then, from the two instances of (7), we obtain $T_1^{\Delta s_x} = u^{\Delta s_1}$, thus $\Delta s_1 = \tilde{\alpha} \cdot \Delta s_x$. Similarly, from (8), (9), (10), (11) and (12), we obtain $\Delta s_2 = \tilde{\beta} \cdot \Delta s_x$, $\Delta s_3 = \tilde{\alpha'} \cdot \Delta s_\alpha$, $\Delta s_4 = \tilde{\beta'} \cdot \Delta s_\alpha$, $\Delta s_5 = \tilde{\alpha'} \cdot \Delta s_\beta$ and $\Delta s_6 = \tilde{\beta'} \cdot \Delta s_\beta$ such that $T_2^{\Delta s_x} = v^{\Delta s_2}$, $S_1^{\Delta s_\alpha} = U^{\Delta s_3}$, $S_2^{\Delta s_\alpha} = V^{\Delta s_4}$, $S_1^{\Delta s_\beta} = U^{\Delta s_5}$ and $S_2^{\Delta s_\beta} = V^{\Delta s_6}$ respectively. From the two instances of (13), we obtain $H^{\Delta s_{\alpha'} + \Delta s_{\beta'}} = (S_3 / \Gamma_0)^c$. Thus, we get

$$\Gamma_0 = \frac{S_3}{H^{(\Delta s_{\alpha'} + \Delta s_{\beta'})/\Delta c}} = \frac{S_3}{H^{\tilde{\alpha'} + \tilde{\beta'}}}$$

which indicates that Γ_0 is indeed embedded in (S_1, S_2, S_3). Finally, from the two instances of equation (14), we obtain

$$\begin{aligned} &\mathsf{e}(T_3, H)^{-\Delta s_{\alpha'} - \Delta s_{\beta'}} \mathsf{e}(T_3, g)^{\Delta s_x} \mathsf{e}(h, S_3)^{-\Delta s_\alpha - \Delta s_\beta} \mathsf{e}(h, H)^{\Delta s_3 + \Delta s_4 + \Delta s_5 + \Delta s_6} \mathsf{e}(h, g)^{-\Delta s_1 - \Delta s_2} \\ &\qquad = (\mathsf{e}(g, g) / \mathsf{e}(T_3, S_3))^{\Delta c} \end{aligned}$$

Taking the Δc-th root on both sides, we get

$$\begin{aligned} \mathsf{e}(T_3, H)^{-\tilde{\alpha'} - \tilde{\beta'}} \mathsf{e}(T_3, g)^{\tilde{x}} \mathsf{e}(h, S_3)^{-\tilde{\alpha} - \tilde{\beta}} \mathsf{e}(h, H)^{\tilde{\alpha}\tilde{\alpha'} + \tilde{\alpha}\tilde{\beta'} + \tilde{\beta}\tilde{\alpha'} + \tilde{\beta}\tilde{\beta'}} \mathsf{e}(h, g)^{-\tilde{\alpha}\tilde{x} - \tilde{\beta}\tilde{x}} &= \mathsf{e}(g, g) / \mathsf{e}(T_3, S_3) \\ \mathsf{e}(T_3, S_3) \mathsf{e}(T_3, H)^{-\tilde{\alpha'} - \tilde{\beta'}} \mathsf{e}(T_3, g)^{\tilde{x}} \mathsf{e}(h, S_3)^{-\tilde{\alpha} - \tilde{\beta}} \mathsf{e}(h, H)^{(-\tilde{\alpha} - \tilde{\beta})(-\tilde{\alpha'} - \tilde{\beta'})} \mathsf{e}(h, g)^{(-\tilde{\alpha} - \tilde{\beta})\tilde{x}} &= \mathsf{e}(g, g) \\ \mathsf{e}(T_3, S_3 H^{-\tilde{\alpha'} - \tilde{\beta'}} g^{\tilde{x}}) \mathsf{e}(h^{-\tilde{\alpha} - \tilde{\beta}}, S_3 H^{-\tilde{\alpha'} - \tilde{\beta'}} g^{\tilde{x}}) &= \mathsf{e}(g, g) \\ \mathsf{e}(T_3 h^{-\tilde{\alpha} - \tilde{\beta}}, S_3 H^{-\tilde{\alpha'} - \tilde{\beta'}} g^{\tilde{x}}) &= \mathsf{e}(g, g) \\ \mathsf{e}(T_3 h^{-\tilde{\alpha} - \tilde{\beta}}, \tilde{\Gamma}_0 g^{\tilde{x}}) &= \mathsf{e}(g, g) \end{aligned}$$

Let $\tilde{A} = T_3 h^{-\tilde{\alpha} - \tilde{\beta}}$. Then we have $\mathsf{e}(\tilde{A}, \Gamma_0 g^{\tilde{x}}) = \mathsf{e}(g, g)$. Thus the extractor obtains an SDH tuple $(\tilde{A}, \tilde{x})$ and a hidden public key $\tilde{\Gamma}_0$ that satisfy the equation above. Moreover, the $\tilde{A}$ in this SDH tuple is the same as that contained in (T_1, T_2, T_3), and Γ_0 is indeed contained in the linear encryption (S_1, S_2, S_3). □

Theorem 4.2 is proved by the following lemmas.

Lemma B.4. *Protocol 2 is complete.*

This lemma simply follows from the completeness and the simulatability of Protocol 1.

Lemma B.5. *Protocol 2 is honest-verifier zero-knowledge under the Decision Linear Assumption.*

Proof. The simulator begins by preparing values $T_1, T_2, T_3, S_1, S_2, S_3$ as in the proof of Lemma B.2. By the decision linear assumption, these values are indistinguishable from those in a real proof. The rest of the simulation is independent of the choice of $(T_1, T_2, T_3, S_1, S_2, S_3)$.

