Des. Codes Cryptogr. (2015) 77:61–98
DOI 10.1007/s10623-014-9986-9

Verifiably encrypted signatures with short keys based on the decisional linear problem and obfuscation for encrypted VES

Ryo Nishimaki · Keita Xagawa

Received: 24 September 2013 / Revised: 12 May 2014 / Accepted: 25 May 2014 /
Published online: 24 June 2014

Abstract Verifiably encrypted signatures (VES) are encrypted signatures under a public key of a trusted third party. We can verify their validity without decryption. VES has useful applications such as online contract signing and optimistic fair exchange. We propose a VES scheme that is secure under the decisional linear (DLIN) assumption in the standard model. We also propose new obfuscators for encrypted signatures (ES) and encrypted VES (EVES) that are secure under the DLIN assumption. All previous VES schemes in the standard model are either secure under standard assumptions (such as the computational Diffie–Hellman assumption) with large verification (or secret) keys or secure under *non-standard dynamic q-type assumptions* (such as the q-strong Diffie–Hellman extraction assumption) with short verification keys. Our scheme is the first VES scheme with short verification (and secret) keys secure under *the DLIN assumption* (standard assumption). We construct new obfuscators for ES/EVES as byproducts of our new VES scheme. They are more efficient than previous obfuscators with respect to public key size. Previous obfuscators for EVES are secure under non-standard assumption and use zero-knowledge (ZK) proof systems and Fiat–Shamir heuristics to obtain non-interactive ZK, i.e., its security is considered in the random oracle model. Thus, our scheme also has an advantage with respect to assumptions and the security model. Our new obfuscator for ES is obtained from our new obfuscator for EVES.

Communicated by C. Padro.

An extended abstract of this paper appeared in Public-Key Cryptography—PKC 2013—16th International Conference on Practice and Theory in Public-Key Cryptography, LNCS 7778, pp 405–422. This is the full version.

R. Nishimaki (✉) · K. Xagawa
3-9-11, Midori-cho, Musashino-shi, Tokyo 180-8585, Japan
e-mail: nishimaki.ryo@lab.ntt.co.jp

K. Xagawa
e-mail: xagawa.keita@lab.ntt.co.jp

Keywords Verifiably encrypted signature · Obfuscation · Encrypted verifiably encrypted signature · Decisional linear assumption

Mathematics Subject Classification 94A60 Cryptography

1 Introduction

1.1 Background

In verifiably encrypted signature (VES) schemes, we consider signers, verifiers, and *a trusted third party, called the adjudicator*. A signer generates a signature, encrypts it under the public key of the adjudicator, and adds extra content to make it verifiable without decryption. The adjudicator can recover ordinary signatures from encrypted ones by using its decryption key.

The concept of VES was introduced by Boneh et al. [12], who proposed the first VES scheme based on the Boneh-Lynn-Shacham signature scheme in the random oracle model (ROM) [10]. VES schemes have useful and important applications such as online contract signing and optimistic fair exchange [2,3]. Suppose a user, Alice, wants to buy digital goods from a company online. Alice gives the company her VES for a contract instead of paying money and the company returns the requested digital goods if it received a valid VES. Alice sends an ordinary signature to the company if she receives the goods. If a malicious company does not return the requested goods when it receives the VES, Alice can claim that the VES is of no use for the contract since it is encrypted. If malicious Alice does not return an ordinary signature when she receives the goods, the company sends the encrypted signature together with the transcript to the adjudicator and the adjudicator extracts an ordinary signature from the VES by using the secret key of the adjudicator and returns it to the company. The adjudicator is offline, that is, it should be active only when malicious Alice cheats the company. Fuchsbauer used a certain kind of VES to construct delegatable anonymous credentials [30]. Anonymous credentials are useful for access control [5]. In some systems with access control, users must prove having the required credentials issued by an authority to use the system. The authority may want to delegate its right to other entities to avoid centralization of power.

Lu et al. [45] proposed a VES scheme that is secure under the computational Diffie–Hellman (CDH) assumption in the standard model, but the verification key is quite long. Rückert and Schröder [48] proposed a VES scheme with short verification keys, but its security relies on a non-standard q-type assumption, called the q-strong Diffie–Hellman (q-SDH) extraction assumption. They did not prove its hardness in the generic group model [50]. Thus, there is no VES scheme that achieves a constant size verification key and signature based on standard assumptions.

Program obfuscation and encrypted signature/VES. An Encrypted VES (EVES) is an extension of an encrypted signature (ES) proposed by Hada [41]. The ES/EVES functionalities output an encryption of a signature/VES. They do not encrypt messages, but can be used as building blocks of signcryption functionalities, as pointed out by Hada [41]. In order to show this application, Hada proposed the notion of Encrypted-Signature-then-Encryption (EStE). In it, we first compute a signature for a message, then encrypt the signature. Finally, we encrypt both the message and the encrypted signature. The combination of the first and second steps is the ES functionality, so Hada claims that the ES functionality is useful for signcryption although the situation is somewhat different (In EStE, a signature is doubly encrypted).

We explain how to use obfuscators (explained below) for ES/EVES functionalities in a realistic scenario in this paragraph. If Alice uses free web-mail services to send mail to Bob on low computational power devices, such as smart-phones, and her web browsers do not have enough resources to sign messages and encrypt them with Bob's public key, then she would want web-mail providers to carry out its process instead of her. *However, she does not want to reveal her signing key.* The obfuscation for ES/EVES will provide a solution. A program obfuscator is an algorithm that transforms a program into a completely unintelligible program whose functionality is the same as the original one [4,40]. Informally speaking, obfuscators should guarantee that what is efficiently computed given an obfuscated program is nothing more than what is computed given *black-box access* to the original program. *This means that no adversary can obtain non-trivial information about the original program.* If Alice provides an obfuscated program for ES/EVES functionality, then she can securely delegate her signing capability to web-mail providers. Moreover, in a situation in which president Alice is on vacation wants to have vice president Carol sign contracts for Bob (only Alice to Bob) instead of her, Alice can provide Carol an obfuscated program for EVES functionality. *In this scenario, Carol cannot obtain any information about the secret key of Alice due to the property of obfuscation. This is a strong motivation of obfuscation for the ES/EVES functionalities.*

In Hada's obfuscator for an ES, if a malicious party has access to Bob's decryption key, then Alice's signing key is extracted from the obfuscated program [41]. However, in our obfuscator for an EVES, even such a malicious party cannot extract Alice's key due to the existence of the adjudicator's key. Thus, obfuscators for an EVES have useful applications.

The problem of program obfuscation is very attractive in the area of cryptology from both the theoretical and practical points of view since program obfuscators completely hide non-trivial information encoded into programs (for example, signing keys of signature/VES) and have many cryptographic applications as pointed out in [4]. A few positive results are shown for cryptographic functionalities [22,41–43].

Unfortunately, in the seminal work by Barak et al. [4], they showed the impossibility result for general-purpose obfuscation. Many other impossibility results have been shown in various settings [6,21,38–40,42,54]. There are a few positive results for very simple functionalities, such as point functions [16–19,46,54], proximity testing [27], testing hyperplane membership [20]. In order to sidestep broad impossibility results, Hohenberger et al. [43] and Hofheinz et al. [42] independently proposed a new definition of secure obfuscation for cryptographic purposes, *average-case secure obfuscation* of randomized functionalities. Moreover, Hohenberger et al. [43] proposed the first (average-case secure) obfuscator for a complicated cryptographic functionality, re-encryption functionality and Chandran et al. [22] proposed an obfuscator for functional re-encryption functionality.

Hada [41] proposed a secure obfuscator for an ES functionality and its application to signcryption. His scheme is secure under the decisional linear (DLIN) assumption in the standard model, but the verification key size is quite large. Cheng et al. [23] proposed a secure obfuscator for an EVES functionality at ProvSec'11. Their VES scheme and obfuscator for EVES use zero-knowledge (ZK) proofs and Fiat–Shamir heuristics to crash the ZK proofs into non-interactive zero-knowledge (NIZK) proofs. That is, their scheme and obfuscator are *secure in the ROM*. Furthermore, they used a non-standard assumption, called exponent 3-weak DH assumption, to prove the unforgeability of their scheme and did not prove the opacity (explained in the next section), which is required for secure VES schemes, of their scheme.

In general, obfuscators for ES/EVES can be obtained from fully homomorphic encryption (FHE) schemes [33]. However, current FHE schemes are still inefficient, though many

Table 1 Summary of previous schemes and ours for VES

Reference	Key size (vk/sk)	VES size	Assumptions	ROM
BGLS [12]	$1\mathbb{G}/1\mathbb{Z}_p$	$2\mathbb{G}$	CDH	Yes
ZSS [55]	$2\mathbb{G}/2\mathbb{Z}_p$	$1\mathbb{G}$	CDH	Yes
LOSSW [44]	$O(\lambda)\mathbb{G}(> 160\mathbb{G})/1\mathbb{Z}_p$	$3\mathbb{G}$	CDH	No
RS [48]	$4\mathbb{G}/2\mathbb{Z}_p$	$2\mathbb{G} + 1\mathbb{Z}_p$	q-strong DH extraction	No
This work	$16\mathbb{G} + 1\mathbb{G}_T/3\mathbb{G}$	$12\mathbb{G} + 2\mathbb{Z}_p$	*DLIN*	No

Table 2 Summary of previous obfuscators for encrypted ES/EVES

Reference	ES/EVES	Key size (vk)	ROM	Assumptions
Hada [41]	ES	$O(\lambda)$	No	DLIN
CZZ [23]	EVES	$O(\lambda)$	Yes	DLIN + Exponent 3-weak DH
This work	ES	$O(1)$	No	DLIN
This work	EVES	$O(1)$	No	DLIN

improvements have been proposed [13–15,24,26,34–37,51]. Therefore we do not rely on expensive FHE schemes but directly construct obfuscators for ES/EVES.

Very recently, Garg et al. [32] proposed an indistinguishability obfuscator for all polynomial-sized circuits by using multilinear maps [31]. The notion of indistinguishability obfuscation was proposed by Barak et al. [4] and is a weaker notion than the black-box obfuscation. Their construction is quite elegant, but it is a generic construction and uses multilinear maps and an ad hoc non-standard assumption. Thus, their obfuscator is not efficient.

1.2 Our contributions and constructions

We propose a VES scheme based on the *DLIN assumption in the standard model*. The main advantages over previous VES schemes are as follows.

1. It is secure under a standard (i.e., not q-type) assumption in the standard model.
2. The verification key and signature size are small (constant).

As a by-product of our VES scheme, we construct secure obfuscators for an ES/EVES functionality based on the DLIN assumption in the standard model. Main advantages of our obfuscators for an ES/EVES over previous obfuscators for an ES/EVES are as follows. They are *secure under the DLIN assumption in the standard model with short verification keys*.

Comparison and related work. Comparisons of our results with previous results of VES schemes and obfuscators for a ES/EVES are shown in Tables 1 and 2, respectively. Let λ denote the security parameter. The CDH assumption stands for the CDH assumption in bilinear groups. There has been no VES scheme and obfuscator for ES/EVES that are secure under standard assumptions in the standard model with short verification keys prior to ours. The VES scheme by Lu et al. requires a large verification key but its signature size is small and its security is based on a standard CDH assumption, so one may believe that the scheme of Lu et al. is better than ours in terms of signature size. However, we believe it is incomparable with our new scheme and we showed a tradeoff between the verification key size and signature size. Rückert proposed a VES scheme based on the full-domain hash RSA signature, but it is

secure in the ROM [47]. Rückert et al. [49] proposed generic constructions for a VES without NIZKs, pairings, and ROM. Their construction is very insightful, but their schemes use an extra adjudication setup phase and Merkle trees, so they need to set-up large parameters and have large keys (non-constant size).

Our construction technique. Loosely speaking, a VES scheme consists of a signature scheme and an encryption scheme as Lu et al. and Rückert and Schröder stated [45,48]. We use Waters' signature scheme presented at CRYPTO'09 [53] as an underlying signature scheme. We call it Waters' dual signature in this paper to distinguish it from Waters' signature at Eurocrypt'05 [52]. Someone may believe that a combination of Waters' dual signature and ElGamal encryption easily yields a secure VES scheme under the DLIN assumption, *but this is not the case*. The reason is as follows. We can prove unforgeability of a VES scheme by relying on the unforgeability of the underlying signature scheme as in previous schemes [12,45,48], *but the opacity is non-trivial*. The opacity means that it is difficult to extract an ordinary signature from a VES, i.e., decrypt a VES. Moreover, it is highly non-trivial whether we can prove opacity from standard assumptions. The reason is as follows. The VES scheme of Lu et al. is a combination of Waters' signature (Eurocrypt'05) [52] and ElGamal encryption scheme, and they proved its opacity from the aggregate extraction assumption [12] (fortunately, it is equivalent to the CDH assumption [25]). On the other hand, the VES scheme of Rückert and Schröder [48] is a combination of the Boneh–Boyen signature [7] and ElGamal encryption schemes, but they proved its opacity from the q-strong DH extraction assumption, which is a *stronger assumption than that of the underlying Boneh–Boyen signature scheme*.

Our construction is a combination of Waters' dual signature and ElGamal encryption schemes. We encrypt only signature elements *related to signing keys*. The security proof of Waters' dual signature is somewhat different from that of many known secure signature schemes, such as Boneh–Boyen [7], and Waters [52], so we must use a somewhat different proof strategy from that of Lu et al. and Rückert and Schröder. Waters' dual signature has two types of signatures, standard signature (Type A) and *semi-functional* signature (Type B). Semi-functional signatures also pass the verification algorithm as standard ones and are *indistinguishable from them* [53]. We extend the proof strategy of this dual form signature technique to prove opacity. First, we use Type B signatures as normal signatures generated by a normal signing algorithm and Type A signatures are used for simulation. Both Type A and B signatures are valid signatures and there is no essential difference in terms of functionality as long as a normal verification algorithm is used. We use this swapping of roles since we do not know how to prove that an adversary cannot extract a valid Type A signature from a given VES when the oracle answers Type A signatures.

In the experiment of the opacity explained in Sect. 2.4, an adversary can output a signature and message pair such that the message is queried to an oracle which returns a VES for the queried message. This causes the main difficulty in proving the opacity since the adversary may output a re-randomized signature obtained using valid signatures from oracles. Unfortunately, Waters' dual signature is re-randomizable. Thus, we modify Waters' dual signature scheme and make it *strongly* unforgeable. Strong unforgeability guarantees that an adversary cannot output a forgery even for a queried message, so it must hold that if the adversary output valid signature for a queried message in the experiment of opacity, then the signature is identical to the signature generated by the VES creation oracle (otherwise, contradict to strong unforgeability). This fact can be used to prove the opacity of our scheme.

In the proof of opacity, we must simulate two oracles. One is the creation oracle, which answers VESs for queried messages. The other is the adjudication oracle, which extracts ordinary signatures from queried messages and VESs and returns them. When we answer

only the encryption of Type B signature for VES creation queries of the adversary, we can prove that the adversary cannot extract Type B signature from a VES under the aggregate extraction assumption. This is why we swap the roles of Type A signatures for that of Type B signatures. We have no way to prove that when we answer only the encryption of a Type A signature for VES creation queries of an adversary, the adversary cannot extract a Type A signature from a VES.

Thus, we can prove that no adversary can output a valid signature for a *queried message* to the VES creation oracle. For non-queried messages, we can use the proof technique for the unforgeability of dual form signatures. We prove that no adversary can output a Type A signature when the oracle returns Type B signatures (VESs).

Next, we change the type of signatures used to generate VESs, which are answered by the VES creation oracle. Answers from the adjudication oracle depend on the type of the VES creation oracle. Thus, we prove that the view of the adversary is indistinguishable even if the type of each answer is changed from Type B to Type A for each query. This order of change is reverse to the original proof, but it is not a major difference. Finally, we prove that no adversary can output a Type B signature when the oracle returns Type A signatures (VESs).

Secure obfuscations for ES and EVES based on Waters' dual signature scheme are also non-trivial because the signing keys of this scheme consist of multiple group elements, and the signing algorithm computes exponentiation of the signing keys with randomness in contrast to Waters' signature presented at Eurocrypt'05, whose signing key is only one group element and signing algorithm only multiplies it by other group elements [52]. We overcome this hurdle by using the homomorphic property of the ElGamal and linear encryption schemes [9]. Cheng et al. use the linear encryption scheme for not only encryption of a VES but also the construction of the VES, so their VES scheme cannot check the validity of ciphertext by using only the pairing technique and they need (NI)ZK. We do not need (NI)ZK because our VES scheme uses the ElGamal encryption scheme and can verify the validity of VES by using only pairings.

2 Preliminaries

Notations and conventions. For any $n \in \mathbb{N}\backslash\{0\}$, let $[n]$ be the set $\{1, \ldots, n\}$. When D is a random variable or distribution, $y \xleftarrow{\mathsf{R}} D$ denotes that y is randomly selected from D according to its distribution. If S is a set, then $x \xleftarrow{\mathsf{U}} S$ denotes that x is uniformly selected from S. We denote y is a set, defined or substituted by z by $y := z$. When b is a fixed value, $A(x) \to b$ (e.g., $A(x) \to 1$) denotes the event in which machine (or algorithm) A outputs b on input x. We say that positive function $f : \mathbb{N} \to \mathbb{R}$ is negligible in $\lambda \in \mathbb{N}$ if for every constant $c \in \mathbb{N}$ there exists $k_c \in \mathbb{N}$ such that $f(\lambda) < \lambda^{-c}$ for any $\lambda > k_c$. Let $\mathcal{X} = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ denote two ensembles of random variables indexed by λ.

Definition 1 We say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable if for every non-uniform probabilistic polynomial-time (PPT) algorithm D,

$$|\Pr[D(X_\lambda) = 1] - \Pr[D(Y_\lambda) = 1]|$$

is negligible in λ.

We write $\mathcal{X} \overset{\mathsf{c}}{\approx} \mathcal{Y}$ to denote that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable.

2.1 Cryptographic bilinear maps (or pairings)

We consider cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p. A bilinear map is an efficient mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfying the following properties:

bilinearity: For all $g \in \mathbb{G}$ and $a, b \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$.
non-degeneracy: If g generates $\mathbb{G}$, then $e(g, g) \neq 1$.

Let $\mathcal{G}_{\mathsf{bmp}}$ be a standard parameter generation algorithm that takes security parameter λ as input and outputs parameters $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. That is, $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$ and Γ is a description of groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p equipped with efficient bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Here, g is a generator in $\mathbb{G}$. We often omit common parameters Γ.

2.2 Complexity assumptions

We review several complexity assumptions that are used to prove the security of cryptographic primitives.

Definition 2 (*Discrete Logarithm (DL) assumption*) The DL problem in bilinear groups is to compute x, given $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$ and g^x for $x \xleftarrow{\mathsf{U}} \mathbb{Z}_p$. The advantage is defined as follows.

$$\mathsf{Adv}^{\mathsf{dl}}_{\mathcal{A}}(\lambda) := \Pr[z = x \mid \Gamma \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda); x \xleftarrow{\mathsf{U}} \mathbb{Z}_p; z \xleftarrow{\mathsf{R}} \mathcal{A}(\Gamma, g^x)].$$

We say that the DL assumption holds in bilinear groups if the DL problem in bilinear groups is hard, that is, for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{dl}}_{\mathcal{A}}(\lambda)$ is negligible in λ.

Definition 3 (*CDH assumption*) The CDH problem in bilinear groups is to compute g^{xy}, given $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$ and (g^x, g^y) for $x, y \xleftarrow{\mathsf{U}} \mathbb{Z}_p$. The advantage is defined as follows.

$$\mathsf{Adv}^{\mathsf{cdh}}_{\mathcal{A}}(\lambda) := \Pr[z = g^{xy} \mid \Gamma \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda); x, y \xleftarrow{\mathsf{U}} \mathbb{Z}_p; z \xleftarrow{\mathsf{R}} \mathcal{A}(\Gamma, g^x, g^y)].$$

We say that the CDH assumption holds in bilinear groups if the CDH problem in bilinear groups is hard, that is, for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{cdh}}_{\mathcal{A}}(\lambda)$ is negligible in λ.

