Accumulable Optimistic Fair Exchange from Verifiably Encrypted Homomorphic Signatures

Jae Hong Seo[1(✉)], Keita Emura[2], Keita Xagawa[3], and Kazuki Yoneyama[4]

[1] Myongji University, Seoul, South Korea
jaehongseo@mju.ac.kr
[2] NICT, Tokyo, Japan
k-emura@nict.go.jp
[3] NTT Secure Platform Laboratories, Tokyo, Japan
xagawa.keita@lab.ntt.co.jp
[4] Ibaraki University, Ibaraki, Japan
kazuki.yoneyama.sec@ibaraki.ac.jp

Abstract. Let us consider a situation where a client (Alice) frequently buys a certain kind of product from a shop (Bob) (e.g., an online music service sells individual songs at the same price, and a client buys songs multiple times in a month). In this situation, Alice and Bob would like to aggregate the total transactions and pay once per month because individual payments are troublesome. Though optimistic fair exchange (OFE) has been considered in order to swap electronic items simultaneously, known OFE protocols cannot provide such aggregate function efficiently because various costs are bounded by the number of transactions in the period. In order to run this aggregation procedure efficiently, we introduce a new kind of OFE called Accumulable OFE (AOFE) that allows clients to efficiently accumulate payments in each period. In AOFE, any memory costs, computational costs, and communication complexity of the payment round must be constant in terms of the number of transactions. Since a client usually has just a low power and poor memory device, these efficiency are desirable in practice. Currently known approaches (e.g., based on verifiably encrypted signature scheme) are not very successful for constructing AOFE. Thus, we consider a new approach based on a new cryptographic primitive called verifiably encrypted homomorphic signature scheme (VEHS). In this paper, we propose a generic construction of AOFE from VEHS, and also present a concrete VEHS scheme over a composite-order bilinear group by using the dual-form signature techniques. This VEHS scheme is also of independent interest. Since we can prove the security of VEHS without random oracles, our AOFE protocol is also secure without random oracles. Finally, we implemented our AOFE protocol, and it is efficient enough for practical use.

Keywords: Optimistic fair exchange · Homomorphic signatures · Verifiably encrypted signatures

T. Malkin et al. (Eds.): ACNS 2015, LNCS 9092, pp. 192–214, 2015.
DOI: 10.1007/978-3-319-28166-7_10

1 Introduction

In a real trade, a buyer and a seller can exchange goods and money simultaneously in a physical way. Conversely, it is difficult to swap electronic items simultaneously because exchange is usually done through an asynchronous network such as the Internet. Then, a client, Alice, must send her e-cash before receiving an item from a shop, Bob. If Bob is malicious, Bob can abscond with the e-cash without sending the item. Thus, to prevent such a malicious Bob, the fairness is considered as one of the most important requirements for electronic commerce.

Protocols that provide the fair exchange of electronic items are called fair exchange (FE) protocols. FE protocols are roughly classified into two types: those with or without a trusted third party (TTP). An FE protocol without TTP, for example, a gradual release protocol [1–3], is far from practical in terms of communication complexity because the secret must be divided and sent gradually. FE with TTP can be achieved efficiently, but in-line [4,5] or on-line [6,7] TTP protocols are also not practical in a sense because TTP must be involved in all sessions in order to relay transactions between parties. Optimistic FE (OFE) [8–14] is the best of both worlds. Most OFE protocols have the following form; first, Alice sends a partial signature, which is a kind of a contract; that is, a valid partial signature itself is not evidence of payment. Next, Bob sends an item or a signature. Finally, Alice sends a full signature, which is like a check; that is, Bob can cash a valid full signature. When Bob does not receive the full signature from Alice after he sends the item, Bob can obtain the full signature from a TTP, called an adjudicator. The adjudicator has the power to convert the valid partial signature into a valid full signature. That is, the adjudicator does not need to participate in a session as long as the protocol is executed as usual.

1.1 Motivation

Let us consider a situation where Alice frequently buys a certain kind of product from Bob. For example, an online music service sells individual songs at the same price, and a client buys songs multiple times in a month. Another example is an online game; that is, exchanging in-game currency or virtual goods for real money. In these situations, it would be much desirable to allow Alice and Bob to aggregate the total transactions and pay once per month than individual payments if possible.

For the above application, Alice and Bob can perform OFE; both would repeatedly run $k \leq n$ sessions to exchange k partial signatures and k items, and Alice finally sends k full signatures in parallel at the end of a period, where k denotes the number of transactions between Alice and Bob and n be the maximum number of transactions. Although the ordinary OFE could be successful (for fairness) in several applications including the above, we point out that the OFE would be suffered from its linear complexity in k; that is, it is not well scalable in terms of k. More precisely,

- *In terms of memory (RAM) for full signatures:* Alice needs to keep k full signatures if she finally sends k full signatures at the end of the period; Of course, Alice can use an external storage unit to store intermediate state information (e.g., messages, signatures, and public keys) for each transaction, but she must send all k full signatures to Bob at the end of the period; that is, the required RAM size depends on k. In particular, clients may only have a device with an insufficient RAM, so that small memory requirement is desirable.
  For Bob, since he is required to receive and verify all full signatures at the end of the period, the required RAM size also depends on k.
- *In terms of computation for verification:* Since k full signatures are sent by Alice, Bob needs to perform verification algorithm k times individually at the end of the period. Furthermore, Bob (shop) has many clients besides Alice, and so he will be very busy to verify all full signatures given by several clients at the end of the period.
- *In terms of communication for sending signatures:* At the end of the period, Alice and Bob exchange all k full signatures. The network bandwidth of Bob will be stringent at the end of the period since all clients send all their full signatures at the almost same time.

The more frequent the transactions become (i.e., k and n become larger), the more these costs cause the protocol to be impractical. Thus, it is desirable to reduce these costs by accumulating full signatures, and we need an OFE protocol to achieve it. Here, we call this special OFE *accumulable OFE* (AOFE).

Küpçü and Lysyanskaya [15] introduced an OFE protocol (called useful OFE) as a partial solution. In their protocol, exchange of k items is solved by k times repetition of cheap computations, and heavy computations are executed only once within a period. However, if the resolution by the adjudicator is done at the end of the period, Alice and Bob must send all unresolved signatures; and thus, the memory problem remains.

1.2 This Work

We propose the first AOFE protocol. The main building block is a new primitive called *verifiably encrypted homomorphic signature scheme* (VEHS). This paper pioneers a new application of a homomorphic signature scheme (HS), that differs from known applications involving network coding [16] and public computation on authenticated data [17]. Our AOFE protocol is categorized as *setup-free*[1] and *stand-alone*,[2] which are desirable properties [18].

Verifiably Encrypted Signatures. A typical construction of OFE is based on a verifiably encrypted signature (VES) scheme such as [19–23]. The structure of

[1] We say an OFE protocol is *setup-free* if the client does not need to contact the adjudicator except when receiving and verifying the public key certificate of the adjudicator.

[2] We say an OFE protocol is *stand-alone* if the full signature is an ordinary signature.

a VES scheme is such that a signer generates an encrypted signature ω, a verifier can check the validity of ω but not decrypt it, and the adjudicator can decrypt ω and output an ordinary signature σ.[3] Thus, it is compatible with OFE when replacing the signer with the client, the verifier with the shop, the encrypted signature with the partial signature, and the ordinary signature with the full signature. Dodis et al. [12] showed a generic construction of OFE from a VES scheme.[4] We basically follow this paradigm. However, if we simply apply this paradigm, it seems not easy to achieve AOFE because ordinary VES schemes do not support a mechanism to accumulate k full signatures. We solve the problem by using a special type of VES which has an accumulation functionality of signatures.

Why Homomorphic Signatures? In order to construct a VES scheme with such an accumulation functionality, a naive idea is to use aggregate signature (AS) or multi-signature (MS). Originally, these signatures do not match the situation of AOFE (i.e., accumulating full signatures of a signer) because AS and MS are used to accumulate signatures of different signers. Thus, we need a special AS or MS that works correctly and is still secure even if all signatures are generated by the same signer. Though some sequential AS schemes [20, 21, 25, 26] can match this purpose, however, it is not clear whether such the sequential aggregating property can be implemented over encrypted signatures. Since Bellare et al. [27] showed that the BGLS AS [20] can be used without any restriction when the signer's public key is appended to each signed message, and the aggregating property is preserved for encrypted signatures in the BGLS AS. Also, history-free AS [28] can preserve the aggregating property due to history-freeness. However, the BGLS AS and all known constructions of history-free AS rely on random oracle (RO) heuristics, and therefore we may also require a RO even if we can construct a VES based on such schemes. Although RO model schemes have better performance in many cryptographic areas, there are evidences to show the riskiness of schemes with security proofs only in the RO model [29]. Though, Hohenberger et al. [30] propose a technique to remove ROs with multi-linear maps and the technique could be used for the above ASs, there is no known practical construction of multi-linear maps. From the above reasons, we do not select AS or MS schemes to create (verifiably) encrypted signatures according to the VES setting.

[3] As an example, let us consider the Waters signature scheme [24] with public key $x = g^{\alpha}$ and secret key h^{α} and the corresponding VES scheme [21]. Let $\sigma = (\sigma_1, \sigma_2) = (h^{\alpha} \cdot H(m)^r, g^r)$ for random r be an ordinary signature. Let $\mathsf{apk} = y = g^{\beta}$ be the adjudicator's public key. Then, we define $\omega = (\omega_1, \omega_2, \omega_3) = (\sigma_1 \cdot y^t, \sigma_2, g^t)$ for random t. The verfication of an encrypted signature checks if $e(\omega_1, g) = e(h, x) \cdot e(H(m), \omega_2) \cdot e(y, \omega_3)$ or not.

[4] Correctly speaking, they constructed OFE from EUF-CMA secure signature, IND-CCA secure public-key encryption, and simulation-sound non-interactive zero-knowledge proof system, which yield a VES scheme.

We consider HS as a candidate for accumulating signatures. HS allows one to compute linear combinations of given signatures with only public information. The key observation here is that it is possible to accumulate homomorphic signatures by linearly combining them, where messages are of special form so that one can recover each message from a linear combination of them. Also, Alice can accumulate signatures during the period so that she needs only small RAM to compute an accumulated signature. Moreover, we can achieve homomorphic property over encrypted signatures thanks to a homomorphic encryption such as the ElGamal encryption. Therefore, HS could be a good candidate to attain scalability for VES (and OFE also) in terms of the number of transactions k.