The simulator picks at randomly two challenge values $c, c^{1-b} \leftarrow \mathbb{Z}_p$, and sets $c^b \leftarrow c - c^{1-b}$. It then runs the simulator of Protocol 1 to produce R, s values for each of c^b and c^{1-b}. Let these values be $(R_1^b, \cdots, R_{12}^b, c^b, s_x^b, s_\alpha^b, s_\beta^b, s_{\alpha'}^b, s_{\beta'}^b, s_1^b, \cdots, s_6^b)$ and $(R_1^{1-b}, \cdots, R_{12}^{1-b}, c^{1-b}, s_x^{1-b}, s_\alpha^{1-b}, s_\beta^{1-b}, s_{\alpha'}^{1-b}, s_{\beta'}^{1-b}, s_1^{1-b}, \cdots, s_6^{1-b})$. Then it outputs $((R_1^0, \cdots, R_{12}^0, c^0, s_x^0, s_\alpha^0, s_\beta^0, s_{\alpha'}^0, s_{\beta'}^0, s_1^0, \cdots, s_6^0), c, (R_1^1, \cdots, R_{12}^1, c^1, s_x^1, s_\alpha^1, s_\beta^1, s_{\alpha'}^1, s_{\beta'}^1, s_1^1, \cdots, s_6^1))$. Guaranteed by Lemma B.2, the output of the simulator is indistinguishable from a real transcript. □

Lemma B.6. *Protocol 2 has special soundness, i.e. there exists an extractor for Protocol 2 which can extract an SDH tuple and the hidden bit b.*

Proof. Suppose that an extractor can rewind a prover in Protocol 2 to the status just before the prover is given a challenge c. At the first step of the protocol, the prover sends $(T_1, T_2, T_3, S_1, S_2, S_3)$ and $((R_1^0, \cdots, R_{12}^0), (R_1^1, \cdots, R_{12}^1))$. Then, to challenge value c, the prover responds with $(c^0, s_x^0, s_\alpha^0, s_\beta^0, s_{\alpha'}^0, s_{\beta'}^0, s_1^0, \cdots, s_6^0, c^1, s_x^1, s_\alpha^1, s_\beta^1, s_{\alpha'}^1, s_{\beta'}^1, s_1^1, \cdots, s_6^1)$. To challenge value $c' \neq c$, the prover responds with $(c'^0, s_x'^0, s_\alpha'^0, s_\beta'^0, s_{\alpha'}'^0, s_{\beta'}'^0, s_1'^0, \cdots, s_6'^0, c'^1, s_x'^1, s_\alpha'^1, s_\beta'^1, s_{\alpha'}'^1, s_{\beta'}'^1, s_1'^1, \cdots, s_6'^1)$.

Since $c' \neq c$, there must be a bit $d \in \{0, 1\}$ such that $c^d \neq c'^d$. Then we consider $(T_1, T_2, T_3, S_1, S_2, S_3, R_1^d, \cdots, R_{12}^d)$ and the two challenge-response tuples, i.e. $(c^d, s_x^d, s_\alpha^d, s_\beta^d, s_{\alpha'}^d, s_{\beta'}^d, s_1^d, \cdots, s_6^d)$ and $(R_1^d, \cdots, R_{12}^d, c'^d, s_x'^d, s_\alpha'^d, s_\beta'^d, s_{\alpha'}'^d, s_{\beta'}'^d, s_1'^d, \cdots, s_6'^d)$. From these two tuples, the extractor can work as that in the proof of Lemma B.3 to extract $\tilde{\alpha}, \tilde{\beta}, \tilde{\alpha'}, \tilde{\beta'}, \tilde{x} \in \mathbb{Z}_p$, $\tilde{A} \in \mathbb{G}$ and Γ_d such that $u^{\tilde{\alpha}} = T_1$, $v^{\tilde{\beta}} = T_2$, $U^{\tilde{\alpha'}} = S_1$, $V^{\tilde{\beta'}} = S_2$, $\tilde{A} = T_3/h^{\tilde{\alpha}+\tilde{\beta}}$, $\Gamma_d = S_3/H^{\tilde{\alpha'}+\tilde{\beta'}}$ and $\mathsf{e}(\tilde{A}, \Gamma_d g^{\tilde{x}}) = \mathsf{e}(g, g)$. The extractor outputs $(\tilde{A}, \tilde{x})$ and the bit d.

For c^{1-d}, we claim that c^{1-d} must be equal to c'^{1-d}. Otherwise, we can run the algorithm of the extractor in the proof of Lemma B.3 on input $(c^{1-d}, s_x^{1-d}, s_\alpha^{1-d}, s_\beta^{1-d}, s_{\alpha'}^{1-d}, s_{\beta'}^{1-d}, s_1^{1-d}, \cdots, s_6^{1-d})$ and $(R_1^{1-d}, \cdots, R_{12}^{1-d}, c'^{1-d}, s_x'^{1-d}, s_\alpha'^{1-d}, s_\beta'^{1-d}, s_{\alpha'}'^{1-d}, s_{\beta'}'^{1-d}, s_1'^{1-d}, \cdots, s_6'^{1-d})$ to extract $\bar{\alpha}, \bar{\beta}, \bar{\alpha'}, \bar{\beta'}, \bar{x} \in \mathbb{Z}_p$ and $\bar{A} \in \mathbb{G}$ such that $U^{\bar{\alpha'}} = S_1$, $V^{\bar{\beta'}} = S_2$, $\Gamma_{1-d} = S_3/H^{\bar{\alpha'}+\bar{\beta'}}$, $u^{\bar{\alpha}} = T_1$, $v^{\bar{\beta}} = T_2$, $\bar{A} = T_3/h^{\bar{\alpha}+\bar{\beta}}$, and $\mathsf{e}(\bar{A}, \Gamma_{1-d} g^{\bar{x}}) = \mathsf{e}(g, g)$.

Since $U^{\bar{\alpha'}} = S_1 = U^{\tilde{\alpha'}}$, we get that $\bar{\alpha'} = \tilde{\alpha'}$. Similarly, we have $\bar{\beta'} = \tilde{\beta'}$, $\bar{\alpha} = \tilde{\alpha}$ and $\bar{\beta} = \tilde{\beta}$. Thus, $\Gamma_{1-d} = S_3/H^{\bar{\alpha'}+\bar{\beta'}} = S_3/H^{\tilde{\alpha'}+\tilde{\beta'}} = \Gamma_d$, and a contradiction is reached, as Γ_0 and Γ_1 are two distinct public keys. Therefore, we obtain that $c^{1-d} = c'^{1-d}$, and thus the two tuples of the response values (i.e. the s values) are also the same. □

C Details of the Generation of (16)

Here is the generation of the signature of knowledge ϑ in details. Note that ϑ consists of two parts, one showing that Γ is correctly encrypted, and the other showing that Γ is correctly decrypted. So we present two separate 3-move (sub-)protocols for proving them respectively. Again, we assume that Alice is the prover and Bob is the verifier.