Definition 4 (*DLIN assumption*) The DLIN problem in bilinear groups is to guess $\beta \in \{0, 1\}$, given $(\Gamma, g, f, v, g^x, f^y, Q_\beta) \xleftarrow{\mathsf{R}} \mathcal{G}^{\mathsf{dlin}}_\beta(1^\lambda)$, where $\mathcal{G}^{\mathsf{dlin}}_\beta(1^\lambda)$: $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$, $f, v \xleftarrow{\mathsf{U}} \mathbb{G}$, $x, y \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, $Q_0 := v^{x+y}$, $Q_1 \xleftarrow{\mathsf{U}} \mathbb{G}$, return $(\Gamma, g, f, v, g^x, f^y, Q_\beta)$. The advantage is defined as follows.

$$\mathsf{Adv}^{\mathsf{dlin}}_{\mathcal{A}}(\lambda) := \left| \Pr\left[\mathcal{A}(\mathcal{I}) \to 1 \mid \mathcal{I} \xleftarrow{\mathsf{R}} \mathcal{G}^{\mathsf{dlin}}_0(1^\lambda)\right] - \Pr\left[\mathcal{A}(\mathcal{I}) \to 1 \mid \mathcal{I} \xleftarrow{\mathsf{R}} \mathcal{G}^{\mathsf{dlin}}_1(1^\lambda)\right] \right|.$$

We say that the DLIN assumption holds if the DLIN problem is hard, that is, for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{dlin}}_{\mathcal{A}}(\lambda)$ is negligible in λ.

Definition 5 (*Aggregate Extraction (AgExt) assumption*) The AgExt problem in bilinear groups is to compute g^{xy}, given $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$ and $(g^x, g^y, g^\beta, g^\delta, g^{xy+\beta\delta})$ for $x, y, \beta, \delta \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ [12,25]. The advantage is defined as follows.

$$\mathsf{Adv}^{\mathsf{agext}}_{\mathcal{A}}(\lambda) := \Pr\left[z = g^{xy} \;\middle|\; \begin{array}{l} \Gamma \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda); x, y, \beta, \delta \xleftarrow{\mathsf{U}} \mathbb{Z}_p; \\ z \xleftarrow{\mathsf{R}} \mathcal{A}(\Gamma, g^x, g^y, g^\beta, g^\delta, g^{xy+\beta\delta}) \end{array} \right].$$

We say that the AgExt assumption holds in bilinear groups if the AgExt problem in bilinear groups is hard, that is, for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{agext}}_{\mathcal{A}}(\lambda)$ is negligible in λ.

The AgExt assumption is equivalent to the CDH assumption, which is implied by the DLIN assumption.

Theorem 1 (The CDH and AgExt are equivalent [25]) *The AgExt and CDH problems are Karp reducible to each other with $O(1)$ computation.*

2.3 Cryptographic primitives

Signature. Signature scheme SIG consists of three PPT algorithms $\mathsf{SIG} = \mathsf{SIG}.\{\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy}\}$ satisfying the following properties.

Key Generation: $\mathsf{SIG.Gen}$ takes as input security parameter 1^λ and outputs a pair of keys, that is, $(vk, sk) \xleftarrow{R} \mathsf{SIG.Gen}(1^\lambda)$. They are called the (public) verification key and the (private) signing key, respectively.

Sign: $\mathsf{SIG.Sign}$ takes as input a signing key and a message and outputs signature σ. That is, $\sigma \xleftarrow{R} \mathsf{SIG.Sign}(sk, m)$, where $m \in \mathcal{M}_{vk}$ and $\mathcal{M}_{vk}$ is a message space defined by the verification key.

Verification: $\mathsf{SIG.Vrfy}$ is deterministic, takes as input a verification key, a message, and a signature, and outputs bit b. If $b = 1$, then the signature is valid. Else, it is invalid. That is, $\mathsf{SIG.Vrfy}(vk, \sigma, m) \to b$.

It is required that $\forall\lambda\ \forall(vk, sk) \xleftarrow{R} \mathsf{SIG.Gen}(1^\lambda)\ \forall m \in \mathcal{M}_{vk}\ \ \mathsf{SIG.Vrfy}(vk, \mathsf{SIG.Sign}(sk, m), m) \to 1$. Signature scheme $\mathsf{SIG} = \mathsf{SIG}.\{\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy}\}$ is said to be existentially unforgeable under adaptive chosen message attacks (EUF-CMA) if the advantage of the following game is negligible.

1. Setup: A challenger generates $(vk, sk) \xleftarrow{R} \mathsf{SIG.Gen}(1^\lambda)$ and sends vk to an adversary.
2. Queries: The adversary sends message $M_i \in \mathcal{M}_{vk}$ to the challenger and answers $\sigma_i \xleftarrow{R} \mathsf{SIG.Sign}(sk, M_i)$. These queries are sent adaptively for $i = 1$ to n. Let Q be the set of messages $M_1, \ldots, M_k \in \mathcal{M}_{vk}$ queried by the adversary.
3. Output: The adversary outputs (m^*, σ^*). If it holds that $\mathsf{SIG.Vrfy}(vk, \sigma^*, m^*) \to 1$ and $m^* \notin Q$, then it is said that the adversary wins the game.

We define $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A}}(\lambda)$ to be the probability that an adversary $\mathcal{A}$ wins in the game.

Definition 6 (*Existentially Unforgeable against Adaptive Chosen Message Attacks*) Signature scheme SIG is existentially unforgeable against adaptive chosen message attacks if for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A}}(\lambda)$ is negligible in λ.

In the game of EUF-CMA, if we replace condition $m^* \notin Q$ with $(m^*, \sigma^*) \neq (\mathrm{M}_i, \sigma_i)$ for any i, then we say that the security is *strongly* existentially unforgeable (sEUF). We define $\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathcal{A}}(\lambda)$ to be the probability that $\mathcal{A}$ wins in the modified game.

Public key encryption. A public key encryption (PKE) scheme consists of three PPT algorithms $\mathsf{PKE}.\{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}\}$ satisfying the following properties.

Key Generation: $\mathsf{PKE.Gen}$ takes as input security parameter 1^λ and outputs a pair of keys, that is, $(pk, sk) \xleftarrow{R} \mathsf{PKE.Gen}(1^\lambda)$. They are called the public key and secret (decryption) key, respectively.

Encryption: $\mathsf{PKE.Enc}$ takes as input pk and plaintext m and outputs ciphertext c. That is, $c \xleftarrow{R} \mathsf{PKE.Enc}(pk, m)$. If we express that we explicitly use randomness r, then we use notation $c := \mathsf{Enc}(pk, m; r)$.

Decryption: $\mathsf{PKE.Dec}$ is deterministic, takes as input sk and c, and outputs plaintext m'. That is, $m' := \mathsf{PKE.Dec}(sk, c)$.

It is required $\forall\lambda\ \forall(pk, sk) \xleftarrow{R} \mathsf{PKE.Gen}(1^\lambda)\ \forall m\ \ m = \mathsf{PKE.Dec}(sk, \mathsf{PKE.Enc}(pk, m))$.

We present indistinguishability against chosen plaintext attacks (IND-CPA), that is the basic security of PKE.

Definition 7 (*IND-CPA security*) The model for proving the IND-CPA security of PKE against $\mathcal{A}$ is given as follows.

1. $\mathsf{PKE.Gen}$ is run with input 1^λ to generate keys pk and sk, and pk is given to $\mathcal{A}$.
2. $\mathcal{A}$ outputs challenge plaintexts (m_0, m_1) such that $|m_0| = |m_1|$.
3. Uniformly random bit b is chosen. $\mathcal{A}$ is given $c^* \xleftarrow{R} \mathsf{Enc}(pk, m_b)$.
4. $\mathcal{A}$ outputs a bit b' and wins if $b' = b$.

The advantage of $\mathcal{A}$ in the above game is defined as $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{A}}(\lambda) := \left|2\Pr[b' = b] - 1\right|$ for any security parameter λ. A PKE scheme is IND-CPA secure if for any PPT adversary $\mathcal{A}$, it holds that $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{A}}(\lambda)$ is negligible in λ.

Collision resistant hash functions (CRHF). Let $\mathcal{H} := \{H_k\}$ be a keyed hash family of functions $H_k : \{0,1\}^* \to \{0,1\}^n$ indexed by $k \in \mathcal{K}_\lambda$ where λ is a security parameter.

Definition 8 We say that $\mathcal{H}$ is (t, ϵ)-collision-resistant if for any $\mathcal{A}$ running in time t, we have that $\mathsf{Adv}^{\mathsf{crhf}}_{\mathcal{A},\mathcal{H}}(\lambda) := \Pr[m_0 \neq m_1 \wedge H_k(m_0) = H_k(m_1) \mid (m_0, m_1) \xleftarrow{R} \mathcal{A}(k)] < \epsilon$, where the probability is over the random choice of $k \in \mathcal{K}_\lambda$ and random coins of $\mathcal{A}$.

2.4 Verifiably encrypted signature (VES)

A VES scheme consists of seven PPT algorithms, $\{\mathsf{AdjGen}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Create}, \mathsf{VesVrfy}, \mathsf{Adj}\}$, satisfying the following properties.

Adjudicator Key Generation: AdjGen takes as input security parameter 1^λ and outputs a pair of keys for an adjudicator, that is, $(apk, ask) \xleftarrow{R} \mathsf{AdjGen}(1^\lambda)$.

Key Generation: Gen takes as input 1^λ and outputs a pair of keys for a signer, that is, $(vk, sk) \xleftarrow{R} \mathsf{Gen}(1^\lambda)$. They are called the verification key and the signing key, respectively.

Signing: Sign takes as input a signing key and a message and outputs signature σ. That is, $\sigma \xleftarrow{R} \mathsf{Sign}(sk, M)$, where $M \in \mathcal{M}_{vk}$ and $\mathcal{M}_{vk}$ is a message space defined by vk.

Verification: Vrfy is deterministic, takes as input vk, M, and σ, and outputs bit b. If $b = 1$ then the signature is valid. Else, it is invalid. That is, $\mathsf{Vrfy}(vk, \sigma, m) \to b$.

VES Creation: Create takes as input sk, apk, and M and outputs VES ω on M. That is, $\omega \xleftarrow{R} \mathsf{Create}(sk, apk, M)$.

VES Verification: $\mathsf{VesVrfy}$ is deterministic, takes as input apk, vk, ω, and M, and outputs bit b, that is, $\mathsf{VesVrfy}(apk, vk, \omega, M) \to b$.

Adjudication: Adj takes as input ask, apk, vk, ω, and M. If ω is valid, it extracts an plain signature σ on M and returns σ, that is $\sigma \xleftarrow{R} \mathsf{Adj}(ask, apk, vk, \omega, M)$ if $\mathsf{VesVrfy}(apk, vk, \omega, M) \to 1$.

It is required that $\forall\lambda\ \forall(apk, ask) \xleftarrow{R} \mathsf{AdjGen}(1^\lambda)\ \forall(vk, sk) \xleftarrow{R} \mathsf{Gen}(1^\lambda)\ \forall m \in \mathcal{M}_{vk}$ $\mathsf{VesVrfy}(apk, pk, \mathsf{Create}(sk, apk, M), M) \to 1$ and $\mathsf{Vrfy}(vk, \mathsf{Adj}(ask, apk, vk, \mathsf{Create}(sk, apk, M), M), M) \to 1$.

A VES scheme is secure if it satisfies the unforgeability and the opacity [12]. In experiments defined below, oracle $\mathcal{CO}(sk, apk, \cdot)$ returns VESs for queried messages, oracle $\mathcal{AO}(ask, apk, vk, \cdot, \cdot)$ extracts and returns signature for queried message/VES pairs, and Q_{C} and Q_{A} are sets of messages queried by the adversary to $\mathcal{CO}$ and $\mathcal{AO}$, respectively.

Definition 9 (*Unforgeability*) Experiments $\mathsf{Forge}^{\mathsf{ves}}_{\mathcal{A}}(\lambda)$ is defined as follows.

Experiment $\mathsf{Forge}^{\mathsf{ves}}_{\mathcal{A}}(\lambda)$
$(apk, ask) \xleftarrow{R} \mathsf{AdjGen}(1^\lambda)$;
$(vk, sk) \xleftarrow{R} \mathsf{Gen}(1^\lambda)$;
$(M^*, \omega^*) \xleftarrow{R} \mathcal{A}^{\mathcal{CO}(sk,apk,\cdot),\mathcal{AO}(ask,apk,vk,\cdot,\cdot)}(vk, apk)$;
Return 1 iff $\mathsf{VesVrfy}(apk, vk, \omega^*, M^*) \to 1$ and $M^* \notin Q_{\mathsf{C}}$ and $M^* \notin Q_{\mathsf{A}}$.

We say that a VES scheme is unforgeable if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ves\text{-}uf}}_{\mathcal{A}}(\lambda) := \Pr[\mathsf{Forge}^{\mathsf{ves}}_{\mathcal{A}}(\lambda) \to 1]$ is negligible in λ.

Definition 10 (*Opacity*) Experiment $\mathsf{Opac}_{\mathcal{A}}(\lambda)$ is defined as follows.

Experiment $\mathsf{Opac}_{\mathcal{A}}(\lambda)$
$(apk, ask) \xleftarrow{R} \mathsf{AdjGen}(1^\lambda)$;
$(vk, sk) \xleftarrow{R} \mathsf{Gen}(1^\lambda)$;
$(M^*, \sigma^*) \xleftarrow{R} \mathcal{A}^{\mathcal{CO}(sk,apk,\cdot),\mathcal{AO}(ask,apk,vk,\cdot,\cdot)}(vk, apk)$;
Return 1 iff $\mathsf{Vrfy}(vk, \sigma^*, M^*) \to 1$ and $M^* \notin Q_{\mathsf{A}}$.

We say that a VES scheme is opaque if for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{opac}}_{\mathcal{A}}(\lambda) := \Pr[\mathsf{Opac}_{\mathcal{A}}(\lambda) \to 1]$ is negligible in λ.

More properties of VES. Rückert and Schröder [48] proposed several properties of a VES to achieve a modular analysis for it. Rückert and Schröder [48,49] defined key-independence and extractability of a VES to prove its unforgeability and collusion-resistance. Key-independence means that a VES creation algorithm consists of a signature generation part and a transformation (into a VES) part and they are independent. Extractability means that if VES ω is valid, then the adjudicator can extract a valid (ordinary) signature σ except with negligible probability. Collusion-resistance means that no adversary can forge a VES even if the adjudicator is corrupted, i.e., the adversary obtains the secret decryption key of the adjudicator. Formal definitions are as follows.

Definition 11 (*Key-Independence of VES* [48]) Let a signer's private key sk consist of two independent elements $sk = (kisk, ssk)$ and let $vk = (kivk, svk)$ be the corresponding verification key pair. A VES scheme is key-independent if there exists an efficient (encryption) algorithm KIEnc such that the distribution of $\mathsf{KIEnc}(apk, kivk, kisk, \mathsf{Sign}(ssk, M), M)$ is perfectly indistinguishable from that of $\mathsf{Create}(sk, apk, M)$ for all $M \in \mathcal{M}_{vk}$.

Definition 12 (*Extractability of VES* [48]) Experiment $\mathsf{Extract}^{\mathsf{ves}}_{\mathcal{A}}$ is as follows.

Experiment $\mathsf{Extract}^{\mathsf{ves}}_{\mathcal{A}}(\lambda)$
$(apk, ask) \xleftarrow{R} \mathsf{AdjGen}(1^{\lambda})$;
$(M^*, \omega^*, vk^*) \xleftarrow{R} \mathcal{A}^{\mathcal{AO}(ask,apk,vk,\cdot,\cdot)}(apk)$;
$\sigma^* \xleftarrow{R} \mathsf{Adj}(ask, apk, vk^*, \omega^*, M^*)$;
Return 1 iff $\mathsf{VesVrfy}(apk, vk, \omega^*, M^*) \to 1$ and $\mathsf{Vrfy}(vk^*, \sigma^*, M^*) \to 0$.

A VES scheme is extractable if for any PPT $\mathcal{A}$, $\Pr[\mathsf{Extract}^{\mathsf{ves}}_{\mathcal{A}}(\lambda) \to 1]$ is negligible in λ.

Definition 13 (*Collusion-Resistance* [48,49]) Experiment $\mathsf{Collusion}^{\mathsf{ves}}_{\mathcal{A}}$ is as follows.

Experiment $\mathsf{Collusion}^{\mathsf{ves}}_{\mathcal{A}}(\lambda)$
$(apk, ask) \xleftarrow{R} \mathsf{AdjGen}(1^{\lambda})$;
$(vk, sk) \xleftarrow{R} \mathsf{Gen}(1^{\lambda})$;
$(M^*, \omega^*) \xleftarrow{R} \mathcal{A}^{\mathcal{CO}(sk,apk,\cdot)}(apk, ask, vk)$;
Return 1 iff $\mathsf{VesVrfy}(apk, vk, \omega^*, M^*) \to 1$ and $M^* \notin Q_{\mathsf{C}}$.

A VES scheme is collusion-resistant if for any PPT $\mathcal{A}$, $\Pr[\mathsf{Collusion}^{\mathsf{ves}}_{\mathcal{A}}(\lambda) \to 1]$ is negligible in λ.

Rückert and Schröder [48] called this notion "abuse-freeness", but later Rückert, Schneider, and Schröder [49] renamed it "collusion-resistance" to avoid confusion. If $\mathcal{A}$ is allowed to select the public key in the above game, then we call it *strong collusion-resistance*.

Rückert and Schröder showed the following theorems.

Theorem 2 ([48]) *Let a VES scheme be an extractable and key-independent verifiably encrypted signature scheme. The VES scheme is unforgeable if and only if the underlying signature scheme is unforgeable.*

Theorem 3 ([48]) *A key-independent, extractable, and secure VES scheme is collusion-resistance if the underlying signature scheme is unforgeable.*

In fact, we can prove strong collusion-resistance in the above theorem since adversaries cannot forge VESs due to the unforgeability of underlying signature schemes even if an adjudicator public key is adversarially selected. Dodis, Lee, and Yum consider strong collusion-resistance (though they did not use the name) to construct optimistic fair exchange protocols [28], thus our VES scheme can be used to construct optimistic fair exchange protocols.

3 Strongly unforgeable Waters' dual signature

In this section, we propose a strongly secure version of Waters' dual signature scheme since we use it as a building block of our VES scheme.

Waters' dual signature scheme. We review a signature scheme proposed by Waters [53] since we use it as an essential building block. However, we add a few minor changes to fit the scheme to our VES scheme. We explain the differences between the original scheme and ours.

$\mathsf{Wd.Gen}(1^\lambda, \Gamma)$: On input security parameter λ and $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{R} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$ (hereafter we often omit input 1^λ), it selects generators $v, v_1, v_2, w, u, h \xleftarrow{U} \mathbb{G}$ and exponents $a_1, a_2, b, \alpha \xleftarrow{U} \mathbb{Z}_p$, computes $\tau_1 := v v_1^{a_1}$, $\tau_2 := v v_2^{a_2}$, and outputs

$$VK := (\Gamma, g^b, g^{a_1}, g^{a_2}, g^{ba_1}, g^{ba_2}, v, v_1, v_2, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, u, h, e(g,g)^{\alpha a_1 b})$$
$$SK := (VK, g^\alpha, g^{\alpha a_1}, g^{a_1 a_2}).$$

$\mathsf{Wd.Sign}(SK, M)$: On input message $M \in \mathbb{Z}_p$, it selects $r_1, r_2, z_1, z_2, \gamma, \mathsf{stag} \xleftarrow{U} \mathbb{Z}_p$, sets $r := r_1 + r_2$, computes

$$\sigma_0 := (u^M w^{\mathsf{stag}} h)^{r_1} \qquad \sigma_1 := g^{\alpha a_1} v^r \cdot g^{-a_1 a_2 \gamma} \qquad \sigma_2 := g^{-\alpha} v_1^r g^{z_1} \cdot g^{a_2 \gamma}$$
$$\sigma_3 := (g^b)^{-z_1} \qquad \sigma_4 := v_2^r g^{z_2} \cdot g^{a_1 \gamma} \qquad \sigma_5 := (g^b)^{-z_2}$$
$$\sigma_6 := (g^b)^{r_2} \qquad \sigma_7 := g^{r_1},$$

and outputs $\mathsf{sig} := (\sigma_0, \sigma_1, \ldots, \sigma_7, \mathsf{stag})$.