Our Contribution. In this paper, we propose a generic method to construct an AOFE protocol based on a VEHS scheme, and also propose a concrete VEHS scheme based on composite-order bilinear groups by using the dual-form signature technique [31,32]. By applying our AOFE protocol, we can achieve that any computational costs of parties and communication complexity of the payment round are constant in terms of k. Moreover, the required RAM space is also constant in terms of k. We describe some technical details of this work.

Security Model of AOFE. We extend the model of OFE in the multi-user setting [12] by introducing algorithms for accumulation of signatures and partial signatures, Acc and PAcc, respectively. We consider three security requirements for clients, shops, and the adjudicator, respectively. The security against clients means that a client cannot produce valid partial signatures such that the verification of the full signature (derived from partial signatures) is not valid. The security against shops means that a shop cannot produce a valid full signature that the adjudicator does not give all ordinary signatures to the shop. The security against the adjudicator means that the adjudicator cannot produce a valid full signature for which the client has not given all partial signatures to the adjudicator.

Security Model of VEHS. Formulating a reasonable security model for VEHS is not a trivial matter, and there exists a subtle issue which is not captured by just simply combining security models of VES and HS. Secure VEHS must satisfy *unforgeability* and *opacity*: Unforgeability guarantees that no adversary can produce a valid encrypted signature which is not generated by the signer. Opacity guarantees that no adversary can produce a valid ordinary signature which is not generated by the adjudicator or the signer. Note that both unforgeability and opacity must be relaxed in the VEHS setting because linear combinations of valid signatures are not regarded as a forgery as they are in the HS setting. Such a complicated situation does not occur with VES and HS. See Sect. 3 for details.

Construction of VEHS. Several types of HS have been studied in the pioneering works [33,34], for example, pairing-based constructions [16,17,35–38], lattice-

based constructions [39,40], an RSA-based construction [41], a multi-source setting [42], and a homomorphic message authentication code [43]. In this paper, we use the composite-order pairing groups for the following technical reason; in the security proof, we must construct a simulator (which solves a hard problem) from an adversary of VEHS. As mentioned above, opacity captures the case where the adversary might forge an ordinary signature of a linear combination of the messages of which the adversary already obtains the encrypted signatures and ordinary signatures. If we use known HS schemes in the prime-order pairing groups, the simulator must guess which messages the adversary uses to forge. However, there are exponentially many combinations of messages; thus, the simulator cannot work. (We will discuss this point in detail in the full version.) On the other hand, if we use HS schemes in the composite-order pairing groups such as the ALP12 signature [17], the simulator does not need to guess messages thanks to the dual-form signature technique [31,32]. This technique allows the simulator to proceed with the simulation while keeping the signing key as long as possible. Fortunately, we can avoid the guessing problem by exploiting this simulation.

According to the classical design principle of VES on the pairing groups, we extend the ALP12 signature to the VES setting; that is, an encrypted signature ω is obtained by encrypting an ordinary signature σ with the ElGamal encryption. We prove that our VEHS scheme is secure in the proposed model under assumptions in [17,44] without ROs. Though the original security proof of the ALP12 signature does not consider opacity, we show that the special situation of opacity is also solved by applying the dual-form signature technique. We finally note that encrypted signatures can also be accumulated thanks to the homomorphism of the ElGamal encryption.

AOFE from VEHS. The design is more complicated than constructing OFE from VES. There are two challenging problems; one is how to encode messages, and the other is how to handle sessions.

For simplicity, we begin with a slightly restricted setting that *a single kind of items is sold with a single rate in a period* (e.g., the on-line music store which sells singles). This setting helps to understand the essential design principle of our construction. After that, we show an extension to the general setting (i.e., items and rates are variable).

For the first problem, we use a vector-representation of the transaction flag as a message. For example, unit vectors $(1, 0, \ldots, 0)$ and $(0, 1, 0, \ldots, 0) \in \mathbb{Z}_N^n$ are signed for the first and second transactions in a period, respectively, and the accumulated signature corresponds to $(1, 1, 0, \ldots, 0)$, where n is the maximum number of transactions in a period and N is an integer. Such an encoded vector is called a *properly augmented vector* [37,45]. Bob can cash the accumulated signature as the full signature according to the Hamming weight of the corresponding message: For example, Alice's signature on $(1, 1, 0, \ldots, 0)$ indicates that Alice buys two items in the period.) This prevents reuse of a valid signature because reuses can be detected by checking if the message vector contains values other than 0 and 1. Attacks except for reuse attacks are prevented thanks to

the unforgeability and opacity of the underlying VEHS. Note that such a message representation does not contain information of items because the amount of money of the item is fixed, and such information is contained in a file identifier.

For the second problem, we use a file identifier τ which is contained in the models of HS and VEHS inherently. τ is used to control the range of linear combination operations that are allowed. If τ is the same for two signatures, signatures can be accumulated. Otherwise, signatures cannot be accumulated. In our AOFE protocol, τ contains identities of a client and a shop (e.g., Alice and Bob), the name and amount of money of an item, and a period of time. Hence, accumulation is possible only for transactions between Alice and Bob for the item in the period, and is prevented for other cases. This ensures security in the multi-user setting, as in [12]. Since a shop does transactions with multiple clients in the same period, security in the multi-user setting is a realistic situation. Note that it is known that security in the single-user setting does not imply security in the multi-user setting [12]. Additionally, we consider more general settings such as when Alice can choose an item from multiple items with a specific amount of money for each transaction, and show a concrete construction.

Our VEHS scheme has $O(n)$ size of public parameters where n is the maximum number of transactions in a period. Hence, Alice and Bob need $O(n)$ size of storages to store public parameters. As mentioned in Sect. 1.1, Alice uses her RAM space by taking necessary information out from the storage. In our instantiation, computation of Alice in each transaction and the end of a period only needs a constant number of contents of public parameters (i.e., each signing needs one of $\{h_i\}_{i\in[n]}$). For example, a message $(1, 1, \ldots, 0) \in \mathbb{Z}_N^n$ will be mapped to a single group element $h_1 h_2$. Thus, our instantiation certainly solves the memory problem. We show an implementation result of our AOFE protocol based on our VEHS scheme in the full version. Though schemes in the composite-order pairing groups usually require a high computational cost, our implementation result shows that our VEHS scheme is efficient enough for practical use.

We also give a way to extend the restricted setting to more general setting such that Alice can choose an item from multiple items with distinct amount of money for each transaction. To deal with distinct values we add a message m_i (for the i-th transaction) representing an amount of money and name of an item to the message vector. We extend the properly augmented vector to $(0, \ldots, 1, \ldots, 0, 0, \ldots, m_i, \ldots, 0)$ where the last half elements are added for containing m_i. The first half elements guarantee secure accumulation as our construction with the restricted setting. In the general setting, file identifier τ just contains identities of a client and a shop (e.g., Alice and Bob), and a period of time because the name and the amount of money may change for each period.

Organization. We define AOFE and VEHS in Sects. 2 and 3, respectively. We construct AOFE from VEHS in Sect. 4. We finally construct VEHS in the composite-order pairing groups in Sect. 5.

2 Definitions of Accumulable Optimistic Fair Exchange

In this section, we define the syntax of AOFE and its security requirements.

An AOFE protocol involves four kinds of parties, clients, shops, an adjudicator, and a trusted party. (We here divide a TTP into an adjudicator and a trusted party that generates only a public parameter.) Roughly, an AOFE protocol is executed as follows: A trusted party generates a public parameter which will be used in every participants in the protocol. Clients and the adjudicator generate their/its key pairs. A client and a shop run the protocol as follows: First, for i-th session of a period, the client generates a partial signature $\omega^{(i)}$ and an ordinary signature $\sigma^{(i)}$ on a message m_i and sends $(\omega^{(i)}, m_i)$ to the shop. The client can also accumulate messages and ordinary signatures to an accumulated message and an accumulated ordinary signature in order to reduce the memory cost. Then, the shop verifies $\omega^{(i)}$ and sends an item corresponding to m_i to the client. The shop can also accumulate messages and partial signatures to an accumulated message and an accumulated partial signature in order to reduce the memory cost. In the end of the period, the client sends the accumulated ordinary signature as the full signature to the shop. The shop verifies the full signature, and he can cash a check if the full signature is correct. Otherwise, the shop asks the resolution to the adjudicator by sending the accumulated message and the accumulated partial signature. The adjudicator verifies them and produces the full signature from the accumulated partial signature.

We note that we formulate a model of AOFE by extending the model of OFE *in the multi-user setting* [12].

First, we define the syntax of AOFE.

Definition 2.1 (Syntax of AOFE)

$\mathsf{OFE.Setup}(1^\kappa)$: *This probabilistic algorithm is run by the trusted third party. It takes security parameter* 1^κ *as input and outputs public parameters* pp. *Hereafter, we omit the public parameter* pp *from the arity of algorithms.*

$\mathsf{OFE.AdjGen}(1^\kappa)$: *This probabilistic algorithm takes as input security parameter* 1^κ *and outputs a pair of keys for an adjudicator* $(\mathsf{apk}, \mathsf{ask})$.

$\mathsf{OFE.Gen}(1^\kappa)$: *This probabilistic algorithm takes as input security parameter* 1^κ, *and outputs a verification/signing key pair* $(\mathsf{vk}_j, \mathsf{sk}_j)$ *for a user* j.

$\mathsf{OFE.Sign}(\mathsf{sk}_j, \mathsf{apk}, m, \mathsf{aux})$: *This probabilistic algorithm takes as input signing key* sk_j, apk,[5] *message* m *to be signed and some session information* aux,[6] *and outputs an ordinary signature* σ.

$\mathsf{OFE.Vrfy}(\mathsf{vk}_j, \mathsf{apk}, m, \sigma, \mathsf{aux})$: *This deterministic algorithm takes as input* vk_j, apk, m, σ *and* aux, *and outputs 1 if* σ *is valid, and 0 otherwise.*

$\mathsf{PSign}(\mathsf{sk}_j, \mathsf{apk}, m, \mathsf{aux})$: *This probabilistic algorithm takes as input* sk_j, apk, m *and* aux, *and outputs a partial signature* ω.