First, suppose that Alice proves to Bob that $(S_1, S_2, S_3, S_4, S_5)$ is a correct encryption of Γ. She actually proves that there exists some $\eta_1, \eta_2 \in Z_p$ such that

$$U^{\eta_1} = S_1 \ \wedge \ V^{\eta_2} = S_2 \ \wedge \ H^{\eta_1+\eta_2} = S_3/\Gamma \tag{17}$$

Alice randomly selects $\hat{r}_1, \hat{r}_2 \in \mathbb{Z}_p$ and computes $\hat{R}_1 \leftarrow U^{\hat{r}_1}$, $\hat{R}_2 \leftarrow V^{\hat{r}_2}$, $\hat{R}_3 \leftarrow H^{\hat{r}_1+\hat{r}_2}$ and sends $\hat{R}_1, \hat{R}_2, \hat{R}_3$ to Bob, which returns a random challenge $\hat{c} \in \mathbb{Z}_p$. Alice then computes the following

response values $\hat{s}_1 \leftarrow \hat{r}_1 + \hat{c} \cdot \eta_1, \hat{s}_2 \leftarrow \hat{r}_2 + \hat{c} \cdot \eta_2$, and sends them back to Bob. After receiving $(\hat{s}_1, \hat{s}_2)$, Bob checks if $U^{\hat{s}_1} = \hat{R}_1 S_1^{\hat{c}}$, $V^{\hat{s}_2} = \hat{R}_2 S_2^{\hat{c}}$, and $H^{\hat{s}_1+\hat{s}_2} = \hat{R}_3 (S_3/\Gamma)^{\hat{c}}$. It accepts if all the equations hold, and rejects otherwise. It's not hard to see that the protocol is an honest-verifier zero-knowledge proof of knowledge for statement (17).

Now we suppose that Alice proves to Bob that $(S_1, S_2, S_3, S_4, S_5)$ can be decrypted to Γ. She actually proves that there exists some $\eta_1, \eta_2 \in Z_p$ such that

$$U^{\eta_1} = H \ \wedge \ V^{\eta_2} = H \ \wedge \ S_1^{\eta_1} S_2^{\eta_2} = S_3/\Gamma \tag{18}$$

Alice randomly selects $\hat{r}_1, \hat{r}_2 \in \mathbb{Z}_p$ and computes $\hat{R}_1 \leftarrow U^{\hat{r}_1}$, $\hat{R}_2 \leftarrow V^{\hat{r}_2}$, $\hat{R}_3 \leftarrow S_1^{\hat{r}_1} S_2^{\hat{r}_2}$, and sends $(\hat{R}_1, \hat{R}_2, \hat{R}_3)$ to Bob, which returns a random challenge $\hat{c} \in \mathbb{Z}_p$. Alice then computes the response values $\hat{s}_1 \leftarrow \hat{r}_1 + \hat{c} \cdot \eta_1$ and $\hat{s}_2 \leftarrow \hat{r}_2 + \hat{c} \cdot \eta_2$, and sends them back to Bob. After receiving $(\hat{s}_1, \hat{s}_2)$, Bob checks if $U^{\hat{s}_1} = \hat{R}_1 H^{\hat{c}}$, $V^{\hat{s}_2} = \hat{R}_2 H^{\hat{c}}$, and $S_1^{\hat{s}_1} S_2^{\hat{s}_2} = \hat{R}_3 (S_3/\Gamma)^{\hat{c}}$. It accepts if all the equations hold, and rejects otherwise. It's not hard to see that the protocol is also an honest-verifier zero-knowledge proof of knowledge for statement (18).

The full protocol for statement (16) is a standard OR-extension of the two sub-protocols above. The signer uses the randomness (α', β') as the witness in the first sub-protocol and simulates the second one; while the arbitrator uses its secret key (ξ_1, ξ_2) as the witness in the second sub-protocol and simulates the first one. The signature of knowledge ϑ is obtained by applying a variant of Fiat-Shamir heuristic to the full protocol, using hash function $\mathtt{H}_2$. The signature is composed of $(\hat{c}^0, \hat{s}_1^0, \hat{s}_2^0, \hat{c}^1, \hat{s}_1^1, \hat{s}_2^1) \in \mathbb{Z}_p^6$. To verify the validity of ϑ, anyone checks

$$\hat{c}^0 + \hat{c}^1 \stackrel{?}{=} \mathtt{H}_2 \left(M \| \mathtt{gpk}_i \| \mathtt{gpk}_j \| \sigma_P, \frac{U^{\hat{s}_1^0}}{S_1^{\hat{c}^0}}, \frac{V^{\hat{s}_2^0}}{S_2^{\hat{c}^0}}, \frac{H^{\hat{s}_1^0+\hat{s}_2^0}}{(S_3/\Gamma)^{\hat{c}^0}}, \frac{U^{\hat{s}_1^1}}{H^{\hat{c}^1}}, \frac{V^{\hat{s}_2^1}}{H^{\hat{c}^1}}, \frac{S_1^{\hat{s}_1^1} S_2^{\hat{s}_2^1}}{(S_3/\Gamma)^{\hat{c}^1}} \right)$$

D Proof of Theorem 5.1

Here we prove the following lemmas, proving the group ambiguity, security against signing groups, security against verifying groups and traceability, respectively. Theorem 5.1 then follows from them:

Lemma D.1. *If the DLN assumption (t, ϵ)-holds in $\mathbb{G}$, the proposed GOFE scheme is $(t', q_{\mathtt{H}_1}, q_{\mathtt{H}_2}, q_{\mathtt{H}_3}, q_r, \epsilon')$-group ambiguous in the random oracle model, where $t' \approx t$ and $\epsilon' \leq 2\epsilon + q_{\mathtt{H}_1}/p$.*

Proof. We prove the theorem by a series of games. Let $\mathcal{D}$ be an adversary against group ambiguity. Let G_i be the i-th game, and X_i be the event that $\mathcal{D}$ wins in game G_i.

G_0: This is the original game defined in Def. 2.2. By definition, we have that $\Pr[X_0] = \epsilon'$.

G_1: It is identical to G_0, except that the SPK θ in the challenge partial signature σ_P is simulated by controlling the output of the random oracle $\mathtt{H}_1$. By the simulatability of Protocol 2, we have that

$$|\Pr[X_1] - \Pr[X_0]| \leq q_{\mathtt{H}_1}/p$$

where the term $q_{\mathtt{H}_1}/p$ comes from the upper bound of the probability that the challenge value c we choose collides with the outputs of the random oracle $\mathtt{H}_1$ when simulating θ in the challenge partial signature.