$\mathsf{Wd.Vrfy}(VK, \mathsf{sig}, M)$: On input VK, M, and sig, it outputs 1 if and only if it holds that

$$e(u^M w^{\mathsf{stag}} h, \sigma_7) = e(g, \sigma_0)$$
$$e(g^b, \sigma_1)\, e(g^{ba_1}, \sigma_2) e(g^{a_1}, \sigma_3) = e(\tau_1, \sigma_6)\, e(\tau_1^b, \sigma_7)$$
$$e(g^b, \sigma_1)\, e(g^{ba_2}, \sigma_4)\, e(g^{a_2}, \sigma_5) = e(\tau_2, \sigma_6)\, e(\tau_2^b, \sigma_7)\, e(g,g)^{\alpha a_1 b}.$$

The differences from the original scheme are as follows. In original Waters' dual signature scheme,

1. The verification equation is only one equation and probabilistic.
2. Values v, v_1, v_2 are included in secret keys.
3. Value $g^{a_1 a_2}$ is not included in the signing key.
4. The (normal) signing algorithm does not multiply $g^{-a_1 a_2 \gamma}$, $g^{a_2 \gamma}$, $g^{a_1 \gamma}$ in σ_1, σ_2, σ_4, respectively, that is, the signing algorithm outputs Type A signatures, as explained below.

First, note that there are two types of signatures in Waters' dual signature scheme, type A (if $\gamma = 0$) and Type B (if $\gamma \neq 0$). Both types are valid signatures. The modified verification equations were introduced by Abe et al. [1]. They proved that if a signature passes the equations, then the signature is either Type A or B, so we use the modified equations.

The original verification equations use ciphertexts and the decryption procedure of Waters' dual encryption scheme, so it is probabilistic and has a semi-functional verification algorithm that uses semi-functional ciphertexts [53]. Type A signatures are signatures with $\gamma = 0$ and pass both the normal and semi-functional verification equations. Type B signatures are signatures with $\gamma \neq 0$ and cannot pass the semi-functional verification equations.

As long as the verification equations are normal, both Type A and Type B signatures are valid signatures and there is no essential difference. Thus, we use Type B signatures in the normal signing algorithm.

Even if v, v_1, v_2 are disclosed, the adversary cannot compute v_2^b (and semi-functional ciphertexts of Waters' dual system encryption [53]). Thus, we add (v, v_1, v_2) to the verification key, which does not affect its security since g^α and $g^{\alpha a_1}$ (and v_2^b) are kept secret and are essential secret signing keys. This was observed by Abe et al. [1].

For the slightly changed version above, the following theorem holds.

Theorem 4 ([53]) *If the DLIN assumption holds, then* $\mathsf{WdSig} := \mathsf{Wd.}\{\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy}\}$ *is existentially unforgeable against adaptive chosen message attacks.*

Waters proposed the signature scheme at CRYPTO'09, but he presented only an outline of the proof and did not write a full proof. We present a full proof for confirmation since we slightly modified the scheme as explained above.

Proof There are two types of signature in Waters' dual signature scheme [53].

Type A: $\gamma = 0$ for $(\sigma_0, \sigma_1, \ldots, \sigma_7, \mathsf{stag}) \xleftarrow{R} \mathsf{Wd.Sign}(SK, M)$.
Type B: $\gamma \neq 0$. We can generate a Type B signature if we have the secret key and $g^{a_1 a_2}$.

Both types of signatures pass the verification algorithm (we can check this by simple calculation). Type B signatures correspond to semi-functional private keys of the dual system encryption [53]. We can use the following lemma proved by Abe et al. in this proof.

Lemma 1 ([1]) *Any signature that is accepted by the verification algorithm must be formed either as a Type A or Type B signature.*

To show that Waters' dual signature scheme satisfies unforgeability, we introduce the following games.

1. We denote a game where the signing oracle answers Type A signatures for all signing queries by Game-0. If adversary $\mathcal{A}$ outputs a forgery of a Type B signature, then we can construct algorithm $\mathcal{B}_1$ (simulator for $\mathcal{A}$), which solves the DLIN problem (Lemma 2). In this case, $\mathcal{B}_1$ generates a Type A signature for simulation of the signing oracle. Thus, in the following games, we consider adversary $\mathcal{A}$ which outputs a forgery of a Type A signature only.
2. We consider Game-i where the signing oracle returns a Type B signature for the first i signing queries and Type A signature for the remaining $q - i$ queries ($i \in [q]$). If $\mathcal{A}$ detects the change (from Type A to Type B answer), we can construct algorithm $\mathcal{B}$ (simulator for $\mathcal{A}$) that solves the DLIN problem (Lemma 3). Thus, we can return a Type B signature for all signing queries in Game-q.
3. Now, all answers for signing queries of $\mathcal{A}$ are Type B signatures. We show that $\mathcal{A}$ cannot forge a Type A signature in this game (Game-q). If $\mathcal{A}$ outputs a forgery of a Type A signature, then we can construct algorithm $\mathcal{B}_2$ that solves the CDH problem (Lemma 4). Thus, if the DLIN assumption holds, the signature scheme is unforgeable (the CDH assumption is implied by the DLIN assumption).

We can reverse the order of the game transitions, so we can use either type of signatures as a signature generated by a normal signing algorithm. We denote $\mathsf{Adv}_i^{\text{Type-A}}$ as the advantage of the adversary which outputs a Type A forgery in Game-i (similarly $\mathsf{Adv}_i^{\text{Type-B}}$). It holds that

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{\text{euf-cma}}(\lambda) &= \mathsf{Adv}_0^{\text{Type-A}} + \mathsf{Adv}_0^{\text{Type-B}} \\
&\leq \mathsf{Adv}_0^{\text{Type-A}} + \mathsf{Adv}_{\mathcal{B}_1}^{\text{DLIN}} \\
&\leq q\mathsf{Adv}_{\mathcal{B}}^{\text{DLIN}} + \mathsf{Adv}_q^{\text{Type-A}} + \mathsf{Adv}_{\mathcal{B}_1}^{\text{DLIN}} \\
&\leq q\mathsf{Adv}_{\mathcal{B}}^{\text{DLIN}} + \mathsf{Adv}_{\mathcal{B}_2}^{\text{CDH}} + \mathsf{Adv}_{\mathcal{B}_1}^{\text{DLIN}}
\end{aligned}$$

by Lemmas 2, 3, and 4. □

Lemma 2 *If there exists adversary $\mathcal{A}$ that outputs a Type B forgery, then we can construct algorithm $\mathcal{B}_1$ that solves the DLIN problem.*

Proof of lemma Algorithm $\mathcal{B}_1$ is given instance $(\Gamma, g, f = g^{a_1}, v = g^{a_2}, g^x, f^y, T)$ of the DLIN problem and simulates the verification key and the signing oracle for the signature scheme ($\mathcal{B}_1$ does not have value a_1, a_2, x, y).

$\mathcal{B}_1$ generates the verification key as follows. It selects exponents $b, \alpha, y_v, y_{v_1}, y_{v_2} \xleftarrow{\cup} \mathbb{Z}_p$ and generators $u, w, h \xleftarrow{\cup} \mathbb{G}$, computes

$$
\begin{array}{lll}
g := g & g^{a_1} := f & g^{a_2} := v \\
g^b := g^b & g^{ba_1} := f^b & g^{ba_2} := v^b \\
v := g^{y_v} & v_1 := g^{y_{v_1}} & v_2 := g^{y_{v_2}} \\
\tau_1 := v v_1^{a_1} = g^{y_v} f^{y_{v_1}} & \tau_2 := v v_2^{a_2} = g^{y_v} v^{y_{v_2}} & e(g,g)^{\alpha a_1 b} := e(g,f)^{\alpha b},
\end{array}
$$

and sets $VK := (\Gamma, g^b, f, v, f^b, v^b, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, u, h, e(g,f)^{\alpha b})$ and $SK := (VK, g^\alpha, f^\alpha, v, v_1, v_2)$. $\mathcal{B}_1$ can generate Type A signatures by using the (normal) signing algorithm since $\mathcal{B}_1$ has α and g^{a_1}.

If adversary $\mathcal{A}$ outputs a Type B forgery $\sigma_1 := (g^{\alpha a_1} v^r) \cdot g^{-a_1 a_2 \gamma}$, $\sigma_2 := (g^{-\alpha} v_1^r g^{z_1}) \cdot g^{a_2 \gamma}$, $\sigma_3 := (g^b)^{-z_1}$, $\sigma_4 := (v_2^r g^{z_2}) \cdot g^{a_1 \gamma}$, $\sigma_5 := (g^b)^{-z_2}$, $\sigma_6 := (g^b)^{r_2}$, $\sigma_7 := g^{r_1}$, and $\sigma_0 := (u^M w^{\mathsf{stag}} h)^{r_1}$ for some $r_1, r_2, z_1, z_2, \gamma \in \mathbb{Z}_p$ ($r = r_1 + r_2$), then $\mathcal{B}_1$ can compute $(g^{-a_1 a_2 \gamma}, g^{a_1 \gamma}, g^{a_2 \gamma})$ from $\sigma_1, \sigma_4, \sigma_2$, respectively. The reason is as follows.

$\mathcal{B}_1$ has b, so can compute $g^{z_1}, g^{z_2}, g^{r_1}, g^{r_2}$ from $\sigma_3 = g^{-bz_1}$, $\sigma_5 = g^{-bz_2}$, $\sigma_7 = g^{r_1}$, $\sigma_6 = g^{br_2}$, respectively, and obtains $g^r = g^{r_1 + r_2}$, $v^r = g^{r y_v}$, $v_1^r = g^{r y_{v_1}}$, $v_2^r = g^{r y_{v_2}}$ ($\mathcal{B}_1$ has y_v, y_{v_1}, y_{v_2}). Thus, $\mathcal{B}_1$ can extract $(g^{-a_1 a_2 \gamma}, g^{a_1 \gamma}, g^{a_2 \gamma})$ from $\sigma_1, \sigma_2, \sigma_4$ and solve the DLIN problem by checking whether $e(g^x, g^{-a_1 a_2 \gamma})^{-1} \cdot e(f^y, g^{a_2 \gamma}) = e(g^{a_1 \gamma}, T)$ because $e(g^x, g^{-a_1 a_2 \gamma})^{-1} \cdot e(g^{a_1 y}, g^{a_2 \gamma}) = e(g,g)^{a_1 a_2 \gamma x + a_1 a_2 \gamma y} = e(g,g)^{a_1 a_2 \gamma (x+y)}$. If $T = g^{a_2 (x+y)} = v^{x+y}$, then the equation holds. Thus, $\mathcal{B}_1$ can solve the DLIN problem if the adversary outputs a Type B forgery. □

Lemma 3 *If there exists adversary $\mathcal{A}$ that makes at most q queries and it holds that for some $k \in [q]$, $\left| \mathsf{Adv}_{k-1}^{\text{Type-A}} - \mathsf{Adv}_k^{\text{Type-A}} \right| = \varepsilon$, then we can construct algorithm $\mathcal{B}$ that solves the DLIN problem with advantage ε.*

Proof of lemma $\mathcal{B}$ is given instance $(\Gamma, g, f = g^b, v, g^x, f^y, T)$ of the DLIN problem. $\mathcal{B}$ generates the verification key as follows. It selects exponents $\alpha, a_1, a_2, y_{v_1}, y_{v_2}, y_w, y_u, y_h, A, B \xleftarrow{\cup} \mathbb{Z}_p$, computes

$$
\begin{array}{lll}
g := g & g^{a_1} := g^{a_1} & g^{a_2} := g^{a_2} \\
g^b := f & g^{ba_1} := f^{a_1} & g^{ba_2} := f^{a_2} \\
v := v^{-a_1 a_2} & v_1 := v^{a_2} g^{y_{v_1}} & v_2 := v^{a_1} g^{y_{v_2}} \\
\tau_1 := v v_1^{a_1} = g^{y_{v_1} a_1} & \tau_2 := v v_2^{a_2} = g^{y_{v_2} a_2} & \tau_1^b := (v v_1^{a_1})^b = f^{y_{v_1} a_1} \\
\tau_2^b := (v v_2^{a_2})^b = f^{y_{v_2} a_2} & w := f g^{y_w} & u := f^{-A} g^{y_u} \\
h := f^{-B} g^{y_h} &&
\end{array}
$$

and $e(g,g)^{\alpha a_1 b} := e(f,g)^{\alpha a_1}$, and sets $VK := (\Gamma, f, g^{a_1}, g^{a_2}, f^{a_1}, f^{a_2}, v, v_1, v_2, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, u, h, e(f,g)^{\alpha a_1})$ and $SK := (VK, g^\alpha, g^{\alpha a_1})$. $\mathcal{B}$ has $g^{a_1 a_2}$ since it has (a_1, a_2). Thus, $\mathcal{B}$ can generate Type B signatures.

We define $\mathcal{F}(M) := AM + B$. If $\mathsf{vtag} := \mathcal{F}(M)$, then it holds $(u^M w^{\mathsf{vtag}} h) = f^{\mathsf{vtag} - AM - B} \cdot g^{M y_u + \mathsf{vtag} y_w + y_h} = g^{M y_u + \mathsf{vtag} y_w + y_h}$.

$\mathcal{B}$ answers signing queries as follows. For the i-th signing query,

Case $i > k$: Returns Type A signatures by using $SK = (VK, g^{\alpha}, g^{\alpha a_1})$.
Case $i < k$: Returns Type B signatures by using SK and $g^{a_1 a_2}$.
Case $i = k$: Embeds the instance as follows. For the k-th query M, $\mathcal{B}$ first generates a Type A signature $(\sigma'_1, \sigma'_2, \sigma'_3, \sigma'_4, \sigma'_5, \sigma'_6, \sigma'_7, \sigma'_0)$ by using tag $\mathsf{stag} := \mathcal{F}(M)$ (with randomness $r'_1, r'_2, z'_1, z'_2 \xleftarrow{\cup} \mathbb{Z}_p$, $r' := r'_1 + r'_2$). Next, it computes

$$\begin{aligned}
\sigma_1 &:= \sigma'_1 \cdot T^{-a_1 a_2} &&= g^{\alpha a_1} (v^{r-(x+y)} T)^{-a_1 a_2} \\
\sigma_2 &:= \sigma'_2 \cdot T^{a_2} (g^x)^{y_{v_1}} &&= g^{-\alpha} (v^{r-(x+y)} T)^{a_2} g^{r y_{v_1}} g^{z_1} \\
\sigma_3 &:= \sigma'_3 \cdot (f^y)^{y_{v_1}} &&= (g^b)^{-z_1} \\
\sigma_4 &:= \sigma'_4 \cdot T^{a_1} (g^x)^{y_{v_2}} &&= (v^{r-(x+y)} T)^{a_1} g^{r y_{v_2}} g^{z_2} \\
\sigma_5 &:= \sigma'_5 \cdot (f^y)^{y_{v_2}} &&= (g^b)^{-z_2} \\
\sigma_6 &:= \sigma'_6 \cdot f^y &&= g^{r_2 b} \\
\sigma_7 &:= \sigma'_7 \cdot g^x &&= g^{r_1} \\
\sigma_0 &:= \sigma'_K \cdot (g^x)^{M y_u + y_h + \mathsf{stag} y_w} &&= (u^M w^{\mathsf{stag}} h)^{r_1}.
\end{aligned}$$

In this case, it implicitly holds that $z_1 := z'_1 - y_{v_1} y$, $z_2 := z'_2 - y_{v_2} y$, $r_1 := r'_1 + x$, $r_2 := r'_2 + y$. $\mathcal{B}$ can generate σ_K correctly since $\mathcal{B}$ set $\mathsf{stag} := \mathcal{F}(M)$.

- If $T = v^{x+y}$, the above signature is a Type A with randomness $r_1 := r'_1 + x$, $r_2 := r'_2 + y$.
- If $T \xleftarrow{\cup} \mathbb{G}$, the above signature is a Type B since $T = v^{x+y} g^{\gamma}$ for some $\gamma \xleftarrow{\cup} \mathbb{Z}_p$.

At some point, $\mathcal{A}$ outputs a signature $\sigma^* = (\sigma_0^*, \ldots, \sigma_7^*, \mathsf{stag}^*)$ and message M^*. $\mathcal{B}_2$ verifies

$$\begin{aligned}
&e(\sigma_0^*, v) e(v^{\mathsf{stag}^* - AM^* - B}, \sigma_6^*) \\
&\quad = e((\sigma_1^* / g^{\alpha a_1})^{-\frac{1}{a_1 a_2}}, f^{\mathsf{stag}^* - AM^* - B}) e(\sigma_7^*, v^{M^* y_u + \mathsf{stag}^* y_w + y_h}).
\end{aligned}$$

It holds that $\sigma_0^* = (f^{\mathsf{stag}^* - AM^* - B} g^{M^* y_u + \mathsf{stag}^* y_w + y_h})^{r_1^*}$, $\sigma_1^* = g^{\alpha a_1} v^{-a_1 a_2 r^*} g^{-a_1 a_2 \gamma^*}$, $\sigma_6^* = g^{b r_2^*}$, and $\sigma_7^* = g^{r_1^*}$. Thus, the left-hand side is

$$e(f, v)^{r_1^* (\mathsf{stag}^* - AM^* - B)} e(g, v)^{r_1^* (M^* y_u + \mathsf{stag}^* y_w + y_h)} e(v, f)^{r_2^* (\mathsf{stag}^* - AM^* - B)},$$

and the right-hand side is

$$e(v, f)^{r^* (\mathsf{stag}^* - AM^* - B)} e(g, f)^{\gamma^* (\mathsf{stag}^* - AM^* - B)} e(g, v)^{r_1^* (M^* y_u + \mathsf{stag}^* y_w + y_h)}.$$

A simplified equation is

$$1 = e(g, f)^{\gamma^* (\mathsf{stag}^* - AM^* - B)}.$$

It holds that $\mathsf{stag}^* \neq AM^* + B$ without negligible probability because $M^* \neq M_i$ for all $i \in [q]$ and A and B are information theoretically hidden from the adversary.

If signature σ^* is Type A, then the above equation holds and $\mathcal{B}_2$ outputs 1. On the other hand, if σ^* is Type B, then it does not hold and $\mathcal{B}_2$ outputs 0.