$\mathsf{PVrfy}(\mathsf{vk}_j, \mathsf{apk}, m, \omega, \mathsf{aux})$: *This deterministic algorithm takes as input* vk_j, apk, m, ω *and* aux, *and outputs 1 if* ω *is valid, and 0 otherwise.*

[5] apk is not always used. However, since the definition of $\mathsf{OFE.Sign}$ in OFE [12] contains apk, we adopt the same formulation.

[6] For example, session information contain the current period, and identities of parties.

$\mathsf{Acc}(\mathsf{vk}_j, \mathsf{apk}, \{m_i, \sigma^{(i)}\}_{i=1}^{\ell}, \mathsf{aux})$: *This probabilistic algorithm takes as input* vk_j, apk, $\{m_i, \sigma^{(i)}\}$ *and* aux, *where* $\sigma^{(i)}$ *is an ordinary signature on* m_i *under* vk_j *and* apk, *and outputs an ordinary signature* σ *on* $\sum_{i=1}^{\ell} m_i$ *under* vk_j *and* apk.

$\mathsf{PAcc}(\mathsf{vk}_j, \mathsf{apk}, \{m_i, \omega^{(i)}\}_{i=1}^{\ell}, \mathsf{aux})$: *This probabilistic algorithm takes as input* vk, apk, $\{m_i, \omega^{(i)}\}$ *and* aux, *where* $\omega^{(i)}$ *is a partial signature on* m_i *under* vk_j *and* apk, *and outputs a partial signature* ω *on* $\sum_{i=1}^{\ell} m_i$ *under* vk_j *and* apk.

$\mathsf{Res}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}_j, m, \omega, \mathsf{aux})$: *This (possibly) probabilistic algorithm takes as input* ask, apk, vk_j, m, ω *and* aux, *and outputs an ordinary signature* σ *on* m *under* vk_j *if* $\mathsf{PVrfy}(\mathsf{vk}_j, \mathsf{apk}, m, \omega, \mathsf{aux}) = 1$.

Correctness of AOFE must guarantee that an accumulated signature from valid partial signatures is always acceptable as well as correctness of ordinary OFE.

Definition 2.2 (Correctness). *We say that AOFE satisfies correctness if the following conditions are satisfied: For all* $\kappa \in \mathbb{N}$, *all* $\mathsf{pp} \leftarrow \mathsf{OFE.Setup}(1^\kappa)$, *all* $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{OFE.AdjGen}(1^\kappa)$, *all* $(\mathsf{vk}_j, \mathsf{sk}_j) \leftarrow \mathsf{OFE.Gen}(1^\kappa)$, *all* $\ell \in \mathbb{N}$, *all* $m, m_i \in \mathcal{M}$ *for* $i = 1, \ldots, \ell$, *and all* $\mathsf{aux} \in \{0,1\}^*$,

1. $\mathsf{OFE.Vrfy}\big(\mathsf{vk}_j, \mathsf{apk}, m, \mathsf{OFE.Sign}(\mathsf{sk}_j, \mathsf{apk}, m, \mathsf{aux}), \mathsf{aux}\big) = 1$,
2. $\mathsf{OFE.Vrfy}\big(\mathsf{vk}_j, \mathsf{apk}, \sum_{i=1}^{\ell} m_i, \mathsf{Acc}(\mathsf{vk}_j, \mathsf{apk}, \{m_i, \mathsf{OFE.Sign}(\mathsf{sk}_j, \mathsf{apk}, m_i, \mathsf{aux})\}_{i=1}^{\ell}, \mathsf{aux}), \mathsf{aux}\big) = 1$,
3. $\mathsf{PVrfy}\big(\mathsf{vk}_j, \mathsf{apk}, m, \mathsf{PSign}(\mathsf{sk}_j, \mathsf{apk}, m, \mathsf{aux}), \mathsf{aux}\big) = 1$,
4. $\mathsf{PVrfy}\big(\mathsf{vk}_j, \mathsf{apk}, \sum_{i=1}^{\ell} m_i, \mathsf{PAcc}(\mathsf{vk}_j, \mathsf{apk}, \{m_i, \mathsf{PSign}(\mathsf{sk}_j, \mathsf{apk}, m_i, \mathsf{aux})\}_{i=1}^{\ell}, \mathsf{aux}), \mathsf{aux}\big) = 1$,
5. $\mathsf{OFE.Vrfy}\big(\mathsf{vk}_j, \mathsf{apk}, m, \mathsf{Res}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}_j, m, \mathsf{PSign}(\mathsf{sk}_j, \mathsf{apk}, m, \mathsf{aux}), \mathsf{aux}), \mathsf{aux}\big) = 1$,
6. *and* $\mathsf{OFE.Vrfy}\big(\mathsf{vk}_j, \mathsf{apk}, \sum_{i=1}^{\ell} m_i, \mathsf{Res}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}_j, \sum_{i=1}^{\ell} m_i, \mathsf{PAcc}(\mathsf{vk}_j, \mathsf{apk}, \{m_i, \mathsf{PSign}(\mathsf{sk}_j, \mathsf{apk}, m_i, \mathsf{aux})\}_{i=1}^{\ell}, \mathsf{aux}), \mathsf{aux}), \mathsf{aux}\big) = 1$.

The ambiguity property guarantees that the resolved signature from a partial signature is indistinguishable from the real signature corresponding to the partial signature. In practice, the ambiguity property is necessary to hide if the transaction has some trouble between a client and a shop. We note that the client who causes a trouble in a transaction with a shop should still keep to participate with a transaction with other shops as in the real world and the shop will hope to avoid that the bank knows if the ordinary signature is obtained from the adjudicator, on cashing a check.

Definition 2.3 (Ambiguity [46]). *We say that AOFE satisfies ambiguity if any resolved signature* $\mathsf{Res}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}_j, m, \mathsf{PSign}(\mathsf{sk}_j, \mathsf{apk}, m, \mathsf{aux}), \mathsf{aux})$ *(resp.* $\mathsf{Res}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}_j, \sum_{i=1}^{\ell} m_i, \mathsf{PAcc}(\mathsf{vk}_j, \mathsf{apk}, \{m_i, \mathsf{PSign}(\mathsf{sk}_j, \mathsf{apk}, m_i, \mathsf{aux})\}_{i=1}^{\ell}, \mathsf{aux}), \mathsf{aux})$*) is computationally indistinguishable from the real signature* $\mathsf{OFE.Sign}(\mathsf{sk}_j, \mathsf{apk}, m, \mathsf{aux})$ *(resp.* $\mathsf{Acc}(\mathsf{vk}_j, \mathsf{apk}, \{m_i, \mathsf{OFE.Sign}(\mathsf{sk}_j, \mathsf{apk}, m_i, \mathsf{aux})\}_{i=1}^{\ell}, \mathsf{aux})$*).*

Next, we consider the security model for AOFE. The model contains three requirements: *security against clients*, *security against shops*, and *security against the adjudicator*.

The security against clients means that a client cannot produce valid partial signatures from which the verification of the full signature derived is not valid.

Definition 2.4 (Security against Clients). *We say that an AOFE scheme satisfies security against clients if no PPT adversary $\mathcal{E}$ has a non-negligible advantage (as a function of κ) in the following game:*

1. *Adversary $\mathcal{E}$ is given* pp *and* apk*, where* $\mathsf{pp} \leftarrow \mathsf{OFE.Setup}(1^\kappa)$ *and* $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{OFE.AdjGen}(1^\kappa)$.
2. *$\mathcal{E}$ is allowed to issue queries to the following oracle:*
   Resolution oracle: *This oracle receives verification key* vk_j*, message* m*, partial signature* ω *and* aux*. It verifies* $\mathsf{PVrfy}(\mathsf{vk}_j, \mathsf{apk}, m, \omega, \mathsf{aux}) = 1$*, and returns ordinary signature* $\sigma \leftarrow \mathsf{Res}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}_j, m, \omega, \mathsf{aux})$ *to the adversary.*
3. *Finally, $\mathcal{E}$ outputs* $(\{m_i^*, \omega^{(i)*}\}_{i=1}^{\ell}, \mathsf{vk}_{\mathcal{E}}, \mathsf{aux}^*)$*. We say that $\mathcal{E}$ wins if for* $i = 1$ *to* ℓ*,* $\mathsf{PVrfy}(\mathsf{vk}_{\mathcal{E}}, \mathsf{apk}, m_i^*, \omega^{(i)*}, \mathsf{aux}^*) = 1$*,* $\sigma^{(i)*} \leftarrow \mathsf{Res}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}_{\mathcal{E}}, m_i^*, \omega^{(i)*}, \mathsf{aux}^*)$ *and* $\mathsf{OFE.Vrfy}(\mathsf{vk}_{\mathcal{E}}, \mathsf{apk}, \sum_{i=1}^{\ell} m_i^*, \mathsf{Acc}(\mathsf{vk}_{\mathcal{E}}, \mathsf{apk}, \{m_i^*, \sigma^{(i)*}\}_{i=1}^{\ell}, \mathsf{aux}^*), \mathsf{aux}^*) = 0$.

The advantage of $\mathcal{E}$ is defined as $\mathsf{Adv}_{\mathcal{E}}^{OFE.Client}(\kappa) := \Pr[\mathcal{E}\ wins]$.

$\mathcal{E}$ can select arbitrary verification key $\mathsf{vk}_{\mathcal{E}}$ for a client to attack. Thus, this definition is for the multi-user setting as [12], and captures the situation that $\mathcal{E}$ generates $\mathsf{vk}_{\mathcal{E}}$ without obeying $\mathsf{OFE.Gen}$ (i.e., there exists no corresponding signing key $\mathsf{sk}_{\mathcal{E}}$).

The security against shops means that no shop can produce a valid full signature unless the shop obtains all ordinary signatures corresponding to the full signature.