G_2: It differs from G_1 in that S_3 in the challenge partial signature is chosen at random from $\mathbb{G}$. Note that this change does not affect the simulation of θ. Thus by the DLN assumption, we get that

$$|\Pr[X_2] - \Pr[X_1]| \leq \epsilon$$

G_3: It differs from G_2 in that T_3 in the challenge partial signature is also chosen at random from $\mathbb{G}$. Again, by the DLN assumption, we get that

$$|\Pr[X_3] - \Pr[X_2]| \leq \epsilon$$

Note that in this game, both (T_1, T_2, T_3) and $(S_1, S_2, S_3, S_4, S_5)$ in the challenge partial signature are encryptions of random elements in $\mathbb{G}$, thus they do not reveal any information about the bit b. Therefore, we have that $\Pr[X_3] = 0$. In a consequence, we obtain that

$$\epsilon' = \Pr[X_0] = |\Pr[X_3] - \Pr[X_0]| \leq \sum_{i=0}^{2} |\Pr[X_{i+1}] - \Pr[X_i]| \leq 2\epsilon + q_{\mathtt{H}_1}/p$$

□

Lemma D.2. *The proposed GOFE scheme is $(t', q_{\mathtt{H}_1}, q_{\mathtt{H}_2}, q_{\mathtt{H}_3}, q_p, q_j, q_t, q_r, \epsilon')$-secure against signing groups if it is $(t, q_{\mathtt{H}_1}, q_{\mathtt{H}_2}, q_{\mathtt{H}_3}, q_p, q_j, \epsilon)$-traceable, where $t' \approx t$ and $\epsilon' \leq (1 - 1/p)\epsilon + 1/p$.*

Proof. Consider the output of an adversary $\mathcal{A}$, i.e. $(M, \sigma_P, \mathtt{gpk}_A)$. Assume that σ_P is a valid partial signature on M under the group public keys $\mathtt{gpk}_A, \mathtt{gpk}_B$ and the arbitration key $\mathtt{apk}$. The special soundness (or extractability) of Protocol 2 implies that with probability at least $1 - 1/p$, the (A, x) and Γ embedded in (T_1, T_2, T_3) and $(S_1, S_2, S_3, S_4, S_5)$ respectively satisfy that $\mathsf{e}(A, \Gamma \cdot g^x) = \mathsf{e}(g, g)$ and Γ is equal to either $\mathtt{gpk}_A$ or $\mathtt{gpk}_B$. On the other side, guaranteed by the traceability (see Lemma D.4), with probability at least $1 - \epsilon$ it holds that $\Gamma \neq \mathtt{gpk}_B$. Otherwise, we obtain a forgery of group B's signature on M. Therefore, the advantage of $\mathcal{A}$ in the game is then bounded as below.

$$\epsilon' \leq 1 - (1 - 1/p)(1 - \epsilon) = (1 - 1/p)\,\epsilon + 1/p$$

□

Lemma D.3. *If DLN assumption (t_1, ϵ_1)-holds in $\mathbb{G}$ and the proposed GOFE scheme is $(t_2, q_{\mathtt{H}_1}, q_{\mathtt{H}_2}, q_{\mathtt{H}_3}, q_p, q_j, \epsilon_2)$-traceable, the proposed GOFE scheme is then $(t', q'_{\mathtt{H}_1}, q'_{\mathtt{H}_2}, q'_{\mathtt{H}_3}, q_p, q_j, q_t, q_r, \epsilon')$-secure against verifying groups, where $t' \approx t_1$, $q'_{\mathtt{H}_1} \leq q_{\mathtt{H}_1} + 1$, $q'_{\mathtt{H}_2} \leq q_{\mathtt{H}_2} + q_r$, $q'_{\mathtt{H}_3} \leq q_{\mathtt{H}_3} + q_r$, and $\epsilon' \leq \epsilon_2 + 6q_p\epsilon_1$.*

Proof. Let $\mathcal{B}$ be an adversary against the security against verifying groups. Let the output of $\mathcal{B}$ be $(M, \mathtt{gpk}_B, \sigma_F)$.

G_0: This is the original game with $\mathcal{B}$ invoked. By definition, we have that $\Pr[X_0] = \epsilon'$.

G_1: It is the same as G_0 except that if $\mathcal{B}$ succeeds in outputting $(M, \mathtt{gpk}_B, \sigma_F)$ without asking $\mathcal{O}_{\mathsf{P}}$ for a partial signature on M with respect to $\mathtt{gpk}_A, \mathtt{gpk}_B$, the game is aborted. Denote this event by F. Since G_1 is the same as G_0 on condition that F does not occur, by the traceability, it holds that

$$|\Pr[X_1] - \Pr[X_0]| \leq \Pr[\mathsf{F}] \leq \epsilon_2$$

Next we show that the advantage of $\mathcal{B}$ in game G_1 is upper bounded as $\Pr[X_1] \leq 6q_p \cdot \epsilon_1$.

Given the adversary $\mathcal{B}$, we use it to construct another algorithm $\mathcal{B}'$ for breaking the decision linear assumption. $\mathcal{B}'$ is given a random instance of decision linear problem, i.e. $(\mathbb{G}, p, \mathsf{u}, \mathsf{v}, \mathsf{h}, \mathsf{u}^a, \mathsf{v}^b, \mathsf{Z}) \in \mathbb{G}^6$ for some unknown $a, b \in \mathbb{Z}_p$. It randomly selects $g, u_A \in \mathbb{G}$ and $\nu_1, \nu_2, \gamma_A \in \mathbb{Z}_p^*$, and computes $\Gamma_A = g^{\gamma_A}$, $h_A = u_A^{\nu_1}$ and $v_A = h_A^{1/\nu_2}$. It randomly selects $\rho_1, \rho_2, \chi^* \in \mathbb{Z}_p$, and computes $K \leftarrow \mathsf{h}^{-\chi^*}\mathsf{u}^{\rho_1}$, $L \leftarrow \mathsf{h}^{-\chi^*}\mathsf{v}^{\rho_2}$. $\mathcal{B}'$ also randomly selects an index $i \leftarrow \{1, 2, \cdots, q_p\}$, and sets $\mathtt{apk} := (U, V, H, K, L) \leftarrow (\mathsf{u}, \mathsf{v}, \mathsf{h}, K, L)$ and $\mathtt{gpk}_A := (\Gamma_A, u_A, v_A, h_A)$. It keeps $\mathtt{gsk}_A := (\gamma_A, \nu_1, \nu_2)$ secret, and invokes $\mathcal{B}$ on input $(\mathtt{apk}, \mathtt{gpk}_A)$. Thanks to the random choices of $g, u_A, \nu_1, \nu_2, \rho_1, \rho_2$, the input to $\mathcal{B}$ is perfectly identical to that in real attacks. $\mathcal{B}'$ then starts to simulate oracles for the adversary.

– $\mathcal{O}_{\mathtt{H}_1}$: Given an input in $\{0,1\}^* \times \mathbb{G}^{19} \times \mathbb{G}_T \times \mathbb{G}^{11} \times \mathbb{G}_T$, $\mathcal{B}'$ randomly selects $c \leftarrow \mathbb{Z}_p$, returns c and stores the input and c in a hash table HT_1 for consistency, i.e. if the input was ever queried before, the oracle returns the answer stored in the table.

– $\mathcal{O}_{\mathtt{H}_2}$: Given an input in $\{0,1\}^* \times \mathbb{G}^{16} \times \mathbb{Z}_p^{24} \times \mathbb{G}^6$, $\mathcal{B}'$ randomly selects $\hat{c} \leftarrow \mathbb{Z}_p$, returns $\hat{c}$ back to $\mathcal{B}$ and stores the input and $\hat{c}$ into a table HT_2 for consistency.