Thus, if $\mathcal{A}$ can distinguish the two games, then $\mathcal{B}_2$ can solve the DLIN problem with the same advantage. □

Lemma 4 *If there exists adversary $\mathcal{A}$ that outputs a Type A forgery when all answers to signing queries are Type B signatures, then we can construct algorithm $\mathcal{B}_2$ that solves the CDH problem.*

Proof of lemma $\mathcal{B}_2$ is given instance (Γ, g, g^x, g^y) of the CDH problem. $\mathcal{B}_2$ generates the verification key as follows. It selects exponents $a_1, b, y_v, y_{v_1}, y_{v_2}, y_w, y_h, y_u \xleftarrow{\cup} \mathbb{Z}_p$, computes

$$
\begin{array}{lllllll}
g := g & g^{a_1} := g^{a_1} & g^{a_2} := g^y & g^b := g^b & g^{ba_2} := (g^y)^b & g^{ba_1} := g^{ba_1} \\
v := g^{y_v} & v_1 := g^{y_{v_1}} & v_2 := g^{y_{v_2}} & w := g^{y_w} & u := g^{y_u} & h := g^{y_h} \\
\tau_1 := v v_1^{a_1} & \tau_1^b := \tau_1^b & \tau_2 := v (g^y)^{y_{v_2}} & \tau_2^b = \tau_2^b & &
\end{array}
$$

and $e(g,g)^{\alpha a_1 b} := e(g^x, g^y)^{a_1 \cdot b}$ (it implicitly holds $\alpha = xy$ though $\mathcal{B}_2$ does not have α), and sets $VK := (\Gamma, g^b, g^{a_1}, g^{a_2}, g^{ba_1}, g^{ba_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, u, h, e(g,g)^{\alpha a_1 b})$. Note that $\mathcal{B}_2$ does not have $g^\alpha = g^{xy}$, so $\mathcal{B}_2$ cannot compute a Type A signature. $\mathcal{B}_2$ outputs Type B signatures for signing query M as follows. It selects $r_1, r_2, z_1, z_2, \gamma', \mathsf{stag} \xleftarrow{\cup} \mathbb{Z}_p$, sets $r := r_1 + r_2$ (we want to set $\gamma := x + \gamma'$), computes

$$
\begin{array}{lll}
\sigma_1 & := (g^y)^{-\gamma' a_1} \cdot v^r & = (g^{\alpha a_1} v^r) \cdot g^{-a_1 a_2 \gamma} \ (a_2 = y, xy = \alpha) \\
\sigma_2 & := (g^y)^{\gamma'} v_1^r g^{z_1} & = (g^\alpha v_1^r g^{z_1}) \cdot g^{a_2 \gamma} \\
\sigma_3 & := (g^b)^{-z_1} & \\
\sigma_4 & := (g^x)^{a_1} g^{a_1 \gamma'} v_2^r g^{z_2} & = (v_2^r g^{z_2}) \cdot g^{a_1 \gamma} \\
\sigma_5 & := (g^b)^{-z_2} & \\
\sigma_6 & := g^{r_2 b} & \\
\sigma_7 & := g^{r_1} & \\
\sigma_0 & := (u^M w^{\mathsf{stag}} h)^{r_1}, &
\end{array}
$$

and outputs signature $(\sigma_1, \ldots, \sigma_0, \mathsf{stag})$ for M.

At some point, $\mathcal{A}$ outputs a Type A forgery, $\sigma_1^* = g^{\alpha a_1} v^{r_1^*}$, $\sigma_2^* = g^{-\alpha} v_1^{r_1^*} g^{z_1^*}$, $\sigma_3^* = (g^b)^{-z_1^*}$, $\sigma_4^* = v_2^{r_1^*} g^{z_2^*}$, $\sigma_5^* = (g^b)^{-z_2^*}$, $\sigma_6^* = g^{r_2^* b}$, $\sigma_7^* = g^{r_1^*}$, and $\sigma_0^* = (u^{M^*} w^{\mathsf{stag}^*} h)^{r_1^*}$ for some $r_1^*, r_2^*, z_1^*, z_2^*, \mathsf{stag}^* \in \mathbb{Z}_p$.

By using these values, $\mathcal{B}_2$ can compute $g^{r_2^*} = (\sigma_6^*)^{1/b}$, $g^{r_1^*} = \sigma_7^*$, $g^{z_1^*} = (\sigma_3^*)^{-1/b}$, $v_1^{r^*} = (g^{r_1^*} \cdot g^{r_2^*})^{y_{v_1}}$ since $v_1 = g^{y_{v_1}}$. Thus, $\mathcal{B}_2$ can compute $g^{z_1^*} \cdot v_1^{r^*} / \sigma_2^* = g^\alpha = g^{xy}$. That is, $\mathcal{B}_2$ can solve the CDH problem. □

Original Waters' dual signature is not strongly unforgeable since it is re-randomizable. "Strong" means that the adversary cannot forge a signature even for a queried message to the signing oracle. To make our VES scheme satisfy opacity, we extend the technique by Boneh, Shen, and Waters [11] and modify Waters' dual signature. They introduced a property called 2-partitioned to convert unforgeable signature schemes into *strongly* unforgeable signature schemes. We extend 2-partitioned to 3-partitioned.

Definition 14 A signature scheme is 3-partitioned if it satisfies the following two properties.

- The signing algorithm consists of three deterministic algorithms F_1, F_2, and F_3.
  1. It selects random $R \in \mathcal{R}$ ($\mathcal{R}$ is a space for randomness).
  2. It computes $\Sigma_1 := F_1(M, R, VK)$, $\Sigma_2 := F_2(R, VK)$, $\Sigma_3 := F_3(R, SK)$.
  3. It outputs signature $\sigma := (\Sigma_1, \Sigma_2, \Sigma_3)$.
- Given M and Σ_2, there is at most one (Σ_1, Σ_3) such that $(\Sigma_1, \Sigma_2, \Sigma_3)$ is a valid signature on M under VK.

A 2-partitioned signature is $\sigma = (\Sigma_1', \Sigma_2')$ where $\Sigma_1' = F_1'(M, R, SK)$ and $\Sigma_2' = F_2'(R, SK)$ [11]. Value Σ_2' binds all randomness R, so M and R fully determine Σ_1'. For a VES, signature elements related to the secret signing key (i.e., Σ_3) should be encrypted, so we cannot use such elements as inputs to hash functions (we will use hash functions to obtain strongly secure signature) and want to isolate the secret signing key from Σ_2'. Otherwise, encrypted signatures are not verifiable. If Σ_3 is not used as an input of a hash function, then hash values are not changed even if Σ_3 is encrypted. This is the reason we introduced 3-partitioned and Σ_1 and Σ_2 do not depend on the secret signing key.

Let $\Pi := (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be an existentially unforgeable signature scheme. The new signature scheme $\Pi' := (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ is as follows.

$\mathsf{Gen}'(1^\lambda)$: It generates $(VK, SK) \xleftarrow{R} \mathsf{Gen}(1^\lambda)$, selects $\bar{h} \xleftarrow{U} \mathbb{G}$ and random hash key $k \in \mathcal{K}$, and sets $(VK', SK') := ((VK, \bar{h}, k), SK)$.

$\mathsf{Sign}'(SK', M)$: On input message $M \in \{0,1\}^\ell$, it selects exponent $\varphi \xleftarrow{U} \mathbb{Z}_p$ and randomness $R \in \mathcal{R}$, computes

$$\Sigma_2 := F_2(R, VK) \qquad \vartheta := H_k(M \parallel \Sigma_2) \qquad m := H_k(g^\vartheta \bar{h}^\varphi)$$
$$\Sigma_1 := F_1(m, R, VK)$$
$$\Sigma_3 := F_3(R, SK)$$

(we consider ϑ as an element in $\mathbb{Z}_p$), and outputs $\mathsf{sig} := (\Sigma_1, \Sigma_2, \Sigma_3, \varphi)$ as a signature.

$\mathsf{Vrfy}'(VK', \mathsf{sig}, M)$: On input VK', M, and signature $\mathsf{sig} = (\Sigma_1, \Sigma_2, \Sigma_3, \varphi)$, it computes $\vartheta' := H_k(M \parallel \Sigma_2)$ (view ϑ' as an element in $\mathbb{Z}_p$), $m' := H_k(g^{\vartheta'} \bar{h}^\varphi)$, It outputs 1 if and only if $\mathsf{Vrfy}(VK, (\Sigma_1, \Sigma_2, \Sigma_3), m') \to 1$.

Theorem 5 *Signature scheme Π' is strongly existentially unforgeable if Π is existentially unforgeable, the DL assumption holds in $\mathbb{G}$, and H is collision-resistant. Specifically,*

$$\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathcal{F}'} \leq \frac{1}{3}\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{F}} + \frac{1}{3}\mathsf{Adv}^{\mathsf{dl}}_{\mathcal{B}} + \frac{1}{3}\mathsf{Adv}^{\mathsf{crhf}}_{\mathcal{C}}.$$

This is easily proved by extending the proof of Boneh et al. [11]. The essential point is that given message M and partial signature Σ_2, the randomness that is used to generate the whole signature is determined and there is at most one (Σ_1, Σ_3) such that $(\Sigma_1, \Sigma_2, \Sigma_3)$ is a valid signature on M under VK. Intuitively, in the construction of Π', we sign not only message M but also randomness R to bind the randomness and prevent re-randomization. Moreover, to prevent message m, which will be signed depending on randomness R, new randomness φ is introduced and chameleon hash functions, $g^\vartheta \bar{h}^\varphi$, are used.

Proof We assume that there is adversary $\mathcal{A}$ that breaks strong unforgeability of Π'. $\mathcal{A}$ is given $(VK, g, \bar{h}, k)$, queries $M_1, \ldots, M_q$, and obtains $(\Sigma_{i,1}, \Sigma_{i,2}, \Sigma_{i,3}, \varphi_i)$ for $i = 1, \ldots, q$, $\vartheta_i := H_k(M_i \parallel \Sigma_{i,2})$ and $m_i := H_k(g^{\vartheta_i} \bar{h}^{\varphi_i})$. At some point, $\mathcal{A}$ outputs forgery $(M^*, (\Sigma_1^*, \Sigma_2^*, \Sigma_3^*, \varphi^*))$ such that $\vartheta^* := H_k(M^* \parallel \Sigma_2^*)$ and $m^* := H_k(g^{\vartheta^*} \bar{h}^{\varphi^*})$. The forgery is one of the following three types.

Type 1: For some $i \in [q]$, $m^* = m_i$ and $\vartheta^* = \vartheta_i$.
Type 2: For some $i \in [q]$, $m^* = m_i$ and $\vartheta^* \neq \vartheta_i$.
Type 3: For all $i \in [q]$, $m^* \neq m_i$.

Type 1 breaks the collision-resistance of the hash function, Type 2 breaks the DL assumption, and Type 3 breaks the existential unforgeability of the underlying signature scheme Π. We construct a simulator that breaks the one of them by using $\mathcal{A}$. First, we randomly guess which type of forgery $\mathcal{A}$ outputs.

Type 1 case. If $\mathcal{A}$ is a Type 1 forger, then we can construct algorithm $\mathcal{B}$ that breaks the collision-resistance of $\mathcal{H}$. $\mathcal{B}$ is given random key k, generates $(VK, SK) \xleftarrow{R} \mathsf{Gen}(1^\lambda)$, selects $g, \bar{h} \xleftarrow{U} \mathbb{G}$, and gives $(VK, g, \bar{h}, k)$ to $\mathcal{A}$. If $\mathcal{A}$ queries M_i, then $\mathcal{B}$ generates $\mathsf{sig}_i \xleftarrow{R} \mathsf{Sign}'(SK, M_i)$ and returns sig_i. If $\mathcal{A}$ outputs $(M^*, \mathsf{sig}^* := (\Sigma_1^*, \Sigma_2^*, \Sigma_3^*))$ such that $(M^*, \mathsf{sig}^*) \notin \{(M_1, \mathsf{sig}_1), \ldots, (M_q, \mathsf{sig}_q)\}$, $m^* = m_i$, and $\vartheta^* = \vartheta_i$ for some $i \in [q]$, then $\mathcal{B}$ outputs $(M^* \parallel \Sigma_2^*, M_i \parallel \Sigma_{i,2})$ as a collision on H_k. $\vartheta^* = \vartheta_i$ means that $H_k(M^* \parallel \Sigma_2^*) = H_k(M_i \parallel \Sigma_{i,2})$ and $m^* = m_i$ means that $H_k(g^{\vartheta^*}\bar{h}^{\varphi^*}) = H_k(g^{\vartheta_i}\bar{h}^{\varphi_i})$ If $M^* \parallel \Sigma_2^* \neq M_i \parallel \Sigma_{i,2}$, then the output of $\mathcal{B}$ is a valid collision. Assume that $M^* = M_i$ and $\Sigma_2^* = \Sigma_{i,2}$ for a contradiction. If $\varphi^* \neq \varphi_i$, then $\mathcal{B}$ outputs $(g^{\vartheta^*}\bar{h}^{\varphi^*}, g^{\vartheta_i}\bar{h}^{\varphi_i})$ as a collision on H_k since $\vartheta^* = \vartheta_i$ in this case. Else if it holds that $\varphi^* = \varphi_i$, then $m^* = m_i$ since $\vartheta^* = \vartheta_i$. Due to the second property of 3-partitioned signatures $m^* = m_i$ and $\Sigma_2^* = \Sigma_{i,2}$ implies that $\Sigma_1^* = \Sigma_{i,1}$ and $\Sigma_3^* = \Sigma_{i,3}$. Such $(M^*, (\Sigma_1^*, \Sigma_2^*, \Sigma_3^*))$ is not a valid forgery of Π'. Thus, it should hold that $M^* \parallel \Sigma_2^* \neq M_i \parallel \Sigma_{i,2}$ and $\mathcal{B}$ can output a valid collision.

Type 2 case. If $\mathcal{A}$ is a Type 2 forger, then we can construct algorithm $\mathcal{B}$ that breaks the DL assumption. $\mathcal{B}$ is given $(g, \bar{h})$ (Implicitly it holds $\bar{h} = g^\eta$), generates $(VK, SK) \xleftarrow{R} \mathsf{Gen}(1^\lambda)$, selects random $k \xleftarrow{R} \mathcal{K}$, and gives $(VK, g, \bar{h}, k)$ to $\mathcal{A}$. If $\mathcal{A}$ queries M_i, then $\mathcal{B}$ generates $\mathsf{sig}_i \xleftarrow{R} \mathsf{Sign}(SK, M_i)$ and returns sig_i. If $\mathcal{A}$ outputs $(M^*, \mathsf{sig}^* = (\Sigma_1^*, \Sigma_2^*, \Sigma_3^*))$ such that $H_k(g^{\vartheta^*}\bar{h}^{\varphi^*}) = H_k(g^{\vartheta_i}\bar{h}^{\varphi_i})$ and $\vartheta^* \neq \vartheta_i$ for some $i \in [q]$, then it holds that $H_k(g^{\vartheta^*}(g^\eta)^{\varphi^*}) = H_k(g^{\vartheta_i}(g^\eta)^{\varphi_i})$. $\mathcal{B}$ computes $\eta := (\vartheta_i - \vartheta^*)/(\varphi^* - \varphi_i)$ and outputs it as a solution of the DL problem. It holds that $\varphi^* \neq \varphi_i$ since if $\varphi^* = \varphi_i$ (here, we assume that $\vartheta^* \neq \vartheta_i$), then we can output $(g^{\vartheta^*}\bar{h}^{\varphi^*}, g^{\vartheta_i}\bar{h}^{\varphi_i})$ as a collision of H_k. Thus, η is a valid solution.

Type 3 case. If $\mathcal{A}$ is a Type 3 forger, then we can construct algorithm $\mathcal{B}$ that breaks the unforgeability of Π. $\mathcal{B}$ is given VK, selects generator $g \in \mathbb{G}$, exponent $\eta \xleftarrow{U} \mathbb{Z}_p$, random key $k \xleftarrow{R} \mathcal{K}$, and sets $\bar{h} := g^\eta$. It gives $(VK, g, \bar{h}, k)$ to $\mathcal{A}$ as a verification key. If $\mathcal{A}$ queries M_i, then $\mathcal{B}$ selects $w \xleftarrow{U} \mathbb{Z}_p$, sets $m_i := H_k(g^w)$, queries m_i to the signing oracle of Π, and obtains signature $\mathsf{sig}_i = (\Sigma_1, \Sigma_2, \Sigma_3)$. It computes $\vartheta_i := H_k(M_i \parallel \Sigma_2)$, sets $\varphi_i := (w - \vartheta_i)/\eta$, and returns $(\Sigma_1, \Sigma_2, \Sigma_3, \varphi_i)$. Note that it holds that $m = H_k(g^w) = H_k(g^{\eta\varphi_i + \vartheta_i}) = H_k(g^{\vartheta_i}\bar{h}^{\varphi_i})$. If $\mathcal{A}$ outputs a Type 3 forgery $(M^*, (\Sigma_1^*, \Sigma_2^*, \Sigma_3^*, \varphi^*))$, then $\mathcal{B}$ computes $\vartheta^* := H_k(M^* \parallel \Sigma_2^*)$, sets $m^* := H_k(g^{\vartheta^*}\bar{h}^{\varphi^*})$, and outputs $(m^*, (\Sigma_1^*, \Sigma_2^*, \Sigma_3^*))$. Now, $\mathcal{A}$ is a Type 3 forgery, so $m^* \notin \{m_1, \ldots, m_q\}$ and $\mathcal{B}$ succeeded in outputting a new signature for m^*. This breaks the security of Π. □

Theorem 6 *Waters' dual signature is* 3*-partitioned.*

Proof Let $\mathcal{R} := \left\{(r_1, r_2, z_1, z_2, \mathsf{stag}, \gamma) \middle| \, r_1, r_2, z_1, z_2, \mathsf{stag}, \gamma \xleftarrow{U} \mathbb{Z}_p\right\}$, then functions F_1, F_2, and F_3 are defined as follows:

$$
\begin{aligned}
F_1(M, R, VK) := \sigma_0 &= (u^M w^{\mathsf{stag}} h)^{r_1} \\
F_2(R, VK) := (\sigma_3, \ldots, \sigma_7, \mathsf{stag}) &= (g^{-bz_1}, v_2^r g^{z_2} \cdot g^{a_1\gamma}, g^{-bz_2}, g^{br_2}, g^{r_1}, \mathsf{stag}) \\
F_3(R, SK) := (\sigma_1, \sigma_2) &= (g^{\alpha a_1} v^r \cdot g^{-a_1 a_2 \gamma}, g^{-\alpha} v_1^r g^{z_1} \cdot g^{a_2\gamma}),
\end{aligned}
$$

where $R \xleftarrow{U} \mathcal{R}$. Here, γ is selected for Type B signatures. If the signature is Type A, then $\gamma := 0$. We can interpret $\sigma_3, \sigma_5, \sigma_6, \sigma_7$ (these are outputs of F_2) as $g^{-bz_1}, g^{-bz_2}, g^{br_2}, g^{r_1}$,

respectively and it follows $\sigma_0 = (u^M w^{\mathsf{stag}} h)^{r_1}$ from the first verification equation, that is, the output of F_1 is fixed. If we interpret σ_4 as $v_2^r g^{z_2} \cdot g^{a_1\gamma}$, then by the second and third equations, two unknowns σ_1 and σ_2 are fixed to $g^{\alpha a_1} v^r \cdot g^{-a_1 a_2 \gamma}$ and $g^{-\alpha} v_1^r g^{z_1}$, respectively, that is, the output of F_3 is fixed. Therefore, if the output of F_2, $(\sigma_3, \ldots, \sigma_7, \mathsf{stag})$, and M are fixed, then the outputs of F_1 and F_3 are also fixed. □

We can see that even if (σ_1, σ_2) is encrypted by the ElGamal encryption, hash value $\vartheta = H_k(M \parallel (\sigma_3, \ldots, \sigma_7, \mathsf{stag}))$ is not changed, so it can be fitted to VES schemes. Note that we assume that each element $g \in \mathbb{G}$ has a unique encoding. We can obtain a strongly secure scheme sWdSig as follows:

$\mathsf{sWd.Gen}(1^\lambda, \Gamma)$: It generates $(VK', SK') \xleftarrow{\mathsf{R}} \mathsf{Wd.Gen}(1^\lambda, \Gamma)$, selects $\bar{h} \xleftarrow{\mathsf{U}} \mathbb{G}$ and random hash key $k \in \mathcal{K}$, and outputs $(VK, SK) := ((VK', \bar{h}, k), SK')$.

$\mathsf{sWd.Sign}(SK, M)$: On input message $M \in \mathbb{Z}_p$, it selects $r_1, r_2, z_1, z_2, \gamma, \mathsf{stag}, \varphi \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, sets $r := r_1 + r_2$, computes

$$\begin{array}{lll}
\sigma_1 := g^{\alpha a_1} v^r \cdot g^{-a_1 a_2 \gamma} & \sigma_2 := g^{-\alpha} v_1^r g^{z_1} \cdot g^{a_2 \gamma} & \sigma_3 := (g^b)^{-z_1} \\
\sigma_4 := v_2^r g^{z_2} \cdot g^{a_1 \gamma} & \sigma_5 := (g^b)^{-z_2} & \sigma_6 := (g^b)^{r_2} \\
\sigma_7 := g^{r_1} & \Sigma_2 := (\sigma_3, \ldots, \sigma_7, \mathsf{stag}) & \vartheta := H_k(M \parallel \Sigma_2) \\
m := H_k(g^\vartheta \bar{h}^\varphi) & \sigma_0 := (u^m w^{\mathsf{stag}} h)^{r_1} &
\end{array}$$

and outputs $\mathsf{sig} := (\sigma_0, \sigma_1, \ldots, \sigma_7, \mathsf{stag}, \varphi)$.