Definition 2.5 (Security against Shops). *We say that an AOFE scheme satisfies security against shops if no PPT adversary $\mathcal{E}$ has a non-negligible advantage (as a function of κ) in the following game:*

1. *Adversary $\mathcal{E}$ is given* pp*,* apk *and* vk_A*, where* $\mathsf{pp} \leftarrow \mathsf{OFE.Setup}(1^\kappa)$*,* $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{OFE.AdjGen}(1^\kappa)$ *and* $(\mathsf{vk}_A, \mathsf{sk}_A) \leftarrow \mathsf{OFE.Gen}(1^\kappa)$ *for the target client A. Tables T_{psig} and T_{res} are initialized as $\emptyset$.*
2. *$\mathcal{E}$ is allowed to issue queries to the following oracles:*
   Partial signing oracle: *This oracle receives message m and* aux*. It returns partial signature* $\omega \leftarrow \mathsf{PSign}(\mathsf{sk}_A, \mathsf{apk}, m, \mathsf{aux})$ *to the adversary, and stores* $((m, \mathsf{aux}), \omega)$ *in table T_{psig}.*
   Resolution oracle: *This oracle receives verification key* vk_j*, message m, partial signature ω and* aux*. It verifies* $\mathsf{PVrfy}(\mathsf{vk}_j, \mathsf{apk}, m, \omega, \mathsf{aux}) = 1$*, returns ordinary signature* $\sigma \leftarrow \mathsf{Res}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}_j, m, \omega, \mathsf{aux})$ *to the adversary, and stores* $((vk_j, m, \omega, \mathsf{aux}), \sigma)$ *in table T_{res}.*
3. *Finally, $\mathcal{E}$ outputs* $(m^*, \sigma^*, \mathsf{aux}^*)$*. We say that $\mathcal{E}$ wins if* $\mathsf{OFE.Vrfy}(\mathsf{vk}_A, \mathsf{apk}, m^*, \sigma^*, \mathsf{aux}^*) = 1$*, and either of the following holds:*
   – $\mathsf{aux}^* \neq \mathsf{aux}$ *for any entry* $((\cdot, \mathsf{aux}), \cdot) \in T_{psig}$ *and* $((\cdot, \cdot, \cdot, \mathsf{aux}), \cdot) \in T_{res}$.

- $m^* \neq \sum_{i=1}^{\ell} m_i^*$ *for all sets* $\{((m_i^*, \mathsf{aux}^*), \cdot)\}_{i=1}^{\ell} \subseteq T_{psig}$.
- $m^* \neq \sum_{i=1}^{\ell} m_i^*$ *for all sets* $\{((vk_A, m_i^*, \cdot, \mathsf{aux}^*), \cdot)\}_{i=1}^{\ell} \subseteq T_{res}$.

The advantage of $\mathcal{E}$ *is defined as* $\mathsf{Adv}_{\mathcal{E}}^{OFE.Shop}(\kappa) := \Pr[\mathcal{E} \text{ wins}]$.

As in the definition of OFE [12], the target client A is chosen at the beginning of the game. $\mathcal{E}$ can pose the target message for arbitrary verification key vk_j except vk_A to the resolution oracle. Thus, this definition is for the multi-user setting as [12]. That means, $\mathcal{E}$ can arbitrarily interact with all clients and establish sessions with them except the target session. Note that $\mathcal{E}$ does not need the ordinary signing oracle because it can be simulated by the combination of the partial signing oracle and the resolution oracle.

The security against the adjudicator means that no adjudicator can produce a valid full signature unless the adjudicator can generate its public key malignly and obtain all partial signatures corresponding to the full signature.

Definition 2.6 (Security against Adjudicator). *We say that an AOFE scheme satisfies security against the adjudicator if no PPT adversary* $\mathcal{E}$ *has a non-negligible advantage (as a function of* κ*) in the following game:*

1. *Adversary* $\mathcal{E}$ *is given* pp*, and* vk_A*, where* $\mathsf{pp} \leftarrow \mathsf{OFE.Setup}(1^\kappa)$ *and* $(\mathsf{vk}_A, \mathsf{sk}_A) \leftarrow \mathsf{OFE.Gen}(1^\kappa)$ *for the target client* A*.* $\mathcal{E}$ *outputs* apk^**. A table* T_{psig} *is initialized as* $\emptyset$*.*
2. $\mathcal{E}$ *is allowed to issue queries to the following oracle:*
   Partial signing oracle: *This oracle receives message* m *and* aux*. It returns partial signature* $\omega \leftarrow \mathsf{PSign}(\mathsf{sk}_A, \mathsf{apk}^*, m, \mathsf{aux})$ *to the adversary, and stores* $((m, \mathsf{aux}), \omega)$ *in table* T_{psig}*.*
3. *Finally,* $\mathcal{E}$ *outputs* $(m^*, \sigma^*, \mathsf{aux}^*)$*. We say that* $\mathcal{E}$ *wins if* $\mathsf{OFE.Vrfy}(\mathsf{vk}_A, \mathsf{apk}^*, m^*, \sigma^*, \mathsf{aux}^*) = 1$*, and either of the following holds:*
   - $\mathsf{aux}^* \neq \mathsf{aux}$ *for any entry* $((\cdot, \mathsf{aux}), \cdot) \in T_{psig}$.
   - $m^* \neq \sum_{i=1}^{\ell} m_i^*$ *for all sets* $\{((m_i^*, \mathsf{aux}^*), \cdot)\}_{i=1}^{\ell} \subseteq T_{psig}$.

The advantage of $\mathcal{E}$ *is defined as* $\mathsf{Adv}_{\mathcal{E}}^{OFE.Adj}(\kappa) := \Pr[\mathcal{E} \text{ wins}]$.

As in the definition of OFE [12], the target client A is chosen at the beginning of the game. Note that $\mathcal{E}$ does not need the resolution oracle because $\mathcal{E}$ can have ask^* corresponding to apk^*.

We additionally note that if the trusted party, who generates $\mathsf{pp} \leftarrow \mathsf{OFE.Setup}(1^\kappa)$, colluded with an adjudicator, the adjudicator could forge a signature. Indeed, our AOFE scheme built upon a VEHS scheme in Sect. 5 is vulnerable to this attack.[7]

In this paper, we focus on the basic security properties as in [12]. However, additional properties such as abuse-freeness [47], non-repudiation [48], and timely termination [9] can be also considered by the same way as previous works.

[7] Consider the malicious adjudicator knowing the discrete logarithm of h_i, $\log_{g_1}(h_i)$.

3 Definitions of Verifiably Encrypted Homomorphic Signature

In this section, we explain the syntax of VEHS and its security definitions. A VEHS scheme VEHS consists of the following ten algorithms. Let the underlying message space $\mathcal{M}$ be represented as $\mathcal{M} := R^n$ for some integer n and ring R, and let $\mathcal{T}$ be a file-identifier space.

Definition 3.1 (Syntax of VEHS). *We here describe the syntax of VEHS.*

$\mathsf{Setup}(1^\kappa, 1^n)$: *This probabilistic algorithm is run by the trusted third party. It takes security parameter 1^κ and the length of vectors to be signed 1^n as input and outputs public parameters pp. Hereafter, we omit the public parameter pp from the arity of algorithms.*

$\mathsf{AdjGen}(1^\kappa)$: *This probabilistic algorithm takes as input security parameter 1^κ and outputs a pair of keys for an adjudicator $(\mathsf{apk}, \mathsf{ask})$.*

$\mathsf{Gen}(1^\kappa)$: *This probabilistic algorithm takes as input security parameter 1^κ, and outputs a verification/signing key pair for a signer $(\mathsf{vk}, \mathsf{sk})$.*

$\mathsf{Sign}(\mathsf{sk}, \tau, \boldsymbol{v})$: *This probabilistic algorithm takes as input a signing key sk, a file identifier $\tau \in \mathcal{T}$, and a vector $\boldsymbol{v} \in \mathbb{Z}_p^n$ to be signed, and outputs a signature σ.*

$\mathsf{Vrfy}(\mathsf{vk}, \tau, \boldsymbol{v}, \sigma)$: *This deterministic algorithm takes as input vk, τ, $\boldsymbol{v}$, and σ, and outputs 1 if σ is valid, and 0 otherwise.*

$\mathsf{Create}(\mathsf{sk}, \mathsf{apk}, \tau, \boldsymbol{v})$: *This probabilistic algorithm takes as input sk, apk, τ, and $\boldsymbol{v}$, and outputs a VES ω.*

$\mathsf{VesVrfy}(\mathsf{apk}, \mathsf{vk}, \tau, \boldsymbol{v}, \omega)$: *This deterministic algorithm takes as input apk, vk, τ, $\boldsymbol{v}$, and ω, and outputs 1 if ω is valid, and 0 otherwise.*

$\mathsf{Derive}(\mathsf{vk}, \tau, \{\gamma_i, \boldsymbol{v}_i, \sigma^{(i)}\}_{i=1}^{\ell})$: *This probabilistic algorithm takes as input vk, τ, and $\{\gamma_i, \boldsymbol{v}_i, \sigma^{(i)}\}$, where γ_i is a weight and $\sigma^{(i)}$ is a signature on $\boldsymbol{v}_i$ with τ under vk, and outputs a signature σ on $\sum_{i=1}^{\ell} \gamma_i \boldsymbol{v}_i$ with τ under vk.*

$\mathsf{VesDerive}(\mathsf{vk}, \mathsf{apk}, \tau, \{\gamma_i, \boldsymbol{v}_i, \omega^{(i)}\}_{i=1}^{\ell})$: *This probabilistic algorithm takes as input vk, τ, and $\{\gamma_i, \boldsymbol{v}_i, \omega^{(i)}\}$, where γ_i is a weight and $\omega^{(i)}$ is a VES on $\boldsymbol{v}_i$ with τ under vk and apk, and outputs a VES ω on $\sum_{i=1}^{\ell} \gamma_i \boldsymbol{v}_i$ with τ under vk and apk.*

$\mathsf{Adj}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}, \omega, \tau, \boldsymbol{v})$: *This (possibly) probabilistic algorithm takes as input (ask, apk, vk, ω, τ, $\boldsymbol{v}$), and outputs an ordinary signature σ on $\boldsymbol{v}$ with τ under vk if $\mathsf{VesVrfy}(\mathsf{apk}, \mathsf{vk}, \omega, \tau, \boldsymbol{v}) = 1$.*

Let us define correctness of VEHS.