– $\mathcal{O}_{\mathtt{H}_3}$: Given the j-th distinct input $(S_{1j}, S_{2j}, \mathtt{gpk}_{0j}, \mathtt{gpk}_{1j}) \in \mathbb{G}^{10}$, $\mathcal{B}'$ randomly selects $\chi \leftarrow \mathbb{Z}_p$. In the unlikely event that $\chi = \chi^*$, $\mathcal{B}'$ aborts and outputs a random bit. This event occurs with probability at most $q_{\mathtt{H}_3}/p$. It stores the input and χ into a hash table HT_3, and returns χ to $\mathcal{B}$.

– $\mathcal{O}_{\mathsf{P}}$: Let $(M_j, \mathtt{ID}_j, \mathtt{gpk}_j)$ be the j-th (distinct) query. If $j \neq i$, the oracle uses $\mathtt{gsk}_A$ to produce a partial signature on M_j with respect to $\mathtt{gpk}_A, \mathtt{gpk}_j$ and $\mathtt{apk}$ as described in the scheme. Otherwise (i.e. $j = i$), $\mathcal{B}'$ uses $\mathtt{gsk}_A$ to derive the user secret key of identity $\mathtt{ID}_i$, i.e. $(A_i, x_i) \leftarrow \mathsf{Join}(\mathtt{gsk}_A, \mathtt{ID}_i)$, randomly selects $\alpha_i, \beta_i \in \mathbb{Z}_p$, computes $(T_{1i}, T_{2i}, T_{3i}) \leftarrow (u^{\alpha_i}, v^{\beta_i}, A_i \cdot h^{\alpha_i+\beta_i})$. It then sets $(S_{1i}, S_{2i}, S_{3i}, S_{4i}, S_{5i}) := (\mathsf{u}^a, \mathsf{v}^b, \Gamma_A \cdot \mathsf{Z}, (\mathsf{u}^a)^{\rho_1}, (\mathsf{v}^b)^{\rho_2})$. $\mathcal{B}'$ also sets $\mathtt{H}_3(S_{1i}, S_{2i}, \mathtt{gpk}_A, \mathtt{gpk}_i) := \chi^*$ by storing $((S_{1i}, S_{2i}, \mathtt{gpk}_A, \mathtt{gpk}_i), \chi^*)$ into table HT_3. Note that

$$(\mathsf{u}^a)^{\rho_1} = \left(\mathsf{h}^{\chi^*} \cdot \mathsf{h}^{-\chi^* \mathsf{u}^{\rho_1}}\right)^a = \left(\mathsf{h}^{\chi^*} \cdot K\right)^a, \quad \text{and} \quad (\mathsf{v}^b)^{\rho_2} = \left(\mathsf{h}^{\chi^*} \cdot \mathsf{h}^{-\chi^* \mathsf{v}^{\rho_2}}\right)^b = \left(\mathsf{h}^{\chi^*} \cdot L\right)^b.$$

It is unlikely that $\mathcal{B}$ ever queried the oracle $\mathcal{O}_{\mathtt{H}_3}$ on input $(S_{1i}, S_{2i}, \mathtt{gpk}_A, \mathtt{gpk}_i)$ due to the random choices of u^a and v^b. This event happens with probability at most $1/p^2$, which is negligible.

In either case, $\mathcal{B}'$ then generates the signature of knowledge θ_j by means of the simulator of Protocol 2, i.e. patching the output of the random oracle $\mathcal{O}_{\mathtt{H}_1}$. In case of any collision of the output, $\mathcal{B}'$ aborts and outputs with a random bit. This case happens with probability upper bounded by $(q_{\mathtt{H}_1} q_p + q_p^2)/p$. The oracle returns $(T_{1j}, T_{2j}, T_{3j}, S_{1j}, S_{2j}, S_{3j}, S_{4j}, S_{5j}, \theta_j)$ back to $\mathcal{B}$.

If $\mathsf{Z} = \mathsf{h}^{a+b}$, the view of $\mathcal{B}$ is identical to that in a real attack except a difference bounded by $(q_{\mathtt{H}_1} q_p + q_p^2)/p$ thanks to the simulation of θ_i. In the other case, i.e. Z is randomly chosen from $\mathbb{G}$, $(S_{1i}, S_{2i}, S_{3i}, S_{4i}, S_{5i})$ is an encryption of a random element of $\mathbb{G}$. Since the signature of knowledge θ_i is simulated, oracle $\mathcal{O}_{\mathsf{P}}$'s answer to the i-th query does not provide $\mathcal{B}$ any help in producing the final output.

– $\mathcal{O}_{\mathsf{R}}$: Given a message M, a partial signature $\sigma_P = (T_1, T_2, T_3, S_1, S_2, S_3, S_4, S_5, \theta)$ and two group public keys $\mathtt{gpk}_{0j}, \mathtt{gpk}_{1j}$, $\mathcal{B}'$ makes a query on input $(S_1, S_2, \mathtt{gpk}_{0j}, \mathtt{gpk}_{1j})$ to $\mathcal{O}_{\mathtt{H}_3}$, and obtains χ. If $\mathsf{GPVer}(M, \sigma_P, \{\mathtt{gpk}_{0j}, \mathtt{gpk}_{1j}\}, \mathtt{apk}) = 0$, or $\mathsf{e}(S_1, \mathsf{h}^\chi K) \neq \mathsf{e}(\mathsf{u}, S_4)$ or $\mathsf{e}(S_2, \mathsf{h}^\chi L) \neq \mathsf{e}(\mathsf{v}, S_5)$, $\mathcal{B}'$ returns $\perp$. If $\chi = \chi^*$, $\mathcal{B}'$ aborts and outputs a random bit; otherwise, it computes $\Gamma \leftarrow S_3 \cdot ((S_1^{\rho_1} S_2^{\rho_2})/(S_4 S_5))^{1/(\chi - \chi^*)}$. The decryption is correctly done because

$$\frac{S_1^{\rho_1} \cdot S_2^{\rho_2}}{S_4 \cdot S_5} = \frac{(\mathsf{u}^{\alpha'})^{\rho_1} \cdot (\mathsf{v}^{\beta'})^{\rho_1}}{(\mathsf{h}^\chi K)^{\alpha'} \cdot (\mathsf{h}^\chi L)^{\beta'}} = \frac{(\mathsf{h}^{\chi^*} K)^{\alpha'} \cdot (\mathsf{h}^{\chi^*} L)^{\beta'}}{(\mathsf{h}^\chi K)^{\alpha'} \cdot (\mathsf{h}^\chi L)^{\beta'}} = \left(\frac{1}{\mathsf{h}^{\alpha'+\beta'}}\right)^{\chi - \chi^*}$$

If $\Gamma \neq \Gamma_{0j}$ and $\Gamma \neq \Gamma_{1j}$, $\mathcal{B}'$ returns $\perp$; otherwise, it simulates ϑ by invoking the simulator of (16) by patching the output of the random oracle $\mathcal{O}_{\mathtt{H}_2}$. There will be a collsion with probability at most $(q_{\mathtt{H}_2} q_r + q_r^2)/p$. The oracle returns $(\sigma_P, \Gamma, \vartheta)$ to $\mathcal{B}$.