$\mathsf{sWd.Vrfy}(VK, \mathsf{sig}, M)$: On input VK, M, and signature $\mathsf{sig} = (\sigma_0, \sigma_1, \ldots, \sigma_7, \mathsf{stag}, \varphi)$, it computes $\vartheta' := H_k(M \parallel (\sigma_3, \ldots, \sigma_7, \mathsf{stag}))$, $m' := H_k(g^{\vartheta'} \bar{h}^\varphi)$, and $\mathsf{Wd.Vrfy}(VK', \mathsf{sig}', m') \to b$ where $\mathsf{sig}' := (\sigma_0, \ldots, \sigma_7, \mathsf{stag})$, and outputs b.

Corollary 1 *The scheme above is strongly unforgeable against adaptive chosen message attacks if the DLIN assumption holds. That is,* $\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathcal{F}', \mathsf{sWdSig}}(\lambda) \le 1/3(\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{F}, \mathsf{WdSig}}(\lambda) + \mathsf{Adv}^{\mathsf{crhf}}_{\mathcal{C}, \mathcal{H}}(\lambda) + \mathsf{Adv}^{\mathsf{dl}}_{\mathcal{B}}(\lambda)) \le \{(q+3)/3\}\mathsf{Adv}^{\mathsf{dlin}}_{\mathcal{B}'} + (1/3)\mathsf{Adv}^{\mathsf{crhf}}_{\mathcal{C}, \mathcal{H}}$. *(The DL assumption is implied by the DLIN assumption).*

4 Construction of our VES

In this section, we present our VES scheme, sWdVES, based on the strongly secure version of Waters' dual signature scheme. The proposed scheme is essentially the same as strongly unforgeable Waters' dual signature scheme explained in Sect. 3 except that we encrypt signature elements that include secret keys (g^α, $g^{\alpha a_1}$, $g^{a_1 a_2}$) by using the ElGamal encryption scheme. That is, in our creation algorithm, only σ_1 and σ_2 are encrypted. To verify an encrypted signature, we add extra elements and cancel out group elements that are generated by pairing computation of encrypted signature elements in the verification equations. Our scheme, sWdVES, is as follows.

$\mathsf{AdjGen}(1^\lambda)$: It selects $\beta \xleftarrow{\mathsf{U}} \mathbb{Z}_p^\times$ and sets $apk := \zeta := g^\beta$ and $ask := \beta$.

$\mathsf{Gen}(1^\lambda)$: It generates key pair (VK', SK') by using $\mathsf{sWd.Gen}(1^\lambda)$, that is,

$$VK' := (g, g^b, g^{a_1}, g^{a_2}, g^{ba_1}, g^{ba_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, v, v_1, v_2, w, u, h, \bar{h}, k, e(g,g)^{\alpha a_1 b})$$
$$SK' := (g^\alpha, g^{\alpha a_1}, g^{a_1 a_2})$$

and outputs $vk := VK'$ and $sk := (VK', SK')$.

Sign and Vrfy: These are the same as sWd.Sign and sWd.Vrfy explained in Sect. 3, respectively.

Create(sk, apk, M): It generates $(\sigma_0, \sigma_1, \ldots, \sigma_7, \mathsf{stag}, \varphi) \xleftarrow{R} \mathsf{sWd.Sign}(SK', M)$, selects $\rho_1, \rho_2 \xleftarrow{U} \mathbb{Z}_p$, sets

$$K_1 := \sigma_1 \cdot \zeta^{\rho_1} \quad K_1' := g^{\rho_1} \quad \hat{K}_1 := (g^b)^{\rho_1}$$
$$K_2 := \sigma_2 \cdot \zeta^{\rho_2} \quad K_2' := g^{\rho_2} \quad \hat{K}_2 := (g^{ba_1})^{\rho_2}$$
$$K_3 := \sigma_3 \quad K_4 := \sigma_4 \quad K_5 := \sigma_5 \quad K_6 := \sigma_6 \quad K_7 := \sigma_7 \quad K_0 := \sigma_0,$$

and outputs $\omega := (K_0, \ldots, K_7, K_1', K_2', \hat{K}_1, \hat{K}_2, \mathsf{stag}, \varphi)$.

VesVrfy(apk, vk, ω, M): It parses $\omega = (K_0, \ldots, K_7, K_1', K_2', \hat{K}_1, \hat{K}_2, \mathsf{stag}, \varphi)$ and computes $\vartheta' := H_k(M \parallel (K_3, \ldots, K_7, \mathsf{stag}))$ and $m' := H_k(g^{\vartheta'} \bar{h}^{\varphi})$. It outputs 1 if and only if it holds that

$$e(u^{m'} w^{\mathsf{stag}} h, K_7) = e(g, K_0)$$
$$\frac{e(g^b, K_1)}{e(\zeta, \hat{K}_1)} \cdot \frac{e(g^{ba_1}, K_2)}{e(\zeta, \hat{K}_2)} \cdot e(g^{a_1}, K_3) = e(\tau_1, K_6)\, e(\tau_1^b, K_7)$$
$$\frac{e(g^b, K_1)}{e(\zeta, \hat{K}_1)} \cdot e(g^{ba_2}, K_4)\, e(g^{a_2}, K_5) = e(\tau_2, K_6)\, e(\tau_2^b, K_7)\, e(g, g)^{\alpha a_1 b}$$
$$e(K_1', g^b) = e(g, \hat{K}_1)$$
$$e(K_2', g^{ba_1}) = e(g, \hat{K}_2).$$

Adj$(ask, apk, vk, \omega, M)$: It parses $\omega = (K_0, \ldots, K_7, \mathsf{stag}, \varphi)$ and computes $\sigma_1 := K_1 \cdot (K_1')^{-\beta}$, $\sigma_2 := K_2 \cdot (K_2')^{-\beta}$, $\sigma_3 := K_3$, $\sigma_4 := K_4$, $\sigma_5 := K_5$, $\sigma_6 := K_6$, $\sigma_7 := K_7$, $\sigma_0 := K_0$. If VesVrfy$(apk, vk, \omega, M) \to 1$, then it outputs $(\sigma_0, \ldots, \sigma_7, \mathsf{stag}, \varphi)$.

Intuitively, the scheme above is secure because the underlying signature scheme is strongly unforgeable. The adversary has no choice but to decrypt a valid VES given by oracles to output a valid signature, but it contradicts the one-wayness of the ElGamal encryption scheme.

As a corollary of Theorem 2, sWdVES is unforgeable under the DLIN assumption since we can easily show that sWdVES based on sWdSig is key-independent and extractable. See Sect. 2.4 for the definitions and Theorem 2.

Key-independence. For sWdVES, we can set $kisk = \emptyset$, $ssk = (g^{\alpha}, g^{\alpha a_1}, g^{a_1 a_2})$, $kivk = (g^b, g^{ba_1}, g^{ba_2})$, and $svk = VK'$. Thus, sWdVES is key-independent.

Extractability.

Proof From VesVrfy, we have

$$e(u^{m'} w^{\mathsf{stag}} h, K_7) = e(g, K_0) \tag{V1}$$

$$\frac{e(g^b, K_1)}{e(\zeta, \hat{K}_1)} \cdot \frac{e(g^{ba_1}, K_2)}{e(\zeta, \hat{K}_2)} \cdot e(g^{a_1}, K_3) = e(\tau_1, K_6) e(\tau_1^b, K_7) \tag{V2}$$

$$\frac{e(g^b, K_1)}{e(\zeta, \hat{K}_1)} \cdot e(g^{ba_2}, K_4) \cdot e(g^{a_2}, K_5) = e(\tau_2, K_6) e(\tau_2^b, K_7) e(g, g)^{\alpha a_1 b} \tag{V3}$$

$$e(K'_1, g^b) = e(g, \hat{K}_1) \tag{V4}$$

$$e(K'_2, g^{ba_1}) = e(g, \hat{K}_2). \tag{V5}$$

For $j = 0, 3, 4, 5, 6, 7$, it holds that $K_j = \sigma_j$. For an extracted signature from ω by Adj, it holds that (in the following calculation, we use the number of equations above.)

$$e(u^{m'} w^{\mathsf{stag}} h, \sigma_7) = e(g, \sigma_0) \qquad \text{(V1)},$$

and

$$\begin{aligned}
&e(g^b, K_1 \cdot (K'_1)^{-\beta}) \cdot e(g^{ba_1}, K_2 \cdot (K'_2)^{-\beta}) \cdot e(g^{a_1}, K_3) \\
&\quad = e(g^b, K_1) e(g^{ba_1}, K_2) e(g^{a_1}, K_3) \cdot e(g^b, (K'_1)^{-\beta}) \cdot e(g^{ba_1}, (K'_2)^{-\beta}) \\
&\quad = e(\tau_1, \sigma_6) e(\tau_1^b, \sigma_7) \cdot e(\zeta, \hat{K}_1) \cdot e(\zeta, \hat{K}_2) \cdot e(g^b, K'_1)^{-\beta} e(g^{ba_1}, K'_2)^{-\beta} && \text{(V2)} \\
&\quad = e(\tau_1, \sigma_6) e(\tau_1^b, \sigma_7) \cdot e(g^b, K'_1)^{\beta} \cdot e(g^{ba_1}, K'_2)^{\beta} \cdot e(g^b, K'_1)^{-\beta} e(g^{ba_1}, K'_2)^{-\beta} && \text{(V4, V5)} \\
&\quad = e(\tau_1, \sigma_6) e(\tau_1^b, \sigma_7),
\end{aligned}$$

and

$$\begin{aligned}
&e(g^b, K_1 \cdot (K'_1)^{-\beta}) \cdot e(g^{ba_2}, K_4) \cdot e(g^{a_2}, K_5) \\
&\quad = e(g^b, K_1) e(g^{ba_2}, K_4) e(g^{a_2}, K_5) \cdot e(g^b, (K'_1)^{-\beta}) \\
&\quad = e(\tau_2, \sigma_6) e(\tau_2^b, \sigma_7) \cdot e(g, g)^{\alpha a_1 b} \cdot e(\zeta, \hat{K}_1) \cdot e(g^b, K'_1)^{-\beta} && \text{(V3)} \\
&\quad = e(\tau_2, \sigma_6) e(\tau_2^b, \sigma_7) \cdot e(g, g)^{\alpha a_1 b} \cdot e(g^b, K'_1)^{\beta} \cdot e(g^b, K'_1)^{-\beta} && \text{(V4)} \\
&\quad = e(\tau_2, \sigma_6) e(\tau_2^b, \sigma_7) e(g, g)^{\alpha a_1 b}.
\end{aligned}$$

This means that the extracted signature passes the verification algorithm of the underlying signature scheme, $\mathsf{Wd.Vrfy}$. □

As a corollary, sWdVES is unforgeable since sWdVES is key-independent and extractable due to Theorem 2.

Corollary 2 sWdVES *is unforgeable.*

By Theorem 3, sWdVES is collusion-resistant under the DLIN assumption.

Next, we prove the opacity of our scheme.

Theorem 7 sWdVES *is opaque if the DLIN assumption holds and there exists CRHF.*

Proof If $\mathcal{A}$ outputs forgery $\sigma^* = (\sigma_0^*, \ldots, \sigma_7^*, \mathsf{stag}^*, \varphi^*)$ and M^* such that M^* is not queried to $\mathcal{AO}$, then it means that $\mathcal{A}$ breaks the opacity of sWdVES. $\mathcal{A}$ directly forges a signature of the underlying sWdSig or extracts a signature by breaking the one-wayness of the ElGamal encryption scheme. To show that sWdVES satisfies opacity, we introduce the following games. Let Game-(i) denote a game where the VES creation oracle answers encrypted signatures of Type A signatures for the first i ($i \in [q_{\mathsf{C}}]$ and q_{C} is the number of creation queries by $\mathcal{A}$) and queries and encrypted signatures of Type B signatures for the remaining $q_{\mathsf{C}} - i$ queries, and the adjudication oracle answers signatures extracted from the queried VES for all q_{A} (the number of adjudication queries) queries. Let $\mathsf{Adv}_i^{\mathsf{forge\text{-}A}}$ (resp. $\mathsf{Adv}_i^{\mathsf{forge\text{-}B}}$) denote the advantage of the adversary in Game-(i) for outputting a Type A (resp. B) forgery for a non-queried message (a message that is not queried to $\mathcal{CO}$). Let $\mathsf{Adv}_0^{\mathsf{extract\text{-}B}}$ denote the advantage of the adversary in Game-0 for extracting a signature from a VES for a queried message (a message that is queried to $\mathcal{CO}$).

1. In Game-(0), the VES creation oracle returns encrypted signatures of Type B signatures and the adjudication oracle returns Type B signatures. First, we show Lemma 5: If $\mathcal{A}$ outputs a valid Type B signature for message M_i that has been *already queried to the VES creation oracle*, $\mathcal{CO}$ in Game-(0), then we can construct algorithm $\mathcal{E}$ that solves the AgExt problem. Thus, in the remaining games, we only consider $\mathcal{A}$ which outputs a forgery for message M^* such that $M^* \neq M_i$ for all $i \in [q]$. We can show Lemma 6: If $\mathcal{A}$ outputs a forgery of a Type A signature in Game-(0), then we can construct algorithm $\mathcal{B}_1$ (simulator for $\mathcal{A}$), which solves the CDH problem.
2. Next, we consider Game-(i). We can show Lemma 7: If $\mathcal{A}$ detects the change of answers by the VES creation oracle (from Type B answer to Type A answer), we can construct algorithm $\mathcal{B}_2$ (simulator for $\mathcal{A}$) that solves the DLIN problem.
3. Finally, we consider Game-(q_{C}), where all answers for VES queries of $\mathcal{A}$ are encrypted signatures of Type A signatures. We can show Lemma 8: If $\mathcal{A}$ outputs a forgery of a Type B signature in Game-(q_{C}), then we can construct algorithm $\mathcal{B}_3$ that solves the DLIN problem.

Thus, if the DLIN assumption holds, the signature scheme is opaque since the AgExt assumption is equivalent to the CDH assumption and the CDH assumption is implied by the DLIN assumption. The core part is Lemma 5. By Lemmas 5, 6, 7, and 8, we can show

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{Opac}}_{\mathcal{A}}(\lambda) &= \mathsf{Adv}^{\mathsf{forge\text{-}A}}_{0} + \mathsf{Adv}^{\mathsf{extract\text{-}B}}_{0} + \mathsf{Adv}^{\mathsf{forge\text{-}B}}_{0} \\
&\leq \mathsf{Adv}^{\mathsf{extract\text{-}B}}_{0} + \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{F},\mathsf{WdSig}} \\
&\leq q_{\mathsf{C}}\mathsf{Adv}^{\mathsf{AgExt}}_{\mathcal{E}} + \mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathcal{F}',\mathsf{sWdSig}} + \mathsf{Adv}^{\mathsf{crhf}}_{\mathcal{C}} + \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{F},\mathsf{WdSig}} \\
&\leq ((7/3)q_{\mathsf{C}} + 3)\mathsf{Adv}^{\mathsf{dlin}}_{\mathcal{B}} + (4/3)\mathsf{Adv}^{\mathsf{crhf}}_{\mathcal{C}}.
\end{aligned}$$

Lemma 5 *If there exists adversary $\mathcal{A}$ that outputs a Type B forgery for a queried message M_i in* Game-0*, then we can construct algorithm $\mathcal{E}$ that solves the AgExt problem.*

Proof of lemma $\mathcal{E}$ is given instance $(\Gamma, g^x, g^y, g^\beta, g^\delta, g^{xy+\beta\delta})$ of the AgExt problem and generates the verification key as follows. It selects exponents $a_1, b, y_v, y_{v_1}, y_{v_2}, y_w, y_h, y_u, \eta \xleftarrow{\cup} \mathbb{Z}_p$, and hash key $k \in \mathcal{K}$, computes

$$\begin{array}{llllll}
g := g & g^{a_1} := g^{a_1} & g^{a_2} := g^{y} & g^{b} := g^{b} & g^{ba_1} := g^{ba_1} & g^{ba_2} := (g^{y})^{b} \\
v := g^{y_v} & v_1 := g^{y_{v_1}} & v_2 := g^{y_{v_2}} & w := g^{y_w} & u := g^{y_u} & h := g^{y_h} \\
\tau_1 := v v_1^{a_1} & \tau_1^b := \tau_1^b & \tau_2 := v(g^{y})^{y_{v_2}} & \tau_2^b := \tau_2^b & \bar{h} := g^{\eta} & \zeta := g^{\beta}
\end{array}$$

and $e(g,g)^{\alpha a_1 b} := e(g^x, g^y)^{a_1 \cdot b}$ (it implicitly holds $\alpha = xy$ though $\mathcal{E}$ does not have α), and sets $VK := (g, g^b, g^{a_1}, g^{a_2}, g^{ba_1}, g^{ba_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, u, h, \bar{h}, k, e(g,g)^{\alpha a_1 b})$ and $apk := \zeta = g^\beta$. Note that $\mathcal{E}$ does not have $g^\alpha = g^{xy}$, so $\mathcal{E}$ cannot directly compute a Type B signature.

Simulation of creation oracle: $\mathcal{E}$ initializes list $\mathsf{QList} := \emptyset$, selects random index $j \xleftarrow{\cup} [q_{\mathsf{C}}]$, i.e., guesses which VES $\mathcal{A}$ selects and outputs its extraction, and outputs encrypted signatures of Type B signatures for the i-th VES creation query M_i as follows. If $i \neq j$, $\mathcal{E}$ then selects $r_1, r_2, z_1, z_2, \gamma', \mathsf{stag}, \varphi_i, \rho_1, \rho_2 \xleftarrow{\cup} \mathbb{Z}_p$, sets $r := r_1 + r_2$ (we want to set $\gamma := x + \gamma'$), computes

$$\sigma_{i,1} := (g^y)^{-\gamma' a_1} \cdot v^r = (g^{\alpha a_1} v^r) \cdot g^{-a_1 a_2 \gamma}$$
(where $a_2 = y$ and $xy = \alpha$)
$$\sigma_{i,2} := (g^y)^{\gamma'} v_1^r g^{z_1} = (g^\alpha v_1^r g^{z_1}) \cdot g^{a_2 \gamma}$$
$$K_1 := \sigma_{i,1} \cdot \zeta^{\rho_1},\ K_1' := g^{\rho_1},\ \hat{K}_1 := (g^b)^{\rho_1}$$
$$K_2 := \sigma_{i,2} \cdot \zeta^{\rho_2},\ K_2' := g^{\rho_2},\ \hat{K}_2 := (g^{ba_1})^{\rho_2}$$
$$K_3 := \sigma_{i,3} := (g^b)^{-z_1}$$
$$K_4 := \sigma_{i,4} := (g^x)^{a_1} g^{a_1 \gamma'} v_2^r g^{z_2} = (v_2^r g^{z_2}) \cdot g^{a_1 \gamma}$$
$$K_5 := \sigma_{i,5} := (g^b)^{-z_2},\ K_6 := \sigma_{i,6} := g^{r_2 b},\ K_7 := \sigma_{i,7} := g^{r_1}$$
$$\vartheta_i := H_k(M_i \parallel \Sigma_{i,2}) \text{ where } \Sigma_{i,2} := (\sigma_{i,3}, \ldots, \sigma_{i,7})$$
$$m_i := H_k(g^{\vartheta_i} \bar{h}^{\varphi_i}),\ K_0 := \sigma_{i,0} := (u^{m_i} w^{\mathsf{stag}} h)^{r_1},$$

stores $(M_i, \sigma_i, R_i := (r_1, r_2, z_1, z_2, \mathsf{stag}, \gamma_i := \gamma'))$ in QList where $\sigma_i := (\sigma_{i,0}, \ldots, \sigma_{i,7}, \mathsf{stag}, \varphi_i)$, and outputs $\omega := (K_0, \ldots, K_7, K_1', K_2', \hat{K}_1, \hat{K}_2, \mathsf{stag}, \varphi_i)$ for M_i. We can verify σ_i is a correct Type B signature.