Definition 3.2 (Correctness). *We say a VEHS scheme VEHS is correct if the following conditions are satisfied: For all $\kappa, n \in \mathbb{N}$, all $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{AdjGen}(1^\kappa)$, all $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$, all $\tau \in \mathcal{T}$ and $\boldsymbol{v} \in \mathcal{M}$, and all $\ell \in \mathbb{N}$, we require the following conditions:*

1. $\mathsf{Vrfy}\big(\mathsf{vk}, \tau, \boldsymbol{v}, \mathsf{Sign}(\mathsf{sk}, \tau, \boldsymbol{v})\big) = 1$.
2. $\mathsf{Vrfy}\big(\mathsf{vk}, \tau, \sum_{i=1}^{\ell} \gamma_i \boldsymbol{v}_i, \mathsf{Derive}(\mathsf{vk}, \tau, \{\gamma_i, \boldsymbol{v}_i, \sigma^{(i)}\}_{i=1}^{\ell})\big) = 1$ *if all* $\mathsf{Vrfy}(\mathsf{vk}, \tau, \boldsymbol{v}_i, \sigma^{(i)}) = 1$.

3. $\mathsf{VesVrfy}\big(\mathsf{apk}, \mathsf{vk}, \tau, \boldsymbol{v}, \mathsf{Create}(\mathsf{sk}, \mathsf{apk}, \tau, \boldsymbol{v})\big) = 1$.
4. $\mathsf{VesVrfy}\big(\mathsf{apk}, \mathsf{vk}, \tau, \sum_{i=1}^{\ell} \gamma_i \boldsymbol{v}_i, \mathsf{VesDerive}(\mathsf{vk}, \mathsf{apk}, \tau, \{\gamma_i, \boldsymbol{v}_i, \omega^{(i)}\}_{i=1}^{\ell})\big) = 1$ *if all* $\mathsf{VesVrfy}(\mathsf{apk}, \mathsf{vk}, \tau, \boldsymbol{v}_i, \omega^{(i)}) = 1$.

We can define additional property *resolution independence* of VEHS as that in the context of VES [49]. Roughly speaking, resolution independence implies that an ordinal signature and resolved signature have the same distribution. Since we omit the detail of proofs, we defer the definition of resolution independence to the full version.

We next extend *extractability* of VES [22] to that of VEHS. Roughly speaking, extractability implies that a signature extracted from a valid VES via the Adj algorithm is always valid. Again, we omit the formal defintion of extractability due to page limit.

We define the two security notions *unforgeability* and *opacity*. We consult the security definitions of [17] (Definition 12: unforgeability of a linearly homomorphic signature scheme) and [23] (Definition 4: unforgeability and opacity of a VES scheme). Since VEHS inherits both properties of homomorphic signatures and VESs, we need to keep in mind the security requirements in both contexts.

Before giving definitions, we briefly review unforgeability and opacity of a VES scheme. In the unforgeability game defined in [23], an adversary $\mathcal{A}$ is allowed to obtain VESs from the creation oracle which returns a VES for a queried message, and is also allowed to access the adjudication oracle which extracts and returns a signature for a queried message/VES pair. We strengthen the adversary by allowing it to be a malicious adjudicator. By this strengthening, unforgeability guarantees that even malicious adjudicator cannot produce a valid VES ω^* which is not generated by the creation oracle. Opacity is also defined under the same design principle, where no adversary can produce a valid ordinary signature σ^* which is not generated by the adjudication oracle.

In both definitions, we need to modify the winning condition of $\mathcal{A}$ in the VEHS context because of the homomorphic property. Therefore, we adopt the winning condition of the unforgeability game of [17]. In their unforgeability game, we say that $\mathcal{A}$ wins if $\mathcal{A}$ can produce a valid signature on a message, where the message does not belong to the subspace spanned by all queried messages, or they have a different file identifier from those previously obtained.

Definition 3.3 (Unforgeability). *A VEHS scheme* VEHS *is said to be* unforgeable *if no PPT adversary* $\mathcal{A}$ *has a non-negligible advantage (as a function of* κ *and* n*) in the following game:*

1. $\mathcal{C}$ *runs* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$*, initializes a table* $T_{ves} \leftarrow \emptyset$*, and gives* pp *to the adversary.* $\mathcal{A}$ *chooses* apk^**, and sends it to the challenger* $\mathcal{C}$*.* $\mathcal{C}$ *runs* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$*, and sends* vk *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *is allowed to issue queries to the following oracle:*
   Creation oracle: *This oracle receives a file identifier* $\tau \in \mathcal{T}$ *and an* n*-dimensional vector* $\boldsymbol{v} \in R^n$*. It computes* $\omega \leftarrow \mathsf{Create}(\mathsf{sk}, \mathsf{apk}^*, \tau, \boldsymbol{v})$*, stores* $((\tau, \boldsymbol{v}), \omega)$ *in the table* T_{ves}*, and returns* ω *to the adversary.*

3. *Finally, $\mathcal{A}$ outputs a file identifier τ^*, a vector $\boldsymbol{v}^*$, and a signature σ^*. We say that $\mathcal{A}$ wins if $(\tau^*, \boldsymbol{v}^*) \in \mathcal{M}$ and $\mathsf{Vrfy}(\mathsf{vk}, \tau^*, \boldsymbol{v}^*, \sigma^*) = 1$ hold, and either of the following holds:*
   Class I: *$\tau^* \neq \tau$ for any entry $((\tau, \cdot), \cdot) \in T_{ves}$ and $\boldsymbol{v}^* \neq \boldsymbol{0}$.*
   Class II: *There exists τ such that $\tau^* = \tau$ and $((\tau, \cdot), \cdot) \in T_{ves}$, and $\boldsymbol{v}^* \notin \mathrm{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$, where $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ are vectors which appeared in T_{ves} such that $((\tau^*, \boldsymbol{v}_j), \cdot) \in T_{ves}$ for all $j \in [k]$.*

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{Forge}_{\mathcal{A}}(\kappa, n) := \Pr[\mathcal{A}\ wins]$.

Definition 3.4 (Opacity). *A VEHS scheme* VEHS *is said to be* opaque *if no PPT adversary $\mathcal{A}$ has a non-negligible advantage (as a function of κ and n) in the following game:*

1. *$\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$, initializes two tables $T_{ves}, T_{sig} \leftarrow \emptyset$, and gives pp to the adversary. $\mathcal{C}$ runs $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{AdjGen}(1^\kappa)$ and $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$, and sends apk and vk to $\mathcal{A}$.*
2. *$\mathcal{A}$ is allowed to issue queries to the following two oracles:*
   Creation oracle: *This oracle is the same as that of the unforgeability game.*
   Adjudication oracle: *This oracle receives a file identifier $\tau \in \mathcal{T}$, an n-dimensional vector $\boldsymbol{y} \in R^n$, and a VES ω. If $\mathsf{VesVrfy}(\mathsf{vk}, \mathsf{apk}, \tau, \boldsymbol{y}, \omega) \to 0$, then it returns $\perp$. Otherwise, it computes $\sigma \leftarrow \mathsf{Adj}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}, \omega, \tau, \boldsymbol{y})$, stores $((\tau, \boldsymbol{y}), \sigma)$ in the table T_{sig}, and returns σ to the adversary.*
3. *Finally, $\mathcal{A}$ outputs an identifier τ^*, a vector $\boldsymbol{y}^*$, and a signature σ^*. We say that $\mathcal{A}$ wins if $(\tau^*, \boldsymbol{y}^*) \in \mathcal{M}$, $\boldsymbol{y}^* \neq \boldsymbol{0}$, and $\mathsf{Vrfy}(\mathsf{vk}, \tau^*, \boldsymbol{y}^*, \sigma^*) = 1$ hold, and either of the following holds:*
   Class I: *$\tau^* \neq \tau$ for any entry $((\tau, \cdot), \cdot) \in T_{ves} \cup T_{sig}$.*
   Class II: *There exists τ such that $\tau^* = \tau$ and $((\tau, \cdot), \cdot) \in T_{ves} \cup T_{sig}$, and $\boldsymbol{y}^* \notin \mathrm{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$, where $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ are vectors which appeared in T_{ves} with τ^*; that is, $((\tau^*, \boldsymbol{v}_j), \cdot) \in T_{ves}$ for all $j \in [k]$.*
   Class III: *There exists τ such that $\tau^* = \tau$ and $((\tau, \cdot), \cdot) \in T_{ves} \cup T_{sig}$, $\boldsymbol{y}^* \in \mathrm{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$, and $\boldsymbol{y}^* \notin \mathrm{span}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l)$, where $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ are vectors which appeared in T_{ves} such that $((\tau^*, \boldsymbol{v}_j), \cdot) \in T_{ves}$ for all $j \in [k]$ and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l$ are vectors which appeared in T_{sig} with τ^*, that is, $((\tau^*, \boldsymbol{y}_j), \cdot) \in T_{sig}$ for all $j \in [l]$.*

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{Opac}_{\mathcal{A}}(\kappa, n) := \Pr[\mathcal{A}\ wins]$.

4 Constructions of Accumulable Optimistic Fair Exchange

4.1 Simple Construction and Its Limitation

First, we consider a simple solution toward our goal, and explain its limitations. Let Alice be a client and Bob be a shop. We suppose that Alice and Bob do transactions k times in a period. The simple construction is based on the conventional VES with simple message aggregation (whereas AOFE is based on the

VES aggregation). That is, in the i-th transaction, Alice computes a VES on a message $m_1||\cdots||m_i$, say $\omega^{(i)}$, and sends $(m_i, \omega^{(i)})$ to Bob as a contract. Then, at the finish, Alice sends a full signature on the message $m_1||\cdots||m_k$, which is used as the check for all transactions in this period. The adjudicator verifies a transcript, decrypts a VES $\omega^{(k)}$, and returns $\sigma^{(k)}$, Alice's full signature on the message $m_1||\cdots||m_k$, to Bob.

The weak point of this solution is that it does not support *history-free accumulation*. This property is desirable for a network with out-of-order delivery: Even Alice sends $(m_1, \omega^{(1)})$ and $(m_2, \omega^{(2)})$, Bob may receive $(m_2, \omega^{(2)})$ at first, and later he receives $(m_1, \omega^{(1)})$. Then, Bob cannot verify the validity of the VES $(m_2, \omega^{(2)})$, because he does not know m_1. Therefore, Bob cannot return goods and Alice will be annoyed. That is, the simple construction requires that Bob sequentially verifies encrypted signatures.

As another example, if Alice sends $(m_i, \omega^{(i)})$, where $\omega^{(i)}$ is a VES on m_i, then this problem seems to be solved. However, such a construction is inefficient as we already discussed in the introduction.