– $\mathcal{O}_{\mathsf{J}}$ and $\mathcal{O}_{\mathsf{T}}$ can be perfectly simulated by $\mathcal{B}'$ using its knowledge of γ_A and ν_1, ν_2 respectively.

Finally, $\mathcal{B}$ outputs $(M, \sigma_F, \mathtt{gpk}_B)$ where $\sigma_F = (\sigma_P, \mathtt{gpk}_A, \vartheta) = ((T_1, T_2, T_3, S_1, S_2, S_3, S_4, S_5, \theta), \mathtt{gpk}_A, \vartheta)$. If $(M, \mathtt{gpk}_B) \neq (M_i, \mathtt{gpk}_i)$, $\mathcal{B}'$ outputs a random bit. If σ_F does not pass the GVer algorithm, $\mathcal{B}'$ outputs 1; otherwise, it outputs 0.

(Probability Analysis). According to the description of $\mathcal{B}'$ above we know that no matter whether Z is equal to h^{a+b} or not, $\mathcal{B}'$ aborts and outputs a random bit with the same probability. Thus, we only concentrate on the case that $\mathcal{B}'$ does not abort below.

According to the game specification of G_1, $\mathcal{B}$ queried $(M, \cdot, \mathtt{gpk}_B)$ for a partial signature. The random guess of the index i of this query is correct with probability $1/q_p$. Let Guess denote the event that the guess of i is correct. Note that, since $\mathcal{B}$ is forbidden from issuing queries like $(M, \cdot, \{\mathtt{gpk}_A, \mathtt{gpk}_B\})$ to $\mathcal{O}_{\mathsf{R}}$, the event that $\chi = \chi^*$ does not occur on condition that there is no collision among the outputs of $\mathcal{O}_{\mathtt{H}_3}$.

Let Abort denote the event that $\mathcal{B}'$ aborts in the attack above. Then we have that the probability that Abort does not occur is lower bounded by

$$\Pr[\neg\mathsf{Abort}] \geq \frac{1}{q_p}\left(1 - \frac{q_{\mathtt{H}_1}q_p + q_p^2}{p}\right)\left(1 - \frac{q_{\mathtt{H}_3}}{p}\right)\left(1 - \frac{1}{p^2}\right)\left(1 - \frac{q_{\mathtt{H}_2}q_r + q_r^2}{p}\right) > \frac{1}{2q_p}$$

Now we assume that $\mathcal{B}'$ does not abort. If $\mathsf{Z} = \mathsf{h}^{a+b}$, $\mathcal{B}$'s view is identical with that in a real attack and thus $\mathcal{B}$ succeeds with probability $\Pr[X_1]$; if Z is randomly selected from $\mathbb{G}$, $(S_{1i}, S_{2i}, S_{3i}, S_{4i}, S_{5i})$ is an encryption of a random element of $\mathbb{G}$. Thanks to the extractability of (the 3-move version of) the arbitrator's signature of knowledge, it turns out that the probability that ϑ is a valid signature showing that $(S_{1i}, S_{2i}, S_{3i}, S_{4i}, S_{5i})$ can be decrypted to Γ_A, is upper bounded by $1/p$.

Let the bit $d = 0$ if $\mathsf{Z} = \mathsf{h}^{a+b}$ and $d = 1$ if $\mathsf{Z} \leftarrow \mathbb{G}$, and let d' be the bit that $\mathcal{B}'$ outputs. We have that

$$\begin{aligned}
\epsilon_1 \geq \mathtt{Adv}^{\mathrm{DLN}}_{\mathcal{B}'}(k) &= \left|\Pr[d' = d] - \frac{1}{2}\right| \\
&= \left|\Pr[\neg\mathsf{Abort}]\left[\Pr[X_1]\left(1 - \frac{1}{p}\right) + (1 - \Pr[X_1])\frac{1}{2}\right] + \Pr[\mathsf{Abort}]\frac{1}{2} - \frac{1}{2}\right| \\
&= \left|\Pr[\neg\mathsf{Abort}]\Pr[X_1]\left(\frac{1}{2} - \frac{1}{p}\right)\right| \\
&\geq \frac{1}{2q_p}\left(\frac{1}{2} - \frac{1}{p}\right)\Pr[X_1] > \frac{1}{6q_p}\Pr[X_1]
\end{aligned}$$

Combining the results above, we obtain that

$$\epsilon' = \Pr[X_0] \leq |\Pr[X_0] - \Pr[X_1]| + \Pr[X_1] \leq \Pr[\mathsf{F}] + \Pr[X_1] < \epsilon_2 + 6q_p\epsilon_1$$

□

Lemma D.4. *If DLN assumption (t_1, ϵ_1)-holds and q-SDH assumption (t_2, ϵ_2)-holds in $\mathbb{G}$, the proposed GOFE scheme is $(t', q_{\mathtt{H}_1}, q_{\mathtt{H}_2}, q_{\mathtt{H}_3}, q_p, q_j, \epsilon')$-traceable in the random oracle model, where*

$$t' \approx \frac{t_2}{2}, \quad q_j < q \quad \textit{and} \quad \epsilon' \leq \frac{q_{\mathtt{H}_1}q_p + q_p^2 + 1}{p} + q_p\epsilon_1 + \left(\frac{16q_{\mathtt{H}_1}\epsilon_2}{1 - 2/p}\right)^{1/2}.$$

Proof. We prove it by a series of games. Let $\mathcal{C}$ be a malicious arbitrator. Let X_i be the event that $\mathcal{C}$ wins the i-th game.

G_0: This is the original game defined in Def. 2.6. By definition, we have that $\Pr[X_0] = \epsilon'$.

G_1: It is the same as G_0 except that the signature of knowledge θ in the answer to each partial signing query is simulated by means of the simulator of Protocol 2, i.e. patching the output of the random oracle $\mathtt{H}_1$. In case of any collision of the output of $\mathtt{H}_1$ in the generation of θ, the game aborts. Clearly, the difference between the advantages of $\mathcal{C}$ in G_0 and G_1 is upper bounded by

$$|\Pr[X_0] - \Pr[X_1]| \leq (q_{\mathtt{H}_1}q_p + q_p^2)/p$$

G_2—G_{q_p+1}: For $1 \le i \le q_p$, game G_{i+1} is the same as G_i except that in game G_{i+1}, to answer the i-th partial signing query, T_3 is chosen at random from $\mathbb{G}$ and the SPK θ is still generated by means of the simulator of Lemma B.5. Obviously, by the decision linear assumption, it holds that

$$|\Pr[X_i] - \Pr[X_{i+1}]| \le \epsilon_1$$

Now we consider game G_{q_p+1}, in which T_3 in all the answers to partial signing queries are chosen at random and θ's are simulated by programming the random oracle $\mathtt{H}_1$. Next we show the upper bound of the advantage of $\mathcal{C}$ in game G_{q_p+1} by constructing another algorithm $\mathcal{F}$ to break the q-SDH assumption.