Embedding the instance: If $i = j$, then $\mathcal{E}$ selects $r_1^*, r_2^*, z_1^*, z_2^*, \gamma^*, \mathsf{stag}^*, \varphi^*, \rho_1, \rho_2 \xleftarrow{\cup} \mathbb{Z}_p$, sets $r^* = r_1^* + r_2^*$, answers

$$K_1^* := (g^{xy+\beta\delta})^{a_1} v^{r^*} (g^y)^{-a_1 \gamma^*} \zeta^{\rho_1} = (g^{\alpha a_1} v^{r^*}) g^{-a_1 a_2 \gamma^*} \zeta^{\rho_1'}$$
(where $a_2 = y$, $xy = \alpha$, $\rho_1' := a_1 \delta + \rho_1$)
$$K_1^{*\prime} := (g^\delta)^{a_1} g^{\rho_1} = g^{\rho_1'}$$
$$\hat{K}_1^* := (g^\delta)^{ba_1} g^{b\rho_1} = (g^b)^{\rho_1'}$$
$$K_2^* := (g^{xy+\beta\delta})^{-1} v_1^{r^*} g^{z_1^*} \cdot (g^{a_2})^{\gamma^*} (g^\beta)^{\rho_2} = (g^{-\alpha} v_1^{r^*} g^{z_1^*}) \cdot g^{a_2 \gamma^*} \zeta^{\rho_2'}$$
(where $\rho_2' := -\delta + \rho_2$)
$$K_2^{*\prime} := (g^\delta)^{-1} g^{\rho_2} = g^{\rho_2'}$$
$$\hat{K}_2^* := (g^\delta)^{-ba_1} g^{ba_1 \rho_2} = (g^{ba_1})^{\rho_2'}$$
$$K_3^* := (g^b)^{-z_1^*},\ K_4^* := v_2^{r^*} g^{z_2^*} g^{a_1 \gamma^*},\ K_5^* := (g^b)^{-z_2^*},\ K_6^* := g^{r_2^* b},\ K_7^* := g^{r_1^*}$$
$$\vartheta^* := H_k(K_3^*, \ldots, K_7^*, \mathsf{stag}^*),\ m^* := H_k(g^{\vartheta^*} \bar{h}^{\varphi^*}),\ K_0^* := (u^{m^*} w^{\mathsf{stag}^*} h)^{r_1^*},$$

and records $(M_j, \omega^* := (K_0^*, \ldots, \varphi^*), j, \gamma^*)$ as the challenge instance. It can be verified that ω^* is a correct encrypted signature of a Type B signature.

Simulation of adjudication oracle: When $\mathcal{A}$ makes the ℓ-th adjudication query (M_ℓ, ω_ℓ), we know that $\mathcal{A}$ must have queried M_ℓ to $\mathcal{CO}$ by the theorem of Rückert and Schröder (Otherwise, it is a forgery. This is the same argument by Rückert and Schröder in [48]). First, $\mathcal{E}$ verifies the query and returns $\perp$ if it is invalid. Otherwise, $\mathcal{E}$ acts as follows. If $M_\ell = M_j$, that is, the guessed index $((M_j, \ldots) \notin \mathsf{QList})$, then $\mathcal{E}$ aborts. Otherwise, there exists $(M_i, \sigma_i, R_i) \in \mathsf{QList}$ for some $i \neq j$ such that $M_\ell = M_i$ and the signature is Type B. In this case $(M_\ell = M_i)$, for query $(M_\ell, \omega = (K_0, \ldots, K_7, K_1', K_2', \hat{K}_1, \hat{K}_2, \mathsf{stag}, \varphi))$, if $\varphi \neq \varphi_i$, then $\mathcal{A}$ breaks the strong unforgeability of our modified Waters' dual signature. We consider an intermediate game where if $\varphi \neq \varphi_i$, then $\mathcal{E}$ aborts. The probability of $\mathcal{E}$ aborting under this condition is at most the success probability of breaking the strong unforgeability

of sWdSig. That is, it holds that $\varphi = \varphi_i$ without negligible probability. If $\varphi = \varphi_i$, then it holds that $K_3 = \sigma_{i,3}$, $K_4 = \sigma_{i,4}$, $K_5 = \sigma_{i,5}$, $K_6 = \sigma_{i,6}$, $K_7 = \sigma_{i,7}$, and $\mathsf{stag} = \mathsf{stag}_i$ since otherwise it means $\mathcal{A}$ outputs $(K_3, \ldots, K_7, \mathsf{stag})$ such that $H_k(\sigma_{i,3}, \ldots, \sigma_{i,7}, \mathsf{stag}_i) = \varphi_i = \varphi = H_k(K_3, \ldots, K_7, \mathsf{stag})$ and $(K_3, \ldots, K_7, \mathsf{stag}) \neq (\sigma_{i,3}, \ldots, \sigma_{i,7}, \mathsf{stag}_i)$. This is a collision of the hash function and contradicts the collision-resistant property. We consider an intermediate game where if $H_k(\sigma_{i,3}, \ldots, \sigma_{i,7}, \mathsf{stag}_i) = \varphi_i = \varphi = H_k(K_3, \ldots, K_7, \mathsf{stag})$ and $(K_3, \ldots, K_7, \mathsf{stag}) \neq (\sigma_{i,3}, \ldots, \sigma_{i,7}, \mathsf{stag}_i)$, then $\mathcal{E}$ aborts. The probability of $\mathcal{E}$ aborting under this condition is less than the success probability of breaking the CRHF. That is, the randomness of $(K_3, \ldots, K_7, \mathsf{stag})$ is the same as that of $(\sigma_{i,3}, \ldots, \sigma_{i,7}, \mathsf{stag}_i)$ without negligible probability.

By using $K_3 = g^{-bz_1}$, $K_5 = g^{-bz_2}$, $K_6 = g^{br_2}$, and $K_7 = g^{r_1}$, $\mathcal{E}$ can compute $g^{r_2} = (K_6)^{1/b}$, $g^{r_1} = K_7$, $g^{z_1} = (K_3)^{-1/b}$, $g^{z_2} = (K_5)^{-1/b}$, $v^r = (g^{r_1} \cdot g^{r_2})^{y_v}$, $v_1^r = (g^{r_1} \cdot g^{r_2})^{y_{v_1}}$, and $v_2^r = (g^{r_1} \cdot g^{r_2})^{y_{v_2}}$ since $\mathcal{E}$ has b, y_v, y_{v_1}, y_{v_2} and it holds that $v = g^{y_v}$ $v_1 = g^{y_{v_1}}$ $v_2 = g^{y_{v_2}}$. $\mathcal{E}$ can use the same computation procedure in the simulation of the VES creation oracle above by using γ_i stored in QList. Therefore, $\mathcal{E}$ can return a valid Type B signature $(\sigma_0, \ldots, \sigma_7, \mathsf{stag}, \varphi)$ such that the randomness r in σ_1, σ_2 is the same as that in K_1, K_2 by using stored information σ_i. That is, the adjudication oracle is perfectly simulated by $\mathcal{E}$.

Solving the problem: At some point, $\mathcal{A}$ outputs a Type B extraction, $M^* = M_j$, stag^*, $\sigma_1^* = g^{\alpha a_1} v^{r^*} g^{-a_1 a_2 \gamma^*}$, $\sigma_2^* = g^{-\alpha} v_1^{r^*} g^{z_1^*} g^{a_2 \gamma^*}$, $\sigma_3^* = (g^b)^{-z_1^*}$, $\sigma_4^* = v_2^{r^*} g^{z_2^*} g^{a_1 \gamma^*}$, $\sigma_5^* = (g^b)^{-z_2^*}$, $\sigma_6^* = g^{r_2^* b}$, $\sigma_7^* = g^{r_1^*}$, $\vartheta^* = H_k(\sigma_3, \ldots, \sigma_7, \mathsf{stag}^*)$, $m^* = H_k(g^{\vartheta^*} \bar{h}^{\varphi})$, and $\sigma_0^* = (u^{m^*} w^{\mathsf{stag}^*} h)^{r_1^*}$ (not queried to $\mathcal{AO}$ but $\mathcal{CO}$) such that *randomnesses are the same as those used when $\mathcal{B}$ embedded the problem instance at the j-th query* and $(M_j, \omega^*, j, \gamma^*)$ is recorded as the challenge instance. This is guaranteed by the strong unforgeability and collision-resistant property we discussed above. By using these values, $\mathcal{E}$ can compute $g^{r_2^*} = (\sigma_6^*)^{1/b}$, $g^{r_1^*} = \sigma_7^*$, $g^{z_1^*} = (\sigma_3^*)^{-1/b}$, $g^{z_2^*} = (\sigma_5^*)^{-1/b}$, $v^{r^*} = (g^{r_1^*} \cdot g^{r_2^*})^{y_v}$, $v_1^{r^*} = (g^{r_1^*} \cdot g^{r_2^*})^{y_{v_1}}$, $v_2^{r^*} = (g^{r_1^*} \cdot g^{r_2^*})^{y_{v_2}}$ since $\mathcal{E}$ has b, y_v, y_{v_1}, y_{v_2} and it holds that $v = g^{y_v}$ $v_1 = g^{y_{v_1}}$ $v_2 = g^{y_{v_2}}$. Thus, $\mathcal{E}$ can compute $g^{z_1^*} \cdot v_1^{r^*} g^{a_2 \gamma^*} / \sigma_2^* = g^{\alpha} = g^{xy}$ (where $g^{a_2} = g^y$) since $\mathcal{E}$ has a_1 and γ^* is recorded. That is, $\mathcal{E}$ can output solution g^{xy} of the AgExt problem if the adversary outputs a Type A extraction for queried message M_j to $\mathcal{CO}$. $\mathcal{E}$ guesses index j, so its success probability is degraded by a factor of $1/q_{\mathsf{C}}$. However, it still breaks the AgExt problem with non-negligible probability ϵ/q_{C}, where ϵ is the success probability of $\mathcal{A}$. □

Lemma 6 *If there exists adversary $\mathcal{A}$ that outputs a Type A forgery when all answers to signing queries are Type B signatures, then we can construct algorithm $\mathcal{B}_1$ that solves the CDH problem.*

Proof of lemma $\mathcal{B}_1$ is given instance (Γ, g, g^x, g^y) of the CDH problem and generates the verification key as follows: It selects exponents $a_1, b, y_v, y_{v_1}, y_{v_2}, y_w, y_h, y_u, \eta, \zeta \xleftarrow{\cup} \mathbb{Z}_p$, computes

$$
\begin{array}{llllll}
g := g & g^{a_1} := g^{a_1} & g^{a_2} := g^{y} & g^{b} := g^{b} & g^{ba_1} := g^{ba_1} & g^{ba_2} := (g^{y})^{b} \\
v := g^{y_v} & v_1 := g^{y_{v_1}} & v_2 := g^{y_{v_2}} & w := g^{y_w} & u := g^{y_u} & h := g^{y_h} \\
\tau_1 := v v_1^{a_1} & \tau_1^{b} & \tau_2 := v (g^{y})^{y_{v_2}} & \tau_2^{b} & \bar{h} := g^{\eta} & \beta := g^{\zeta}
\end{array}
$$

and $e(g, g)^{\alpha a_1 b} := e(g^x, g^y)^{a_1 \cdot b}$ (implicitly $\alpha = xy$ though $\mathcal{B}_1$ does not have α), and sets $VK := (\Gamma, g^b, g^{a_1}, g^{a_2}, g^{ba_1}, g^{ba_2}, v, v_1, v_2, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, u, h, \bar{h}, k, e(g, g)^{\alpha a_1 b})$ and

$apk := g^{\beta}$. Note that $\mathcal{B}_1$ does not have $g^{\alpha} = g^{xy}$, so $\mathcal{B}_1$ cannot directly compute a Type A signature. $\mathcal{B}_1$ outputs encrypted signatures of Type B signatures for creation query M as follows. It selects $r_1, r_2, z_1, z_2, \gamma', \mathsf{stag}, \varphi \xleftarrow{\cup} \mathbb{Z}_p$, sets $r := r_1 + r_2$ (we want to set $\gamma := x + \gamma'$), and computes

$$
\begin{aligned}
\sigma_1 &:= (g^{y})^{-\gamma' a_1} \cdot v^{r} && = (g^{\alpha a_1} v^{r}) \cdot g^{-a_1 a_2 \gamma} \ (a_2 = y, xy = \alpha)\\
\sigma_2 &:= (g^{y})^{\gamma'} v_1^{r} g^{z_1} && = (g^{\alpha} v_1^{r} g^{z_1}) \cdot g^{a_2 \gamma}\\
\sigma_3 &:= (g^{b})^{-z_1}\\
\sigma_4 &:= (g^{x})^{a_1} g^{a_1 \gamma'} v_2^{r} g^{z_2} && = (v_2^{r} g^{z_2}) \cdot g^{a_1 \gamma}\\
\sigma_5 &:= (g^{b})^{-z_2}\\
\sigma_6 &:= g^{r_2 b}\\
\sigma_7 &:= g^{r_1}\\
\vartheta &:= H_k(M \parallel \sigma_3, \ldots, \sigma_7, \mathsf{stag})\\
m &:= H_k(g^{\vartheta} \bar{h}^{\varphi})\\
\sigma_0 &:= (u^{m} w^{\mathsf{stag}} h)^{r_1}.
\end{aligned}
$$

$\mathcal{B}_1$ outputs VES for M by using signature $(\sigma_0, \ldots, \sigma_7, \mathsf{stag}, \varphi)$ for M and algorithm Create with $apk = \beta$. For an adjudication query, $\mathcal{B}_1$ verifies its validity and returns the decrypted value by using β if it is valid.

At some point, $\mathcal{A}$ outputs a Type A forgery $\mathsf{sig}^* = (\sigma_0^*, \ldots, \sigma_7^*, \mathsf{stag}^*, \varphi^*)$. If sig^* is a valid Type A forgery, then it passes the verification equations and it should hold that $\sigma_1^* = g^{\alpha a_1} v^{r_1^*}$, $\sigma_2^* = g^{-\alpha} v_1^{r^*} g^{z_1^*}$, $\sigma_3^* = (g^b)^{-z_1^*}$, $\sigma_4^* = v_2^{r^*} g^{z_2^*}$, $\sigma_5^* = (g^b)^{-z_2^*}$, $\sigma_6^* = g^{r_2^* b}$, $\sigma_7^* = g^{r_1^*}$, $\vartheta^* = H_k(M^* \parallel \sigma_3^*, \ldots \sigma_7^*, \mathsf{stag}^*)$, $m^* = H_k(g^{\vartheta^*} \bar{h}^{\varphi^*})$, and $\sigma_0^* = (u^{m^*} w^{\mathsf{stag}^*} h)^{r_1^*}$, for some $r_1^*, r_2^*, z_1^*, z_2^*, \mathsf{stag}^* \in \mathbb{Z}_p$.

By using these values, $\mathcal{B}_1$ can compute $g^{r_2^*} = (\sigma_6^*)^{1/b}$, $g^{r_1^*} = \sigma_7^*$, $g^{z_1^*} = (\sigma_3^*)^{-1/b}$, $v_1^{r^*} = (g^{r_1^*} \cdot g^{r_2^*})^{y_{v_1}}$ since $v_1 = g^{y_{v_1}}$. Thus, $\mathcal{B}_1$ can compute $g^{z_1^*} \cdot v_1^{r^*} / \sigma_2^* = g^{\alpha} = g^{xy}$. That is, $\mathcal{B}_1$ can solve the CDH problem. □

Lemma 7 *If there exists $\mathcal{A}$ that makes at most q_{C} creation and q_{A} adjudication queries and it holds $\left|\mathsf{Adv}^{\mathsf{forge\text{-}B}}_{k-1} - \mathsf{Adv}^{\mathsf{forge\text{-}B}}_{k}\right| = \varepsilon$ for some $k \in [q_{\mathsf{C}}]$, then we can construct algorithm $\mathcal{B}_2$ that solves the DLIN problem with advantage ε.*

Proof of lemma $\mathcal{B}_2$ is given instance $(\Gamma, g, f = g^b, v, g^x, f^y, T)$ of the DLIN problem and generates the verification key as follows. It selects exponents $\alpha, a_1, a_2, y_{v_1}, y_{v_2}, y_w, y_u, y_h, \eta, A, B, \beta \xleftarrow{\cup} \mathbb{Z}_p$, computes

$$
\begin{aligned}
&g := g && g^{a_1} := g^{a_1} && g^{a_2} := g^{a_2}\\
&g^b := f && g^{ba_1} := f^{a_1} && g^{ba_2} := f^{a_2}\\
&v := v^{-a_1 a_2} && v_1 := v^{a_2} g^{y_{v_1}} && v_2 := v^{a_1} g^{y_{v_2}}\\
&w := f g^{y_w} && u := f^{-A} g^{y_u} && h := f^{-B} g^{y_h}\\
&\tau_1 := v v_1^{a_1} = g^{y_{v_1} a_1} && \tau_2 := v v_2^{a_2} = g^{y_{v_2} a_2} && \tau_1^b := (v v_1^{a_1})^b = f^{y_{v_1} a_1}\\
&\tau_2^b := (v v_2^{a_2})^b = f^{y_{v_2} a_2} && e(g,g)^{\alpha a_1 b} := e(f,g)^{\alpha a_1}\\
&\bar{h} := g^{\eta} && \zeta := g^{\beta}
\end{aligned}
$$

and sets $VK := (\Gamma, f, g^{a_1}, g^{a_2}, f^{a_1}, f^{a_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, v, v_1, v_2, w, u, h, \bar{h}, k, e(f,g)^{\alpha a_1})$, $SK := (VK, g^{\alpha}, g^{\alpha a_1}, g^{a_1 a_2})$, and $apk := \zeta$. $\mathcal{B}_2$ has $g^{a_1 a_2}$ since it has (a_1, a_2), thus, it can generate a Type B signature. We define $\mathcal{F}(M) := AM + B$. If $\mathsf{stag} := \mathcal{F}(M)$, then $(u^M w^{\mathsf{stag}} h) = f^{\mathsf{stag}-AM-B} \cdot g^{My_u + \mathsf{stag} y_w + y_h} = g^{My_u + \mathsf{stag} y_w + y_h}$. $\mathcal{B}_2$ generates signatures as follows. For the i-th creation query,

Case $i > k$: Returns encrypted signatures of Type B signatures by using SK, $g^{a_1 a_2}$ and ζ.
Case $i < k$: Returns encrypted signatures of Type A signatures by using $SK = (g^{\alpha}, g^{\alpha a_1})$ and ζ.
Case $i = k$: Embeds the instance as follows. For the k-th creation query M, $\mathcal{B}_2$ first generates a Type A signature as follows. It selects $m_0 \xleftarrow{\cup} \mathbb{Z}_p$, sets $m := H_k(g^{m_0})$ and $\mathsf{stag} := \mathcal{F}(m)$, and generates a Type A signature $(\sigma_0', \sigma_1', \sigma_2', \sigma_3', \sigma_4', \sigma_5', \sigma_6', \sigma_7')$ for m by using tag stag (with randomness $r_1', r_2', z_1', z_2' \xleftarrow{\cup} \mathbb{Z}_p$, $r' := r_1' + r_2'$). Next, it computes

$$
\begin{aligned}
\sigma_1 &:= \sigma_1' \cdot T^{-a_1 a_2} && = g^{\alpha a_1} (v^{r-(x+y)} T)^{-a_1 a_2} \\
\sigma_2 &:= \sigma_2' \cdot T^{a_2} (g^x)^{y_{v_1}} && = g^{-\alpha} (v^{r-(x+y)} T)^{a_2} g^{r y_{v_1}} g^{z_1} \\
\sigma_3 &:= \sigma_3' \cdot (f^x)^{y_{v_1}} && = (g^b)^{-z_1} \\
\sigma_4 &:= \sigma_4' \cdot T^{a_1} (g^x)^{y_{v_2}} && = (v^{r-(x+y)} T)^{a_1} g^{r y_{v_2}} g^{z_2} \\
\sigma_5 &:= \sigma_5' \cdot (f^x)^{y_{v_2}} && = (g^b)^{-z_2} \\
\sigma_6 &:= \sigma_6' \cdot f^y && = g^{r_2 b} \\
\sigma_7 &:= \sigma_7' \cdot g^x && = g^{r_1} \\
\vartheta &:= H_k(M \parallel \sigma_3, \ldots, \sigma_7, \mathsf{stag}) && \\
\varphi &:= (m_0 - \vartheta)/\eta && m = H_k(g^{\vartheta} \bar{h}^{\varphi}) \\
\sigma_0 &:= \sigma_0' \cdot (g^x)^{m y_u + y_h + \mathsf{stag} y_w} && = (u^m w^{\mathsf{stag}} h)^{r_1}.
\end{aligned}
$$

In this case, it implicitly holds that $z_1 := z_1' - y_{v_1} y$, $z_2 := z_2' - y_{v_2} y$, $r_1 := r_1' + x$, $r_2 := r_2' + y$. $\mathcal{B}_2$ can generate σ_0 correctly since $\mathcal{B}_2$ sets $\mathsf{stag} := \mathcal{F}(m)$.