Therefore, the approach based on ordinary VES does not fully satisfy our definition of security; and thus, we must consider another approach (i.e., an approach based on VEHS).

4.2 Generic Construction of AOFE from VEHS

Here, we show our generic construction of AOFE ($\mathsf{OFE.Setup}, \mathsf{OFE.AdjGen}, \mathsf{OFE.Gen}, \mathsf{OFE.Sign}, \mathsf{OFE.Vrfy}, \mathsf{PSign}, \mathsf{PVrfy}, \mathsf{Acc}, \mathsf{PAcc}, \mathsf{Res}$) from VEHS ($\mathsf{Setup}, \mathsf{AdjGen}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Create}, \mathsf{VesVrfy}, \mathsf{Derive}, \mathsf{VesDerive}, \mathsf{Adj}$) in the restricted setting (i.e., the name and the cost of an item is fixed in a period). Compared to the simple construction, our generic construction satisfies the ambiguity property, and allows Bob to verify VESs in parallel (or regardless of the order).

Recall that a file identifier τ can be an arbitrary string in VEHS due to our security definitions for VEHS in Sect. 3. We use it to designate identities of a client and a shop, the name and amount of money of an item, and a certain period, e.g., $\tau := H(\text{Alice}||\text{Bob}||\text{Music}||\$10||\text{May})$, where H is a collision resistance hash function. τ is set as session information aux. We suppose that Alice and Bob do transactions k times in a period, where transactions occur at most n times, i.e., $k \leq n$. Let $\mathcal{M} = R^n$ be a message space of VEHS with ring R and integer n. Let $\boldsymbol{v}_i \in R^n$ be a unit vector whose i-th element is 1 and the other elements are 0, that is, $\boldsymbol{v}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$. In the i-th phase, a message is defined as a properly augmented vector $\boldsymbol{v}_i$

Setup Phase

1. $\mathsf{OFE.Setup}(1^\kappa)$: $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, 1^n)$ is provided to all users and the adjudicator.
2. $\mathsf{OFE.AdjGen}(1^\kappa)$: The adjudicator generates $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{AdjGen}(1^\kappa)$.
3. $\mathsf{OFE.Gen}(1^\kappa)$: User i generates $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$.[8]

[8] Because key generation algorithms for a signer and the adjudicator are independent in VEHS, our AOFE protocol is setup-free.

Transaction Phase (For $i = 1$ to k). Alice's key is $(\mathsf{vk}_A, \mathsf{sk}_A)$. Identities of Alice and Bob, the name and amount of money of the item, and the period of the transaction are specified by τ. Initially, Alice sets $\boldsymbol{v} = (0, \ldots, 0)$ and $\sigma = \perp$, and Bob sets $\boldsymbol{v} = (0, \ldots, 0)$ and $\omega = \perp$.

1. $\mathsf{OFE.Sign}(\mathsf{sk}_A, \mathsf{apk}, \boldsymbol{v}_i, \tau)$: Alice generates signature $\sigma^{(i)} \leftarrow \mathsf{Sign}(\mathsf{sk}_A, \tau, \boldsymbol{v}_i)$ as the ordinary signature.
2. $\mathsf{Acc}(\mathsf{vk}_A, \mathsf{apk}, \{(\boldsymbol{v}, \sigma), (\boldsymbol{v}_i, \sigma^{(i)})\}, \tau)$: If $i = 1$, then Alice sets $\boldsymbol{v} := \boldsymbol{v}_1$ and $\sigma := \sigma^{(1)}$. Otherwise, Alice updates $\boldsymbol{v} \leftarrow \boldsymbol{v} + \boldsymbol{v}_i$, and $\sigma \leftarrow \mathsf{Derive}(\mathsf{vk}_A, \tau, \{(1, \boldsymbol{v}, \sigma), (1, \boldsymbol{v}_i, \sigma^{(i)})\})$.[9]
3. $\mathsf{PSign}(\mathsf{sk}_A, \mathsf{apk}, \boldsymbol{v}_i, \tau)$: Alice generates VES $\omega^{(i)} \leftarrow \mathsf{Create}(\mathsf{sk}_A, \mathsf{apk}, \tau, \boldsymbol{v}_i)$ as the partial signature. Alice sends $(\omega^{(i)}, \boldsymbol{v}_i)$ to Bob as a contract.
4. $\mathsf{PVrfy}(\mathsf{vk}_A, \mathsf{apk}, \boldsymbol{v}_i, \omega^{(i)}, \tau)$: Bob verifies that $\boldsymbol{v}_i$ is a unit vector $(\overbrace{0, \ldots, 0}^{i-1}, 1, 0, \ldots, 0)$, and $\mathsf{VesVrfy}(\mathsf{apk}, \mathsf{vk}_A, \tau, \boldsymbol{v}_i, \omega^{(i)}) = 1$. If so, Bob sends the item to Alice.
5. $\mathsf{PAcc}(\mathsf{vk}_A, \mathsf{apk}, \{(\boldsymbol{v}, \omega), (\boldsymbol{v}_i, \omega^{(i)})\}, \tau)$: If $i = 1$, then Bob sets $\boldsymbol{v} := \boldsymbol{v}_1$ and $\omega := \omega^{(1)}$. Otherwise, Bob updates $\boldsymbol{v} \leftarrow \boldsymbol{v} + \boldsymbol{v}_i$, and $\omega \leftarrow \mathsf{VesDerive}(\mathsf{vk}_A, \mathsf{apk}, \tau, \{(1, \boldsymbol{v}, \omega), (1, \boldsymbol{v}_i, \omega^{(i)})\})$.[10]

Check Phase (The end of the period)

1. Alice sends σ as the full signature as a check.[11]
2. $\mathsf{OFE.Vrfy}(\mathsf{vk}_A, \mathsf{apk}, \boldsymbol{v}, \sigma, \tau)$: Bob verifies that $\boldsymbol{v}$ has the form $(\overbrace{1, \ldots, 1}^{k}, 0, \ldots, 0)$, and $\mathsf{Vrfy}(\mathsf{vk}_A, \tau, \boldsymbol{v}, \sigma) = 1$. If so, Bob can cash a check with σ.
3. $\mathsf{Res}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}_A, \boldsymbol{v}, \omega, \tau)$: If $\mathsf{OFE.Vrfy}(\mathsf{vk}_A, \mathsf{apk}, \boldsymbol{v}, \sigma, \tau) = 0$ or Bob has not received the full signature by the end of the period, Bob sends $(\boldsymbol{v}, \omega)$ to the adjudicator. The adjudicator verifies that both $\boldsymbol{v}$ has the form $(\overbrace{1, \ldots, 1}^{k}, 0, \ldots, 0)$, and $\mathsf{VesVrfy}(\mathsf{apk}, \mathsf{vk}_A, \tau, \boldsymbol{v}, \omega) = 1$. Then, the adjudicator runs $\sigma \leftarrow \mathsf{Adj}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}_A, \omega, \tau, \boldsymbol{v})$, and sends σ to Bob.

Correctness and ambiguity of our AOFE protocol are trivially derived from correctness and resolution independence of VEHS; and thus, we omit to prove it.

Note that Bob seems to be able to choose the weight values, and can get a weighted signature from the adjudicator that might not be agreed by Alice. However, this problem does not occur by syntax of messages: the adjudicator verifies the validity of the received sum of VESs by checking the form of $\boldsymbol{v}$ in the step 3 of the Check phase. The adjudicator refuses the malformed weight even if Bob chooses the invalid ones: for example, the adjudicator rejects a message

[9] Then, Alice needs to store just one message and one ordinary signature in her memory during a transaction period.

[10] Then, Bob also needs to store just one message and one partial signature in his memory during a transaction period.

[11] Since the full signature is also an ordinary signature, our protocol is stand-alone.

$\boldsymbol{v} = (2, 1, 1, 0, \ldots, 0)$. Moreover, unforgeability of VEHS guarantees that Bob cannot forge any partial signature of a message vector that the i-th value is 1 when the i-th transaction between Alice and Bob does not occur.

4.3 Security

Due to page limits, we defer the proofs of Theorems 4.1, 4.2, and 4.3 to the full version. We only comment intuition.

Theorem 4.1 (Security against Clients). *Our AOFE protocol is secure against clients if the underlying VEHS scheme is extractable.*

Since the client cannot forge a valid encrypted signature such that the corresponding ordinary signature is not valid because of extractability of the underlying VEHS, this property is guaranteed.

Theorem 4.2 (Security against Shops). *Our AOFE protocol is secure against shops if the underlying VEHS scheme is opaque and resolution independent.*

Since the shop cannot forge a valid full signature without knowing one of corresponding ordinary signatures because of opacity of the underlying VEHS, this property is guaranteed. Also, we need resolution independence to simulate the resolution oracle.

Theorem 4.3 (Security against Adjudicator). *Our AOFE protocol is secure against the adjudicator if the underlying VEHS scheme is unforgeable.*

Since the adjudicator cannot forge both a valid encrypted signature and the corresponding valid ordinary signature without knowing the signing key of the client because of unforgeability of the underlying VEHS, this property is guaranteed.

4.4 Extension to General Setting

The above AOFE protocol only supports the case in which the name and cost of an item in a period are fixed. This protocol covers a situation where Alice frequently buys a certain kind of product from Bob at a flat rate (e.g., an online music service sells individual songs at the same price, and a client buys songs multiple times in a month). Here, we consider the general setting where each item has a distinct amount of money, and Alice can choose an arbitrary item in each transaction. We provide a key idea to extend our basic AOFE protocol into general setting, and the details of the extended AOFE protocol and its security analysis appear in the full version of this paper.

On choosing an item, Alice must include the name and cost of the item in the message field instead of in the file identifier. That is, we add a message m_i (e.g., $H(\text{Music}||\$10)$) to the message vector $\boldsymbol{v}_i$ as $(\boldsymbol{v}_i||m_i\boldsymbol{v}_i) = (0, \ldots, 0, 1, 0, \ldots, 0, m_i, 0, \ldots, 0) \in R^{2n}$, and the file identifier just designates the identities of a client

and a shop, and a period (e.g., $\tau := H(\text{Alice}||\text{Bob}||\text{May})$), where H is a collision resistance hash function. Thus, at the end of the period, the accumulated message vector $\boldsymbol{v}$ is $(1, \ldots, 1, 0, \ldots, 0, m_1, \ldots, m_k, 0, \ldots, 0)$. As the restricted setting, if Bob tries to choose the weight values, and to get a weighted signature from the adjudicator that might not be agreed by Alice, it is prevented by checking the form of the first k elements of $\boldsymbol{v}$. Thus, the security of the construction in the general setting can be proved in the same way as the restricted setting.