Given $\mathbb{G}, p, \bar{g}, \bar{g}^{\gamma}, \cdots, \bar{g}^{\gamma^q}$ for $\bar{g} \in \mathbb{G}$ and some unknown $\gamma \in \mathbb{Z}_p^*$ as input, $\mathcal{F}$ first constructs $g, g^{\gamma}, A_1, x_1, \cdots, A_{q-1}, x_{q-1}$ as in [5], where $A_j = g^{1/(\gamma+x_j)}$ and $g = \bar{g}^{f(\gamma)} = \bar{g}^{\prod_{j=1}^{q-1}(\gamma+x_j)}$. It then invokes $\mathcal{C}$ on input $(\mathbb{G}, p, g)$ which returns $\mathtt{apk} = (U, V, H)$. $\mathcal{F}$ randomly chooses $\nu_1, \nu_2 \in \mathbb{Z}_p^*$, $u_A \in \mathbb{G}$, and sets $h_A = u_A^{\nu_1}$, $v_A = h_A^{1/\nu_2}$. It sets $\Gamma_A = g^{\gamma}$ and gives $\mathtt{gpk}_A := (\Gamma_A, u_A, v_A, h_A)$ to $\mathcal{C}$, and then begins to simulate the oracles for $\mathcal{C}$.

The three random oracles are simulated naturally, i.e. on input a distinct query from its respective domain, a random element is selected from $\mathbb{Z}_p$ and returned to $\mathcal{C}$. According to the description of game G_{q_p+1}, to answer each partial signature query, $\mathcal{F}$ chooses T_1, T_2, T_3 at random from $\mathbb{G}$, generates $(S_1, S_2, S_3, S_4, S_5)$ according to the protocol, and produces θ by means of the simulator of Protocol 2. For the j-th query $\mathtt{ID}_j$ to oracle $\mathcal{O}_{\mathsf{J}}$, where $1 \le j < q$, $\mathcal{F}$ returns $\mathtt{usk}_j := (A_j, x_j)$ to $\mathcal{C}$ and stores $(\mathtt{ID}_j, A_j, x_j)$ in its database which is initially empty.

Finally $\mathcal{C}$ outputs $(M, \mathtt{gpk}_B, \sigma_F)$ which satisfying the winning condition, where $\sigma_F = (\sigma_P, \Gamma_A, \vartheta) = ((T_1, T_2, T_3, S_1, S_2, S_3, S_4, S_5, \theta), \Gamma_A, \vartheta)$. We parse θ as $(c^0, s_x^0, s_\alpha^0, s_\beta^0, s_{\alpha'}^0, s_{\beta'}^0, s_1^0, \cdots, s_6^0, c^1, s_x^1, s_\alpha^1, s_\beta^1, s_{\alpha'}^1, s_{\beta'}^1, s_1^1, \cdots, s_6^1)$. Let $\mathtt{ID} \leftarrow \mathsf{Trace}(M, \sigma_F, \mathtt{gsk}_A, \mathtt{gpk}_A, \mathtt{gpk}_B, \mathtt{apk})$. By the validity of $\mathcal{C}$'s output, we have that $\mathtt{ID} \neq \perp$, $\mathtt{ID} \notin Query(\mathcal{C}, \mathcal{O}_{\mathsf{J}})$, and $(M, \mathtt{ID}, \mathtt{gpk}_B) \notin Query(\mathcal{C}, \mathcal{O}_{\mathsf{P}})$. The rest of the proof follows the methodology and notation of the Forking Lemma of [20].

Let $c := c^0 + c^1$, $\sigma_0 := (T_1, T_2, T_3, S_1, S_2, S_3, S_4, S_5, R_1, \cdots, R_{12})$, the values input, together with message M and group public keys $\mathtt{gpk}_A, \mathtt{gpk}_B$, to the random oracle $\mathtt{H}_1$, and from which c is derived. Let $\sigma_1 := (c^0, s_x^0, s_\alpha^0, s_\beta^0, s_{\alpha'}^0, s_{\beta'}^0, s_1^0, \cdots, s_6^0, s_x^1, s_\alpha^1, s_\beta^1, s_{\alpha'}^1, s_{\beta'}^1, s_1^1, \cdots, s_6^1)$. Then the output of $\mathcal{C}$ can be abbreviated as $(M, \mathtt{gpk}_B, ((\sigma_0, c, \sigma_1), \Gamma_A, \vartheta))$. Note that those values omitted from the signature, i.e. the R's, can be recovered as in the verification algorithm (see Sec. 4).

A run of $\mathcal{C}$ is completely described by randomness string ω used by the adversary $\mathcal{C}$ and randomness $\mathbf{h}$ used in simulating the random oracles. Let S be the set of pairs of $(\omega, \mathbf{h})$ such that $\mathcal{C}$ outputs successfully the forgery $(M, \mathtt{gpk}_B, ((\sigma_0, c, \sigma_1), \Gamma_A, \vartheta))$ and $\mathcal{C}$ queries the random oracles $\mathtt{H}_1$ on $(M \| \mathtt{gpk}_A \| \mathtt{gpk}_B, \sigma_0)$. In this case we denote by $\mathrm{Ind}(\omega, \mathbf{h})$ the index of $\mathbf{h}$ at which $\mathcal{C}$ queried $(M \| \mathtt{gpk}_A \| \mathtt{gpk}_B, \sigma_0)$. We define $\mu := \Pr[\mathsf{S}] - 1/p$, where the $1/p$ term accounts for the possibility that $\mathcal{C}$ guessed the hash value of $(M \| \mathtt{gpk}_A \| \mathtt{gpk}_B, \sigma_0)$ without the hash oracle $\mathtt{H}_1$'s help. For each $1 \le i \le q_{\mathtt{H}_1}$, let S_i be the set of pairs of $(\omega, \mathbf{h})$ as above and such that $\mathrm{Ind}(\omega, \mathbf{h}) = i$. Let I be the set of auspicious indices i such that $\Pr[\mathsf{S}_i \mid \mathsf{S}] \ge 1/(2q_{\mathtt{H}_1})$. Then as shown in [20] we have $\Pr[\mathrm{Ind}(\omega, \mathbf{h}) \in I \mid \mathsf{S}] \ge 1/2$.