- If $T = v^{x+y}$, the above signature is Type A with randomness $r_1 := r_1' + x$, $r_2 := r_2' + y$.
- If $T \xleftarrow{\cup} \mathbb{G}$, the above signature is Type B since $T = v^{x+y} g^{\gamma}$ for some $\gamma \xleftarrow{\cup} \mathbb{Z}_p$.

For VES creation query M, $\mathcal{B}_2$ returns an encrypted signature of the above signature for M by using ζ. For adjudication query (M, ω), if it is a valid VES, then $\mathcal{B}_2$ decrypts it by using β and returns extracted signatures for M. That is, if $T = v^{x+y}$ (linear), then $\mathcal{A}$ is in Game-$(k-1)$, otherwise $\mathcal{A}$ is in Game-(k).

At some point, $\mathcal{A}$ outputs a signature $\sigma^* = (\sigma_0^*, \ldots, \sigma_7^*, \mathsf{stag}^*, \varphi^*)$ and message M^*. Note that in this game, M^* is not queried to $\mathcal{CO}$. $\mathcal{B}_2$ computes $\vartheta' := H_k(M^* \parallel (\sigma_3^*, \ldots, \sigma_7^*, \mathsf{stag}^*))$, $m' := H_k(g^{\vartheta'} \bar{h}^{\varphi^*})$ and verifies

$$
\begin{aligned}
& e(\sigma_0^*, v) e(v^{\mathsf{stag}^* - Am' - B}, \sigma_6^*) \\
&\quad = e((\sigma_1^* / g^{\alpha a_1})^{-\frac{1}{a_1 a_2}}, f^{\mathsf{stag}^* - Am' - B}) e(\sigma_7^*, v^{m' y_u + \mathsf{stag}^* y_w + y_h}).
\end{aligned}
$$

It holds that $\sigma_0^* = (f^{\mathsf{stag}^* - Am' - B} g^{m' y_u + \mathsf{stag}^* y_w + y_h})^{r_1^*}$, $\sigma_1^* = g^{\alpha a_1} v^{-a_1 a_2 r^*} g^{-a_1 a_2 \gamma^*}$, $\sigma_6^* = g^{b r_2^*}$, and $\sigma_7^* = g^{r_1^*}$. Thus, the left-hand side is

$$
e(f, v)^{r_1^* (\mathsf{stag}^* - Am' - B)} e(g, v)^{r_1^* (m' y_u + \mathsf{stag}^* y_w + y_h)} e(v, f)^{r_2^* (\mathsf{stag}^* - Am' - B)},
$$

and the right-hand side is

$$e(v, f)^{r^*(\mathsf{stag}^* - Am' - B)} e(g, f)^{\gamma^*(\mathsf{stag}^* - Am' - B)} e(g, v)^{r_1^*(m' y_u + \mathsf{stag}^* y_w + y_h)}.$$

A simplified equation is

$$1 = e(g, f)^{\gamma^*(\mathsf{stag}^* - Am' - B)}.$$

It holds that $\mathsf{stag}^* \neq Am' + B$ without negligible probability because $M^* \neq M_i$ for all $i \in [q_{\mathsf{C}}]$ and A and B are information theoretically hidden from the adversary. If signature σ^* is Type A, then the above equation holds and $\mathcal{B}_2$ outputs 1. On the other hand, if σ^* is Type B, then it does not hold and $\mathcal{B}_2$ outputs 0.

Thus, if $\mathcal{A}$ can distinguish the two games, then $\mathcal{B}_2$ can solve the DLIN problem with the same advantage. □

Lemma 8 *If $\mathcal{A}$ outputs a Type B forgery, then we can construct algorithm $\mathcal{B}_3$ that solves the DLIN problem.*

Proof of lemma Algorithm $\mathcal{B}_3$ is given instance $(\Gamma, g, f = g^{a_1}, v = g^{a_2}, g^x, f^y, T)$ of the DLIN problem and simulates the verification key and the signing oracle for the signature scheme ($\mathcal{B}_3$ does not have values a_1, a_2, x, y).

$\mathcal{B}_3$ generates the verification key as follows. It selects exponents $b, \alpha, y_v, y_{v_1}, y_{v_2}, \beta, \eta \xleftarrow{\cup} \mathbb{Z}_p$ and generators $u, w, h \xleftarrow{\cup} \mathbb{G}$, computes

$$\begin{array}{lll}
g := g & g^{a_1} := f & g^{a_2} := v \\
g^b := g^b & g^{ba_1} := f^b & g^{ba_2} := v^b \\
v := g^{y_v} & v_1 := g^{y_{v_1}} & v_2 := g^{y_{v_2}} \\
\tau_1 := v v_1^{a_1} = g^{y_v} f^{y_{v_1}} & \tau_2 := v v_2^{a_2} = g^{y_v} v^{y_{v_2}} & e(g, g)^{\alpha a_1 b} := e(g, f)^{\alpha b} \\
\bar{h} := g^{\eta} & \zeta := g^{\beta} &
\end{array}$$

and sets $VK := (\Gamma, g^b, f, v, f^b, v^b, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, u, h, \bar{h}, k, e(g, f)^{\alpha b})$, $SK := (VK, g^{\alpha}, f^{\alpha}, v, v_1, v_2)$, and $apk := \zeta$. $\mathcal{B}_3$ can generate a Type A signature by using the (normal) signing algorithm since it has α and g^{a_1}. Thus, $\mathcal{B}_3$ can return valid encrypted signatures under $apk = \zeta$ for queries to $\mathcal{CO}$. It also has β, so it can decrypt all valid VESs and perfectly simulate all queries to $\mathcal{AO}$.

If adversary $\mathcal{A}$ outputs a Type B forgery $\sigma_1 := (g^{\alpha a_1} v^r) \cdot g^{-a_1 a_2 \gamma}, \sigma_2 := (g^{-\alpha} v_1^r g^{z_1}) \cdot g^{a_2 \gamma}$, $\sigma_3 := (g^b)^{-z_1}, \sigma_4 := (v_2^r g^{z_2}) \cdot g^{a_1 \gamma}, \sigma_5 := (g^b)^{-z_2}, \sigma_6 := (g^b)^{r_2}, \sigma_7 := g^{r_1}, \vartheta := H_k(M \parallel \sigma_3, \ldots, \sigma_7)$, $m := H_k(g^{\vartheta} \bar{h}^{\varphi})$, φ, and $\sigma_0 := (u^m w^{\mathsf{stag}} h)^{r_1}$ for some $r_1, r_2, z_1, z_2, \gamma \in \mathbb{Z}_p$ ($r = r_1 + r_2$), then $\mathcal{B}_3$ can compute $(g^{-a_1 a_2 \gamma}, g^{a_1 \gamma}, g^{a_2 \gamma})$ from $\sigma_1, \sigma_4, \sigma_2$, respectively. The reason is as follows.

$\mathcal{B}_3$ has b, so it can compute $g^{z_1}, g^{z_2}, g^{r_1}, g^{r_2}$ from $\sigma_3 = g^{-bz_1}, \sigma_5 = g^{-bz_2}, \sigma_7 = g^{r_1}$, $\sigma_6 = g^{br_2}$, respectively, and obtain $g^r = g^{r_1 + r_2}$, $v^r = g^{r y_v}$, $v_1^r = g^{r y_{v_1}}$, $v_2^r = g^{r y_{v_2}}$ ($\mathcal{B}_3$ has y_v, y_{v_1}, y_{v_2}). Thus, $\mathcal{B}_3$ can extract $(g^{-a_1 a_2 \gamma}, g^{a_1 \gamma}, g^{a_2 \gamma})$ from $\sigma_1, \sigma_2, \sigma_4$ and can solve the DLIN problem by checking whether $e(g^x, g^{-a_1 a_2 \gamma})^{-1} \cdot e(f^y, g^{a_2 \gamma}) = e(g^{a_1 \gamma}, T)$ because $e(g^x, g^{-a_1 a_2 \gamma})^{-1} \cdot e(g^{a_1 y}, g^{a_2 \gamma}) = e(g, g)^{a_1 a_2 \gamma x + a_1 a_2 \gamma y} = e(g, g)^{a_1 a_2 \gamma (x+y)}$. If $T = g^{a_2 (x+y)} = v^{x+y}$, then the equation holds. Thus, $\mathcal{B}_3$ can solve the DLIN problem if $\mathcal{A}$ outputs a Type B forgery. Note that in this game, $\mathcal{A}$ outputs $M^* \neq M_i$ for all $i \in [q_{\mathsf{C}}]$. □

We complete the proof of Theorem 7 by using Lemmas 5, 6, 7, and 8. □

5 Application to obfuscators for ES and EVES

sWdVES can be used to construct new obfuscators for an ES and EVES. Hada [41] constructed an obfuscator for an ES by combining Waters' signature [52] and the linear encryption scheme. The linear encryption scheme proposed by Boneh et al. [9] is as follows.

$\mathsf{L.Gen}(1^\lambda)$: On input security parameter λ, it generates $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, g, e) \xleftarrow{R} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$, selects exponents $x_e, y_e \xleftarrow{U} \mathbb{Z}_p$, and outputs $pk := (f_e, h_e) := (g^{x_e}, g^{y_e})$, $dk := (x_e, y_e)$.

$\mathsf{L.Enc}(pk_e, m)$: On input $m \in \mathbb{G}$ and $pk = (f_e, h_e)$ it selects $r, s \xleftarrow{U} \mathbb{Z}_p$ and outputs $c := (f_e^r, h_e^s, g^{r+s}m)$.

$\mathsf{L.Dec}(dk, c)$: On input $c = (c_1, c_2, c_3)$ and dk, it outputs $m := c_3/(c_1^{1/x_e} c_2^{1/y_e})$.

Hada's idea is as follows. Suppose that signature σ is computed as $\sigma = sk \cdot G(m)$ where $sk \in \mathbb{G}$ is the signing key, $m \in \mathbb{Z}_p$ is the message and $G : \mathbb{Z}_p \to \mathbb{G}$ is an efficiently computable function. Then, for ciphertext $c = \mathsf{L.Enc}(pk_e, sk)$, we can compute $\tilde{c} := c \cdot G(m) = \mathsf{L.Enc}(pk_e, sk \cdot G(m))$ by the homomorphic property of the linear encryption scheme. This is exactly an encrypted signature. The ciphertext of sk can be seen as an obfuscated circuit for encrypted signatures since the linear encryption scheme is semantically secure and no information about sk is revealed. We extend Hada's construction, that is, we combine sWdVES based on strongly unforgeable Waters' dual signature and the linear encryption scheme. However, Waters' dual signature is more complex than Waters' signature at Eurocrypt'05, so it is non-trivial whether we can directly use Hada's technique. In Waters' signature at Eurocrypt'05, the signing algorithm does not exponentiate sk, but in Waters' dual signature, it does. However, we can resolve this problem by using multiplicatively homomorphic property of the linear encryption scheme, that is, we can compute $c^r \cdot G(m) = \mathsf{L.Enc}(pk, sk^r \cdot G(m))$. Therefore, if we encrypt $sk = (g^\alpha, g^{\alpha a_1}, g^{a_1 a_2})$ by linear encryption, then we can construct an obfuscator for an ES/EVES. We propose secure obfuscators for an ES and EVES in the section.

5.1 Secure obfuscation

We review some definitions for secure obfuscators for ESs/EVESs.

Average-case secure obfuscation. We review an extended notion of the average-case virtual black-box property (ACVBP) by Hohenberger et al. [43]. They proposed the definition to overcome the impossibility results by Barak et al. [4].

Definition 15 (*Average-Case Secure Obfuscation* [43]) An efficient algorithm Obf that takes as input a (probabilistic) circuit and outputs a new (probabilistic) circuit, is an average-case secure obfuscator for the family $\mathcal{C} = \{\mathcal{C}_\lambda\}$ where $\mathcal{C}_\lambda$ are the circuits in $\mathcal{C}$ with input length λ if it satisfies the following properties:

Preserving Functionality: For any input length λ and $C \in \mathcal{C}_\lambda$, it holds that

$$\Pr_{\text{coins of Obf}}[\exists x \in \{0,1\}^\lambda : \Delta(\mathsf{Obf}(C)(x), C(x)) \text{ is not negligible in } \lambda]$$

is negligible in λ.
The distributions $\mathsf{Obf}(C)(x)$ and $C(x)$ are taken over $\mathsf{Obf}(C)$'s and C's random coins, respectively. Δ denotes statistical distance.

Polynomial Blowup: There exists a polynomial p such that for sufficiently large input length λ and any $C \in \mathcal{C}_\lambda$, $|\mathsf{Obf}(C)| \leq p(|C|)$.

Average-Case Secure Virtual Black-Box: For any efficient adversary $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ such that for every efficient distinguisher $\mathcal{D}$ and every auxiliary input $\mathsf{z} \in \mathsf{poly}(\lambda)$ (where poly is any polynomial),

$$\left| \Pr\left[\begin{array}{l} \mathcal{D}^C(\mathcal{A}(\mathsf{Obf}(C), \mathsf{z}), \mathsf{z}) \to 1 \mid \\ C \xleftarrow{\mathsf{R}} \mathcal{C}_\lambda \end{array} \right] - \Pr\left[\begin{array}{l} \mathcal{D}^C(\mathcal{S}^C(1^\lambda, \mathsf{z}), \mathsf{z}) \to 1 \mid \\ C \xleftarrow{\mathsf{R}} \mathcal{C}_\lambda \end{array} \right] \right|$$

is negligible in λ. The probability is over the selection of a random circuit C from $\mathcal{C}_\lambda$, and the coins of the distinguisher, simulator, oracle and obfuscator. Note that entities with black-box access to C cannot set C's random tape.

We review the ACVBP *with respect to dependent oracles* introduced by Hada [41] since our obfuscator is secure in this sense. Hada proposed the definition to consider the security of obfuscators for ES functionality. In the setting of ES functionalities, the adversary can access a signing oracle and is given a public-key used for encrypted signatures. However, the setting is not sufficient for meaningful security and a stronger security is required. That is, the adversary should be given not only the public-key but also an obfuscated circuit of an ES functionality (not black-box access). Hada showed that if an ES scheme satisfies the weaker security requirement (the adversary is given only a public-key) and its obfuscator satisfies ACVBP with respect to dependent oracles, then it also satisfies the stronger security requirement (the adversary is given an obfuscated circuit).

Definition 16 (*Average-Case Secure Obfuscation w.r.t. Dependent Oracles* [41]) An efficient algorithm Obf that takes as input a (probabilistic) circuit and outputs a new (probabilistic) circuit, is an average-case secure obfuscator for the family $\mathcal{C} = \{\mathcal{C}_\lambda\}$ if it satisfies the following properties:

Preserving Functionality: This is the same as Definition 15.

Polynomial Blowup: This is the same as Definition 15.

Average Case Virtual Black-Box Property w.r.t. Dependent Oracles: Let $T(C)$ be a set of oracles dependent on the circuit C. A circuit obfuscator Obf for $\mathcal{C}$ satisfies the ACVBP w.r.t. dependent oracle set T if the following condition holds. For any PPT $\mathcal{A}$, there exists a PPT $\mathcal{S}$ such that for any PPT $\mathcal{D}$ and every auxiliary input $\mathsf{z} \in \{0,1\}^{\mathsf{poly}(\lambda)}$,

$$\left| \Pr\left[\begin{array}{l} \mathcal{D}^{C,T(C)}(\mathcal{A}(\mathsf{Obf}(C), \mathsf{z}), \mathsf{z}) \to 1 \mid \\ C \xleftarrow{\mathsf{R}} \mathcal{C}_\lambda \end{array} \right] - \Pr\left[\begin{array}{l} \mathcal{D}^{C,T(C)}(\mathcal{S}^C(1^\lambda, \mathsf{z}), \mathsf{z}) \to 1 \mid \\ C \xleftarrow{\mathsf{R}} \mathcal{C}_\lambda \end{array} \right] \right|$$

is negligible in λ. Note that entities with black-box access to C and $T(C)$ cannot set C's and $T(C)$'s random tapes, respectively.

5.2 Security of ES and EVES

We review the definitions proposed by Hada [41]. In the following definitions, $\mathcal{SO}$ is a signing oracle, that is, if a message M is queried by the adversary, then $\mathcal{SO}$ returns a valid signature for M by using the secret key. First, we consider a weak security game where the adversary can access the signing oracle and is given a public key for encrypted signature.

Definition 17 (*Existential Unforgeability w.r.t. ES Functionality*) Let PKE and SIG be a pair of PKE and SIG schemes. SIG is existentially unforgeable w.r.t. ES functionality if the

following condition holds. For every PPT $\mathcal{A}$ and every auxiliary input $\mathsf{z} \in \{0,1\}^{\mathsf{poly}(\lambda)}$,

$$\Pr\left[\begin{array}{l|l} \mathsf{SIG.Vrfy}(vk, M^*, \sigma^*) \to 1 & \Gamma \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda); \\ \wedge M^* \notin Q & (vk, sk) \xleftarrow{\mathsf{R}} \mathsf{SIG.Gen}(\Gamma); \\ & (pk_e, dk) \xleftarrow{\mathsf{R}} \mathsf{PKE.Gen}(\Gamma); \\ & (M^*, \sigma^*, Q) \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{SO}(sk,\cdot)}(\Gamma, vk, pk_e, \mathsf{z}) \end{array}\right]$$

is negligible in λ.

Next, we consider a strong security game where the adversary is given an obfuscated circuit of an ES functionality.

Definition 18 (*Existential Unforgeability w.r.t. ES Obfuscator*) Let PKE and SIG be a pair of PKE and SIG schemes. Let $\mathsf{Obf}_{\mathsf{ES}}$ be a circuit obfuscator for ES_λ. SIG is existentially unforgeable w.r.t. $\mathsf{Obf}_{\mathsf{ES}}$ if the following condition holds. For any PPT $\mathcal{A}$ and every auxiliary input $\mathsf{z} \in \{0,1\}^{\mathsf{poly}(\lambda)}$,

$$\Pr\left[\begin{array}{l|l} \mathsf{SIG.Vrfy}(vk, M^*, \sigma^*) \to 1 & \Gamma \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda); \\ \wedge M^* \notin Q & (vk, sk) \xleftarrow{\mathsf{R}} \mathsf{SIG.Gen}(\Gamma); \\ & (pk_e, dk) \xleftarrow{\mathsf{R}} \mathsf{PKE.Gen}(\Gamma); \\ & C' \xleftarrow{\mathsf{R}} \mathsf{Obf}_{\mathsf{ES}}(C_{sk,pk_e}); \\ & (M^*, \sigma^*, Q) \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{SO}(sk,\cdot)}(\Gamma, vk, pk_e, C', \mathsf{z}) \end{array}\right]$$

is negligible in λ.