5 Construction of Verifiably Encrypted Homomorphic Signature

We first give the definition of bilinear groups. We then propose our VEHS scheme.

5.1 Bilinear Groups with Composite Order

For a set X and an element $x \in X$, $x \leftarrow_{\$} X$ denotes x is chosen uniformly at random from X.

Let us recall the property of composite-order pairing groups. Let $(\mathbb{G}, \mathbb{G}_T)$ be a bilinear group of composite order $N = p_1p_2p_3$, let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, and let $\mathcal{G}$ be a group generator, where $\mathcal{G}$ with the security parameter κ outputs $(\mathbb{G}, \mathbb{G}_T, e, N = p_1p_2p_3)$. For $i, j \in \{1, 2, 3\}$, let $\mathbb{G}_i$ denote the subgroup of $\mathbb{G}$ of the order p_i, and $\mathbb{G}_{i,j}$ $(i \neq j)$ denote the subgroup $\mathbb{G}$ of the order p_ip_j. We note that "orthogonality" of subgroups is as follows. For all $g_i \in \mathbb{G}_i$ and $h_j \in \mathbb{G}_j$ where $i, j \in \{1, 2, 3\}$ and $i \neq j$, $e(g_i, h_j) = 1_T$ holds. Here, 1_T is the unit of $\mathbb{G}_T$. This property is applied in our verification algorithms such that elements of $\mathbb{G}_3$ contained in signatures/VESs are canceled out by pairing computations.

In the proposed scheme, we require that algorithms except for Setup randomly choose an element from the subgroups of $\mathbb{G}$ without knowing the corresponding orders. To do so, generators of subgroups ($g \in \mathbb{G}_1$ and $X_{p_3} \in \mathbb{G}_3$ in the scheme) are included in pp. That is, algorithms just choose a random value $r \in \mathbb{Z}_N$, and compute its exponentiation, e.g., $u := g^r$ and $R_3 = X_{p_3}^r$, and so on. We simply denote these procedures as $u \leftarrow_{\$} \mathbb{G}_1$ and $R_3 \leftarrow_{\$} \mathbb{G}_3$, respectively.

Assumptions. We will employ the following assumptions in the literature in order to prove the security. Due to the space limit, we informally introduce the assumptions. For strict definitions, see the papers [17,44] or the full version of this paper. We note that they are hard in the generic group model.

Assumption LW1' [44]**:** Let $g \leftarrow_{\$} \mathbb{G}_1$, $X_3 \leftarrow_{\$} \mathbb{G}_3$, $T_b \leftarrow_{\$} \mathbb{G}_{1,2}$, and $T_{1-b} \leftarrow_{\$} \mathbb{G}_1$ for $b \leftarrow_{\$} \{0, 1\}$. Given (g, X_3, T_0, T_1), it is infeasible to decide b.[12]

Assumption LW2 [44]**:** Let $g, X_1 \leftarrow_{\$} \mathbb{G}_1$, $X_2, Y_2 \leftarrow_{\$} \mathbb{G}_2$, $Y_3, Z_3 \leftarrow_{\$} \mathbb{G}_3$, and $T \leftarrow_{\$} \mathbb{G}$. Given (g, X_1X_2, Z_3, Y_2Y_3) and T, it is infeasible to decide if $T \leftarrow_{\$} \mathbb{G}$ or $T \leftarrow_{\$} \mathbb{G}_{1,3}$.

[12] In the original assumption LW1 [44], given $g \leftarrow_{\$} \mathbb{G}_1$, $X_3 \leftarrow_{\$} \mathbb{G}_3$, and $T \in \mathbb{G}$, it is infeasible to decide if $T \leftarrow_{\$} \mathbb{G}_1$ or $T \leftarrow_{\$} \mathbb{G}_{1,2}$.

Assumption ALP3 [17]**:** Let $g, f, g^{\xi}, X_1 \leftarrow_{\$} \mathbb{G}_1$ where $\xi \leftarrow_{\$} \mathbb{Z}_N$, $X_2, Y_2, Z_2 \leftarrow_{\$} \mathbb{G}_2$, and $X_3, Y_3, Z_3 \leftarrow_{\$} \mathbb{G}_3$. Given $(g, f, g^{\xi}, X_1X_2, X_3, Y_2Y_3)$ and T, it is infeasible to decide if $T = f^{\xi}Z_3$ or $f^{\xi}Z_2Z_3$.

Assumption ALP4 [17]**:** Let $g \leftarrow_{\$} \mathbb{G}_1$, $X_2, Y_2, Z_2 \leftarrow_{\$} \mathbb{G}_2$, $X_3 \leftarrow_{\$} \mathbb{G}_3$, and $a, b, c \leftarrow_{\$} \mathbb{Z}_N$. Given $(g, g^a, g^b, g^{ab}X_2, X_3, g^cY_2, Z_2)$, it is infeasible to compute $e(g, g)^{abc}$.

5.2 VEHS in Composite-Order Pairing Groups

Our scheme is based on the Attrapadung-Libert-Peters linearly-homomorphic signature scheme [17], the ALP12 scheme in short, which is based on the Lewko-Waters signature scheme [44] in the composite-order pairing groups, and the ElGamal encryption scheme [50]. Thanks to the pairing, we can verify a VES, i.e., an encrypted signature.

One might wonder why we employ the composite-order setting because we already have VES schemes and HS schemes in the prime-order setting. The reason is that there are technical hurdles we cannot solve by our best efforts, although we can simply construct VEHSs from HS schemes in the prime-order setting and the ElGamal encryption scheme. Let $\sigma = (\sigma_1, \sigma_{\mathrm{rest}})$ be an ordinary signature. Let $\mathsf{apk} = y = g^{\beta}$ be the adjudicator's public key. Then, we let a VES $\omega = (\omega_1, \omega_2, \omega_3)$ as $\omega_1 \leftarrow \sigma_1 \cdot y^t$, $\omega_2 \leftarrow \sigma_{\mathrm{rest}}$, and $\omega_3 \leftarrow g^t$. The main hurdle is the security proof on class-III opacity in Definition 3.4. Roughly speaking, we have to solve the problem in an assumption by using the adversary's power to strip y^t off ω_1. With VES schemes, one can guess which VES will be stripped out and thus embed the problem into ω. We fail to adopt this technique in the HS setting: it is hard to guess which vector $\boldsymbol{v}_i$ on τ^* the adversary will use to forge $\boldsymbol{y}^*$. Fortunately, we can prove class-III opacity in the composite-order setting by using the dual-form signature technique as we discuss in the introduction.

Our VEHS Scheme

- $\mathsf{Setup}(1^{\kappa}, 1^n)$: Choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of order $N = p_1p_2p_3$ such that $(\mathbb{G}, \mathbb{G}_T, e, N) \leftarrow_{\$} \mathcal{G}$. Choose $g, u, v, h_1, \ldots, h_n \leftarrow_{\$} \mathbb{G}_1$ and $X_{p_3} \leftarrow_{\$} \mathbb{G}_3$. $\mathsf{pp} = (\mathbb{G}, \mathbb{G}_T, e, N, g, X_{p_3}, u, v, \{h_i\}_{i\in[n]})$. Here, we let $H_{\mathsf{hom}}(\boldsymbol{v}) := \prod_{i\in[n]} h_i^{v_i}$, where $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_N^n$. Note that $\prod_{i\in[\ell]} H_{\mathsf{hom}}(\boldsymbol{v}_i)^{\gamma_i} = H_{\mathsf{hom}}(\sum_{i\in[\ell]} \gamma_i \boldsymbol{v}_i)$ holds. We omit the public parameter pp from inputs of following algorithms.
- $\mathsf{AdjGen}(1^{\kappa})$: Choose $\beta \leftarrow_{\$} \mathbb{Z}_N$ and compute $y \leftarrow g^{\beta}$. Output $(\mathsf{apk}, \mathsf{ask}) = (y, \beta)$.
- $\mathsf{Gen}(1^{\kappa})$: Choose $\alpha \leftarrow_{\$} \mathbb{Z}_N$ and compute g^{α}. Output $\mathsf{vk} = g^{\alpha}$ and $\mathsf{sk} = \alpha$.
- $\mathsf{Sign}(\mathsf{sk}, \tau, \boldsymbol{v})$: Return $\perp$ if $\boldsymbol{v} = \boldsymbol{0}$. Choose $r \leftarrow_{\$} \mathbb{Z}_N$ and $R_3, R_3' \leftarrow_{\$} \mathbb{G}_3$. Compute $\sigma_1 \leftarrow H_{\mathsf{hom}}(\boldsymbol{v})^{\alpha} \cdot (u^{\tau}v)^r \cdot R_3$ and $\sigma_2 \leftarrow g^r \cdot R_3'$. Output $\sigma = (\sigma_1, \sigma_2)$.[13]
- $\mathsf{Vrfy}(\mathsf{vk}, \tau, \boldsymbol{v}, \sigma)$: Parse $\sigma = (\sigma_1, \sigma_2)$. Return 1 iff $e(\sigma_1, g) = e(H_{\mathsf{hom}}(\boldsymbol{v}), g^{\alpha}) \cdot e(u^{\tau}v, \sigma_2)$ holds. Otherwise, return 0.

[13] As a remark, a client Alice needs to compute H_{hom} for a vector $\boldsymbol{v} = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$ in the AOFE protocol based on our VEHS scheme. Therefore, no n-dependent computation is required for Alice in our AOFE protocol.