Let $\mathbf{h}|_a^b$ be the restriction of $\mathbf{h}$ to its elements at indices $a, a+1, \cdots, b$. For each $i \in I$, we consider the Splitting Lemma [20] with rows $X = (\omega, \mathbf{h}|_1^{i-1})$ and columns $Y = (\mathbf{h}|_i^{q_{\mathtt{H}_1}})$. Clearly $\Pr_{x,y}[(x, y) \in \mathsf{S}_i] \ge \nu/(2q_{\mathtt{H}_1})$. Let Ω_i be the set of those rows such that $\forall (x, y) \in \Omega_i : \Pr_{y'}[(x, y') \in \mathsf{S}_i] \ge \nu/(4q_{\mathtt{H}_1})$. Then by the Splitting Lemma, $\Pr[\Omega_i \mid \mathsf{S}_i] \ge 1/2$. A simple argument (similar to Lemma 3 in [20]) then shows that $\Pr[\exists i \in I : \Omega_i \cap \mathsf{S}_i \mid \mathsf{S}] \ge 1/4$. Thus, with probability $\nu/4$, $\mathcal{C}$ succeeds and outputs $(M, \mathtt{gpk}_B, ((\sigma_0, c, \sigma_1), \Gamma_A, \vartheta))$ such that $(M, \sigma_0, c, \sigma_1)$ that derives from some $(x, y) \in \Omega_i$ for some $i \in I$, i.e., an execution $(\omega, \mathbf{h})$ such that $\Pr_{\mathbf{h}'}[(\omega, \mathbf{h}') \in \mathsf{S}_i \mid \mathbf{h}'|_1^{i-1} = \mathbf{h}|_1^{i-1}] \ge \nu/(4q_{\mathtt{H}_1})$.

Now if $\mathcal{F}$ rewinds the adversary $\mathcal{C}$ to the i-th query and proceed with another oracle vector $\mathbf{h}'$ that differs from $\mathbf{h}$ from the i-th entry on, we obtain that with probability at least $\nu/(4q_{\mathtt{H}_1})$ $\mathcal{C}$ succeeds in

outputting another forgery $\sigma'_F = (M, \mathtt{gpk}_B, ((\sigma_0, c', \sigma'_1), \Gamma_A, \vartheta')$ with $(M \| \mathtt{gpk}_A \| \mathtt{gpk}_B, \sigma_0)$ still queried at $\mathcal{C}$'s i-th hash query to $\mathtt{H}_1$. The success of $\mathcal{C}$ indicates that it did not ask $\mathcal{F}$ for a partial signature on M with respect to $\mathtt{gpk}_B$. By using the extractor of Protocol 2 (Lemma B.6) we obtain from (σ_0, c, σ_1) and $(\sigma_0, c', \sigma'_1)$ an SDH tuple (A, x) and Γ, such that $\mathsf{e}(A, g^x \cdot \Gamma) = \mathsf{e}(g, g)$ and either $\Gamma = \Gamma_A$ or $\Gamma = \Gamma_B$, along with randomness $(\alpha, \beta, \alpha', \beta')$ used for the generation of $(T_1, T_2, T_3, S_1, S_2, S_3, S_4, S_5)$. Note that in the simulation above oracles $\mathtt{H}_2$ and $\mathtt{H}_3$ are simulated in a natural way, namely, on input a distinct query, the oracles return a random element selected from $\mathbb{Z}_p$.

We then look at the signature of knowledge in the full signature σ_F, i.e. ϑ. The extractability of (the three-move version of) the signature of knowledge (see Appendix C) implies that the probability that the group public key hidden in $(S_1, S_2, S_3, S_4, S_5)$ is not the one claimed in the signature of knowledge ϑ, is at most $2/p$. Therefore, with probability at least $1-2/p$ we have that $\Gamma = \mathtt{gpk}_A (= g^\gamma)$. Therefore, the pair (A, x) we obtained satisfies that $\mathsf{e}(A, g^\gamma g^x) = \mathsf{e}(g, g)$.

Let $\mathtt{ID}' \leftarrow \mathsf{Trace}(M, \sigma'_F, \mathtt{gsk}_A, \mathtt{gpk}_A, \mathtt{gpk}_B, \mathtt{apk})$. Since σ_0 remains unchanged in the two runs, it turns out that $\mathtt{ID}' = \mathtt{ID}$. By the validity of $\mathcal{C}$'s outputs in the two runs, we know that $\mathcal{C}$ did not ever query $\mathcal{O}_\mathsf{J}$ on input $\mathtt{ID}$. Hence, the pair (A, x) is distinct from all the pairs given to $\mathcal{C}$ and thus x is distinct from $x_1, \cdots, x_{q-1}$. Now we have that

$$A = g^{1/(\gamma+x)} = \bar{g}^{f(\gamma)/(\gamma+x)} = \bar{g}^{f'(\gamma)} \bar{g}^{q_0/(\gamma+x)}$$

for some polynomial $f'(\cdot)$ with $\deg(f') = q-2$ and some $q_0 \neq 0 \in \mathbb{Z}_p$. $\mathcal{F}$ then computes

$$\bar{A} := \bar{g}^{1/(\gamma+x)} = (A / \bar{g}^{f'(\gamma)})^{1/q_0}$$

and outputs $(\bar{A}, x)$, which obviously is a solution to the given q-SDH problem instance.

Put everything above together, we have that the probability that $\mathcal{F}$ solves the q-SDH problem is as below:

$$\begin{aligned} \epsilon_2 \;\geq\; \mathtt{Adv}^{\text{q-SDH}}(k) \geq \frac{\nu}{4} \cdot \frac{\nu}{4q_{\mathtt{H}_1}} \left(1 - \frac{2}{p}\right) \;&=\; \frac{1}{16 q_{\mathtt{H}_1}} \left(\Pr[\mathsf{S}] - \frac{1}{p}\right)^2 \left(1 - \frac{2}{p}\right) \\ \Longrightarrow \quad \Pr[X_{q_p+1}] = \Pr[\mathsf{S}] \;&\leq\; \sqrt{\frac{16 q_{\mathtt{H}_1} \epsilon_2}{1-2/p}} + \frac{1}{p} \end{aligned}$$

Combining all the results above, we then obtain that

$$\epsilon' = \Pr[X_0] \leq \sum_{i=0}^{q_p} |\Pr[X_{i+1}] - \Pr[X_i]| + \Pr[X_{q_p+1}] \leq \frac{q_{\mathtt{H}_1} q_p + q_p^2}{p} + q_p \epsilon_1 + \left(\frac{16 q_{\mathtt{H}_1} \epsilon_2}{1-2/p}\right)^{1/2} + \frac{1}{p}$$

□