- $\mathsf{Create}(\mathsf{sk}, \mathsf{apk}, \tau, \boldsymbol{v})$: Run $\sigma := (\sigma_1, \sigma_2) \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau, \boldsymbol{v})$. Choose $t \leftarrow_{\$} \mathbb{Z}_N$ and $R_3'' \leftarrow_{\$} \mathbb{G}_3$, compute $\omega_1 \leftarrow \sigma_1 \cdot y^t$, $\omega_2 \leftarrow \sigma_2$, and $\omega_3 \leftarrow g^t \cdot R_3''$. Output $\omega = (\omega_1, \omega_2, \omega_3)$.
- $\mathsf{VesVrfy}(\mathsf{apk}, \mathsf{vk}, \tau, \boldsymbol{v}, \omega)$: Parse $\omega = (\omega_1, \omega_2, \omega_3)$. Return 1 iff $e(\omega_1, g) = e(H_{\mathsf{hom}}(\boldsymbol{v}), g^{\alpha}) \cdot e(u^{\tau} v, \omega_2) \cdot e(y, \omega_3)$ holds. Otherwise, return 0.
- $\mathsf{Derive}(\mathsf{vk}, \tau, \{\gamma_i, \boldsymbol{v}_i, \sigma^{(i)}\}_{i=1}^{\ell})$: Parse $\sigma^{(i)} = (\sigma_{i,1}, \sigma_{i,2})$. Choose $\tilde{r} \leftarrow_{\$} \mathbb{Z}_N$ and $\tilde{R}_3, \tilde{R}_3' \leftarrow_{\$} \mathbb{G}_3$. Compute $\sigma_1 \leftarrow \left(\prod_{i \in [\ell]} \sigma_{i,1}^{\gamma_i}\right) \cdot (u^{\tau} v)^{\tilde{r}} \cdot \tilde{R}_3$ and $\sigma_2 \leftarrow \left(\prod_{i \in [\ell]} \sigma_{i,2}^{\gamma_i}\right) \cdot g^{\tilde{r}} \cdot \tilde{R}_3'$. Output $\sigma = (\sigma_1, \sigma_2)$.
- $\mathsf{VesDerive}(\mathsf{vk}, \mathsf{apk}, \tau, \{\gamma_i, \boldsymbol{v}_i, \omega^{(i)}\}_{i=1}^{\ell})$: Parse $\omega^{(i)} = (\omega_{i,1}, \omega_{i,2}, \omega_{i,3})$. Choose $\tilde{r}$, $\tilde{t} \leftarrow_{\$} \mathbb{Z}_N$ and $\tilde{R}_3, \tilde{R}_3', \tilde{R}_3'' \leftarrow \mathbb{G}_3$. Compute $\omega_1 \leftarrow \left(\prod_{i \in [\ell]} \omega_{i,1}^{\gamma_i}\right) \cdot (u^{\tau} v)^{\tilde{r}} \cdot y^{\tilde{t}} \cdot \tilde{R}_3$, $\omega_2 \leftarrow \left(\prod_{i \in [\ell]} \omega_{i,2}^{\gamma_i}\right) \cdot g^{\tilde{r}} \cdot \tilde{R}_3'$, and $\omega_3 \leftarrow \left(\prod_{i \in [\ell]} \omega_{i,3}^{\gamma_i}\right) \cdot g^{\tilde{t}} \cdot \tilde{R}_3''$. Output $\omega = (\omega_1, \omega_2, \omega_3)$.
- $\mathsf{Adj}(\mathsf{ask}, \mathsf{apk}, \mathsf{vk}, \omega, \tau, \boldsymbol{v})$: Parse $\omega = (\omega_1, \omega_2, \omega_3)$. Return $\perp$ if $\mathsf{VesVrfy}(\mathsf{apk}, \mathsf{vk}, \tau, \boldsymbol{v}, \omega) \to 0$. Choose $\tilde{r} \leftarrow_{\$} \mathbb{Z}_N$ and $\tilde{R}_3, \tilde{R}_3' \leftarrow_{\$} \mathbb{G}_3$. Compute $\sigma_1 \leftarrow (\omega_1 / \omega_3^{\beta}) \cdot (u^{\tau} v)^{\tilde{r}} \cdot \tilde{R}_3$ and $\sigma_2 \leftarrow \omega_2 \cdot g^{\tilde{r}} \cdot \tilde{R}_3'$. Output $\sigma = (\sigma_1, \sigma_2)$.

Remark 5.1. We note that a HS scheme $HS = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Derive})$ is the ALP12 scheme [17, Sect. 4]. We build our VEHS scheme upon them by introducing AdjGen, Create, $\mathsf{VesVrfy}$, $\mathsf{VesDerive}$, and Adj.

Security. We show correctness and security of our VEHS scheme.

Theorem 5.1 (Informal). VEHS *is correct, resolution-independent, and extractable unconditionally.* VEHS *is unforgeable and opaque under the assumptions* **LW1′**, **LW2**, **ALP3**, *and* **ALP4**.

Due to space limit, we defer the proofs of correctness and security to the full version.

References

1. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM **28**(6), 637–647 (1985)
2. Ben-Or, M., Goldreich, O., Micali, S., Rivest, R.L.: A fair protocol for signing contracts. IEEE Trans. IT **36**(1), 40–46 (1990)
3. Boneh, D., Naor, M.: Timed commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)
4. Bahreman, A., Tygar, J.D.: Certified electronic mail. In: NDSS 1994, pp. 3–19 (1994)
5. Coffey, T., Saidha, P.: Non-repudiation with mandatory proof of receipt. ACM SIGCOMM Comput. Commun. Rev. **26**(1), 6–17 (1996)
6. Cox, B., Tygar, J.D., Sirbu, M.: NetBill security and transaction protocol. USENIX Workshop on Electronic Commerce 1995, 77–88 (1995)
7. Deng, R.H., Gong, L., Lazar, A.A., Wang, W.: Practical protocols for certified electronic mail. J. Netw. Syst. Manage. **4**(3), 279–297 (1996)

8. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: ACM CCS 1997, pp. 7–17 (1997)
9. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998)
10. Asokan, N., Shoup, V., Waidner, M.: Asynchronous protocols for optimistic fair exchange. In: IEEE Symposium on S&P 1998, pp. 86–99 (1998)
11. Dodis, Y., Reyzin, L.: Breaking and repairing optimistic fair exchange from PODC 2003. In: Digital Rights Management Workshop 2003, pp. 47–54 (2003)
12. Dodis, Y., Lee, P.J., Yum, D.H.: Optimistic fair exchange in a multi-user setting. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 118–133. Springer, Heidelberg (2007)
13. Huang, X., Mu, Y., Susilo, W., Wu, W., Xiang, Y.: Further observations on optimistic fair exchange protocols in the multi-user setting. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 124–141. Springer, Heidelberg (2010)
14. Micali, S.: Simple and fast optimistic protocols for fair electronic exchange. In: PODC, pp. 12–19 (2003)
15. Küpçü, A., Lysyanskaya, A.: Usable optimistic fair exchange. Comput. Netw. **56**(1), 50–63 (2012)
16. Freeman, D., Katz, J., Waters, B., Boneh, D.: Signing a linear subspace: signature schemes for network coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009)
17. Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: new privacy definitions and constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 367–385. Springer, Heidelberg (2012)
18. Zhu, H., Bao, F.: Stand-alone and setup-free verifiably committed signatures. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 159–173. Springer, Heidelberg (2006)
19. Camenisch, J., Damgård, I.: Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, p. 331. Springer, Heidelberg (2000)
20. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
21. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
22. Rückert, M., Schröder, D.: Security of verifiably encrypted signatures and a construction without random oracles. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 17–34. Springer, Heidelberg (2009)
23. Nishimaki, R., Xagawa, K.: Verifiably Encrypted Signatures with Short Keys Based on the Decisional Linear Problem and Obfuscation for Encrypted VES. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 405–422. Springer, Heidelberg (2013)
24. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

25. Lee, K., Lee, D.H., Yung, M.: Sequential aggregate signatures with short public keys: design, analysis and implementation studies. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 423–442. Springer, Heidelberg (2013)
26. Lee, K., Lee, D.H., Yung, M.: Aggregating CL-Signatures revisited: extended functionality and better efficiency. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 171–188. Springer, Heidelberg (2013)
27. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (2007)
28. Fischlin, M., Lehmann, A., Schröder, D.: History-free sequential aggregate signatures. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 113–130. Springer, Heidelberg (2012)
29. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM **51**(4), 557–594 (2004)
30. Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: full domain hash from indistinguishability obfuscation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 201–220. Springer, Heidelberg (2014)
31. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
32. Gerbush, M., Lewko, A., O'Neill, A., Waters, B.: Dual form signatures: an approach for proving security from static assumptions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 25–42. Springer, Heidelberg (2012)
33. Desmedt, Y.: Computer security by redefining what a computer is. In: NSPW 1993, pp. 160–166 (1993)
34. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
35. Attrapadung, N., Libert, B.: Homomorphic network coding signatures in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 17–34. Springer, Heidelberg (2011)
36. Catalano, D., Fiore, D., Warinschi, B.: Adaptive pseudo-free groups and applications. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 207–223. Springer, Heidelberg (2011)
37. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 680–696. Springer, Heidelberg (2012)
38. Attrapadung, N., Libert, B., Peters, T.: Efficient completely context-hiding quotable and linearly homomorphic signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 386–404. Springer, Heidelberg (2013)
39. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
40. Boneh, D., Freeman, D.M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) Public-Key Cryptography – PKC 2011, vol. 6571, pp. 1–16. Springer, Heidelberg (2011)
41. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure network coding over the integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer, Heidelberg (2010)

42. Agrawal, S., Boneh, D., Boyen, X., Freeman, D.M.: Preventing pollution attacks in multi-source network coding. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 161–176. Springer, Heidelberg (2010)
43. Agrawal, S., Boneh, D.: Homomorphic MACs: MAC-based integrity for network coding. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 292–305. Springer, Heidelberg (2009)
44. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
45. Freeman, D.M.: Improved security for linearly homomorphic signatures: a generic framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012)
46. Huang, Q., Yang, G., Wong, D.S., Susilo, W.: Ambiguous optimistic fair exchange. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 74–89. Springer, Heidelberg (2008)
47. Garay, J.A., Jakobsson, M., MacKenzie, P.D.: Abuse-free optimistic contract signing. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 449–466. Springer, Heidelberg (1999)
48. Zhou, J., Gollmann, D.: A fair non-repudiation protocol. In: IEEE Symposium on S&P 1996, pp. 55–61 (1996)
49. Calderon, T., Meiklejohn, S., Shacham, H., Waters, B.: Rethinking verifiably encrypted signatures: a gap in functionality and potential solutions. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 349–366. Springer, Heidelberg (2014)
50. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. IT **31**(4), 469–472 (1985)