We extend these definitions to consider EVES functionalities. We can easily achieve this by adding a public-key of an adjudicator as input to $\mathcal{A}$ and giving the creation and adjudication oracles (the signing oracle in the security game for ES functionality is replaced with a oracle set that consists of those two oracles).

Definition 19 (*Existential Unforgeability w.r.t. EVES Functionality*) Let PKE and VES be a pair of PKE and VES schemes. VES is existentially unforgeable w.r.t. EVES functionality if the following condition holds. For any PPT $\mathcal{A}$ and every auxiliary input $\mathsf{z} \in \{0,1\}^{\mathsf{poly}(\lambda)}$,

$$\Pr\left[\begin{array}{l} \mathsf{VesVrfy}(apk, vk, \omega^*, M^*) \to 1 \wedge M^* \notin Q_{\mathsf{C}} \wedge M^* \notin Q_{\mathsf{A}} \mid \\ \Gamma \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda); \\ (apk, ask) \xleftarrow{\mathsf{R}} \mathsf{AdjGen}(\Gamma); (vk, sk) \xleftarrow{\mathsf{R}} \mathsf{Gen}(\Gamma); (pk_e, dk) \xleftarrow{\mathsf{R}} \mathsf{PKE.Gen}(\Gamma); \\ (M^*, \omega^*, Q_{\mathsf{C}}, Q_{\mathsf{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{CO}(sk,apk,\cdot),\mathcal{AO}(ask,apk,vk,\cdot,\cdot)}(\Gamma, vk, pk_e, apk, \mathsf{z}) \end{array}\right]$$

is negligible in λ.

Definition 20 (*Opacity w.r.t. EVES Functionality*) Let PKE and VES be a pair of PKE and VES schemes. VES is opaque w.r.t. EVES functionality if the following condition holds. For any PPT $\mathcal{A}$ and every auxiliary input $\mathsf{z} \in \{0,1\}^{\mathsf{poly}(\lambda)}$,

$$\Pr\left[\begin{array}{l} \mathsf{Vrfy}(vk, \sigma^*, M^*) \to 1 \wedge M^* \notin Q_{\mathsf{A}} \mid \\ \Gamma \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda); \\ (apk, ask) \xleftarrow{\mathsf{R}} \mathsf{AdjGen}(\Gamma); (vk, sk) \xleftarrow{\mathsf{R}} \mathsf{Gen}(\Gamma); (pk_e, dk) \xleftarrow{\mathsf{R}} \mathsf{PKE.Gen}(\Gamma); \\ (M^*, \sigma^*, Q_{\mathsf{C}}, Q_{\mathsf{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{CO}(sk,apk,\cdot),\mathcal{AO}(ask,apk,vk,\cdot,\cdot)}(\Gamma, vk, pk_e, apk, \mathsf{z}) \end{array}\right]$$

is negligible in λ.

Definition 21 (*Existential Unforgeability w.r.t. EVES Obfuscator*) Let PKE and VES be a pair of PKE and VES schemes. Let $\mathsf{Obf}_{\mathsf{EVES}}$ be a circuit obfuscator for EVES_λ. VES is unforgeability w.r.t. $\mathsf{Obf}_{\mathsf{EVES}}$ if the following condition holds. For any PPT $\mathcal{A}$ and every auxiliary input $\mathsf{z} \in \{0,1\}^{\mathsf{poly}(\lambda)}$,

$$\Pr\left[\begin{array}{l} \mathsf{VesVrfy}(apk, vk, \omega^*, M^*) \to 1 \wedge M^* \notin Q_{\mathsf{C}} \wedge M^* \notin Q_{\mathsf{A}} \mid \\ \Gamma \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda); \\ (apk, ask) \xleftarrow{\mathsf{R}} \mathsf{AdjGen}(\Gamma); (vk, sk) \xleftarrow{\mathsf{R}} \mathsf{Gen}(\Gamma); (pk_e, dk) \xleftarrow{\mathsf{R}} \mathsf{PKE.Gen}(\Gamma); \\ C' \xleftarrow{\mathsf{R}} \mathsf{Obf}_{\mathsf{EVES}}(C_{sk,pk_e,apk}); \\ (M^*, \omega^*, Q_{\mathsf{C}}, Q_{\mathsf{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{CO}l(sk,apk,\cdot),\mathcal{AO}(ask,apk,vk,\cdot,\cdot)}(\Gamma, vk, pk_e, apk, C', \mathsf{z}) \end{array}\right]$$

is negligible in λ.

Definition 22 (*Opacity w.r.t. EVES Obfuscator*) Let PKE and VES be a pair of PKE and VES schemes. Let $\mathsf{Obf}_{\mathsf{EVES}}$ be a circuit obfuscator for EVES_λ. VES is opaque w.r.t. $\mathsf{Obf}_{\mathsf{EVES}}$ if the following condition holds. For any PPT $\mathcal{A}$ and every auxiliary input $\mathsf{z} \in \{0,1\}^{\mathsf{poly}(\lambda)}$,

$$\Pr\left[\begin{array}{l} \mathsf{Vrfy}(vk, \sigma^*, M^*) \to 1 \wedge M^* \notin Q_{\mathsf{A}} \mid \\ \Gamma \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^\lambda); \\ (apk, ask) \xleftarrow{\mathsf{R}} \mathsf{AdjGen}(\Gamma); (vk, sk) \xleftarrow{\mathsf{R}} \mathsf{Gen}(\Gamma); (pk_e, dk) \xleftarrow{\mathsf{R}} \mathsf{PKE.Gen}(\Gamma); \\ C' \xleftarrow{\mathsf{R}} \mathsf{Obf}_{\mathsf{EVES}}(C_{sk,pk_e,apk}); \\ (M^*, \sigma^*, Q_{\mathsf{C}}, Q_{\mathsf{A}}) \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{CO}(sk,apk,\cdot),\mathcal{AO}(ask,apk,vk,\cdot,\cdot)}(\Gamma, vk, pk_e, apk, C', \mathsf{z}) \end{array}\right]$$

is negligible in λ.

Theorem 8 *Let oracle set $T(C_{\Gamma,sk,apk,pk_e})$ be $\{\mathcal{CO}(sk, apk, \cdot), \mathcal{AO}(ask, apk, vk, \cdot, \cdot)\}$ where C_{Γ,sk,apk,pk_e} is a circuit for the EVES functionality. If* Obf *for EVES satisfies the ACVBP w.r.t. dependent oracle set T, then the existential-unforgeability/opacity w.r.t. EVES functionality implies existential-unforgeability/opacity w.r.t. the EVES obfuscator, respectively.*

The proof of this theorem is almost the same as that of Hada [41], but we write it for confirmation.

Proof We show that if we assume that the existential-unforgeability/opacity w.r.t. an EVES functionality is satisfied but there exists $\mathcal{A}$ that breaks existential-unforgeability/opacity w.r.t. the EVES obfuscator, then Obf does not satisfy the ACVBP w.r.t. dependent oracle set T, that is, there exists a distinguisher $\mathcal{D}$ for ACVBP w.r.t. dependent oracle set T. We construct $\mathcal{D}$ as follows.

1. It is given an auxiliary-input z and a target circuit $\hat{C}$ that is either a real obfuscated circuit or a simulated circuit by a simulator.
2. It obtains keys (Γ, vk, apk, pk_e) by oracle access to C_{Γ,sk,apk,pk_e} with input keys.
3. It gives keys (Γ, vk, apk, pk_e), $\hat{C}$ and auxiliary input z to $\mathcal{A}$ as inputs.
4. It simulates the creation and adjudication oracles by using oracle access to oracle set $T(C_{\Gamma,sk,apk,pk_e})$.
5. It outputs 1 if and only if the winning condition of the existential-unforgeability/opacity w.r.t. the EVES obfuscator holds, respectively.

It holds that $T(C_{\Gamma,sk,apk,pk_e}) = \{\mathcal{CO}(sk, apk, \cdot), \mathcal{AO}(ask, apk, vk, \cdot, \cdot)\}$, so the oracle simulation above is perfect. If the target circuit is the real obfuscated circuit, then the simulation above perfectly simulates the experiment of existential-unforgeability/opacity w.r.t. the

EVES obfuscator and $\mathcal{A}$ has a non-negligible advantage. If the target circuit is the simulated one, then the distribution of inputs to $\mathcal{A}$ is not correct and it does not have a non-negligible advantage. Therefore, $\mathcal{D}$ can distinguish two circuits by using $\mathcal{A}$. □

5.3 Constructions of obfuscator for ES/EVES

To satisfy the virtual black-box property, we must construct simulator $\mathcal{S}$ that outputs an indistinguishable view from the real one. $\mathcal{S}$ does not have sk, so it encrypts random dummy elements over $\mathbb{G}$ instead of sk and sets them as an obfuscated circuit. Encrypted signatures/VESs under semantic secure PKE are indistinguishable from ciphertexts of dummy messages. By semantic security of the linear encryption, the view by $\mathcal{S}$ is indistinguishable from the real one. Thus, we can achieve obfuscators for ESs/EVESs.

5.3.1 Obfuscator for EVES

A naive construction of an EVES is the sequential composition of the VES scheme discussed in Sect. 4 and the linear encryption scheme (create-then-encrypt). We define the family of circuits as

$$\mathsf{EVES}_\lambda := \left\{ C_{\Gamma,sk,apk,pk_e} \;\middle|\; \begin{array}{l} (vk, sk) \xleftarrow{\mathsf{R}} \mathsf{Gen}(\Gamma), \\ (apk, ask) \xleftarrow{\mathsf{R}} \mathsf{AdjGen}(\Gamma), \\ (pk_e, dk) \xleftarrow{\mathsf{R}} \mathsf{L.Gen}(\Gamma) \end{array} \right\},$$

where each circuit $C_{\Gamma,sk,apk,pk_e} \in \mathsf{EVES}_\lambda$ (for notational convention, we often omit Γ, sk, apk, pk_e and use just C if it is clear from the context) is a probabilistic circuit that consists of the following two algorithms. $\mathsf{EVES}_{sk,apk,pk_e}(M)$ takes M as input, generates $\omega \xleftarrow{\mathsf{R}} \mathsf{Create}(sk, apk, M)$, and computes $C_\omega := \mathsf{L.Enc}(pk, \omega)$; $\mathsf{Keys}_{sk,apk,pk_e}(\mathsf{keys})$ takes keys as input and outputs (Γ, vk, apk, pk_e).

We introduce a re-randomization algorithm for the linear encryption $\mathsf{ReRand}(pk_e, c_1, c_2, c_3)$ that takes as inputs $pk_e = (f_e, h_e)$ and a ciphertext of the linear encryption, selects $r', s' \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, and outputs $(c_1 \cdot f_e^{r'}, c_2 \cdot h_e^{s'}, c_3 \cdot g^{r'+s'})$.

Our obfuscator for an EVES is described in Fig. 1. We can see ct_{sk} as a ciphertext of signing key sk. It is encrypted by pk_e and $apk = \zeta$. If we set $\rho_1' := \varrho_2 + \gamma\varrho_3 + \rho_1$ and $\rho_2' := \varrho_1 + \rho_2$, then, by using the homomorphic property of the linear and ElGamal encryption schemes, we can write

$$\begin{aligned}
(c_{1,1}, c_{1,2}, \Psi_1) &\xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{\alpha a_1} v^r g^{-a_1 a_2 \gamma} \cdot \zeta^{\rho_1'} = K_1), \\
(K_1', \hat{K}_1) &= (g^{\varrho_2+\varrho_3\gamma+\rho_1}, (g^b)^{\varrho_2+\varrho_3\gamma+\rho_1}) = (g^{\rho_1'}, (g^b)^{\rho_1'}), \\
(c_{2,1}, c_{2,2}, \Psi_2) &\xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{-\alpha} v_1^r g^{z_1} g^{a_2\gamma} \cdot \zeta^{\rho_2'} = K_2), \\
(K_2', \hat{K}_2) &= (g^{\varrho_1+\rho_2}, (g^{ba_1})^{\varrho_1+\rho_2}) = (g^{\rho_2'}, (g^{ba_1})^{\rho_2'})
\end{aligned}$$

by simple calculation. That is, we can write $C_i \xleftarrow{\mathsf{R}} \mathsf{Enc}(pk_e, K_i)$ for not only $i = 0, 3, 4, 5, 6, 7$ but also $i = 1, 2$. Thus, we can verify that the output of $\mathsf{Obf}_{\mathsf{EVES}}(M)$ is identically distributed to the output of $C_{sk,apk,pk_e}(M)$. We need the re-randomization procedure of the linear and ElGamal encryption schemes to prove the security.

Remark The proposed obfuscator encrypts elements in $\mathbb{Z}_p$ by using the linear encryption scheme. The message space of the linear encryption scheme is $\mathbb{G}$, so if we do not have an

Our obfuscator $\mathsf{Obf}_{\mathsf{EVES}}$ takes C as input and proceeds as follows.

1. Reads (Γ, sk, apk, pk_e) from C parses $sk = (g^{\alpha}, g^{\alpha a_1}, g^{a_1 a_2})$ and $apk = \zeta$.
2. Selects $\varrho_1, \varrho_2, \varrho_3 \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and computes

$$
\begin{aligned}
&(c_{\alpha,1}, c_{\alpha,2}, c_{\alpha,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{-\alpha})\\
&c_{\alpha} := (c_{\alpha,1}, c_{\alpha,2}, c_{\alpha,3} \cdot \zeta^{\varrho_1}, g^{\varrho_1}, g^{b a_1 \varrho_1})\\
&(c_{\alpha a_1,1}, c_{\alpha a_1,2}, c_{\alpha a_1,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{\alpha a_1})\\
&c_{\alpha a_1} := (c_{\alpha a_1,1}, c_{\alpha a_1,2}, c_{\alpha a_1,3} \cdot \zeta^{\varrho_2}, g^{\varrho_2}, g^{b \varrho_2})\\
&(c_{a_1 a_2,1}, c_{a_1 a_2,2}, c_{a_1 a_2,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{-a_1 a_2})\\
&c_{a_1 a_2} := (c_{a_1 a_2,1}, c_{a_1 a_2,2}, c_{a_1 a_2,3} \cdot \zeta^{\varrho_3}, g^{\varrho_3}, g^{b \varrho_3}).
\end{aligned}
$$

3. Sets $\mathsf{ct}_{sk} := (c_{\alpha}, c_{\alpha a_1}, c_{a_1 a_2})$.
4. Constructs an obfuscated circuit as follows. (1) On input keys, outputs (Γ, vk, apk, pk_e). (2) On input a message M,
   (a) Selects $r_1, r_2, \gamma, \rho_1, \rho_2, \mathsf{stag}, \varphi \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and sets $r := r_1 + r_2$.
   (b) Computes

$$
\begin{aligned}
&\Psi_1 := c_{\alpha a_1,3} \cdot v^r \cdot c_{a_1 a_2,3}^{\gamma} \cdot \zeta^{\rho_1},\\
&K_1' := c_{\alpha a_1,4} \cdot c_{a_1 a_2,4}^{\gamma} \cdot g^{\rho_1},\\
&\hat{K}_1 := c_{\alpha a_1,5} \cdot c_{a_1 a_2,5}^{\gamma} \cdot (g^b)^{\rho_1}\\
&c_{1,1} := c_{\alpha a_1,1} \cdot c_{a_1 a_2,1}^{\gamma},\ c_{1,2} := c_{\alpha a_1,2} \cdot c_{a_1 a_2,2}^{\gamma}\\
&C_1 := (c_{1,1}', c_{1,2}', c_{1,3}') \xleftarrow{\mathsf{R}} \mathsf{ReRand}(pk_e, (c_{1,1}, c_{1,2}, \Psi_1))\\
&C_1' \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, K_1')\\
&\hat{C}_1 \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, \hat{K}_1).
\end{aligned}
$$

   (c) Computes

$$
\begin{aligned}
&\Psi_2 := c_{\alpha,3} \cdot v_1^r \cdot g^{z_1} \cdot g^{a_2 \gamma} \cdot \zeta^{\rho_2},\ K_2' := c_{\alpha,4} \cdot g^{\rho_2},\ \hat{K}_2 := c_{\alpha,5} \cdot (g^{b a_1})^{\rho_2}\\
&c_{2,1} := c_{\alpha,1},\ c_{2,2} := c_{\alpha,2}\\
&C_2 := (c_{2,1}', c_{2,2}', c_{2,3}') \xleftarrow{\mathsf{R}} \mathsf{ReRand}(pk_e, (c_{2,1}, c_{2,2}, \Psi_2))\\
&C_2' \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, K_2')\\
&\hat{C}_2 \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, \hat{K}_2).
\end{aligned}
$$

   (d) Computes $K_3 := (g^b)^{-z_1}$, $K_4 := v_2^r g^{z_2} g^{a_1 \gamma}$, $K_5 := (g^b)^{-z_2}$, $K_6 := (g^b)^{r_2}$, $K_7 := g^{r_1}$, $\vartheta := H_k(M, K_3, \ldots, K_7, \mathsf{stag})$, $m := H_k(g^{\vartheta} \bar{h}^{\varphi})$, $K_0 := (u^m w^{\mathsf{stag}} h)^{r_1}$, $C_i \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, K_i)$ $(i = 0, 3, 4, 5, 6, 7)$, $C_t \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, \mathsf{stag})$, and $C_{\varphi} := \mathsf{L.Enc}(pk_e, \varphi)$.
   (e) Outputs $(C_0, \ldots, C_7, C_t, C_{\varphi}, C_1', C_2', \hat{C}_1, \hat{C}_2)$.

Fig. 1 Obfuscator for EVES

encoding between $\mathbb{Z}_p$ and $\mathbb{G}$, then we cannot use the linear encryption scheme. If we use a special elliptic curve defined by equation $y^2 = x^3 + b$, then we have such an encoding [8] and our construction works. Moreover, if we encrypt elements in $\mathbb{Z}_{p'}$ such that $p' < p$, then we can use an injective encoding proposed by Fouque, Joux, and Tibouchi [29].

Theorem 9 *Let* $T(C_{sk,apk,pk_e})$ *be* $\{\mathcal{CO}(sk, apk, \cdot), \mathcal{AO}(ask, apk, vk, \cdot, \cdot)\}$. *If the DLIN assumption holds, then* $\mathsf{Obf}_{\mathsf{EVES}}$ *satisfies ACVBP w.r.t. dependent oracle set* T.

Proof We construct a simulator, $\mathcal{S}$. It does not have secret signing key sk, so it acts as follows.

1. Queries oracle C on `keys` to obtain (Γ, vk, pk_e, apk).
2. Parses $vk = (g, g^b, g^{a_1}, g^{a_2}, g^{ba_1}, g^{ba_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, u, h, \bar{h}, k, e(g,g)^{\alpha a_1 b})$ and $apk = \zeta$.
3. Samples dummy elements $J_\alpha, J_{\alpha a_1}, J_{a_1 a_2} \xleftarrow{\mathsf{U}} \mathbb{G}$ and exponents $\varrho_1, \varrho_2, \varrho_3 \xleftarrow{\mathsf{U}} \mathbb{Z}_p$.
4. Generates $(c'_{\alpha,1}, c'_{\alpha,2}, c'_{\alpha,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, J_\alpha)$.
5. Sets $c_\alpha := (c'_{\alpha,1}, c'_{\alpha,2}, c'_{\alpha,3} \cdot \zeta^{\varrho_1}, g^{\varrho_1}, g^{ba_1\varrho_1})$.
6. Generates $(c'_{\alpha a_1,1}, c'_{\alpha a_1,2}, c'_{\alpha a_1,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, J_{\alpha a_1})$.
7. Sets $c_{\alpha a_1} := (c'_{\alpha a_1,1}, c'_{\alpha a_1,2}, c'_{\alpha a_1,3} \cdot \zeta^{\varrho_2}, g^{\varrho_2}, g^{b\varrho_2})$.
8. Generates $(c'_{a_1a_2,1}, c'_{a_1a_2,2}, c'_{a_1a_2,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, J_{a_1a_2})$.
9. Sets $c_{a_1a_2} := (c'_{a_1a_2,1}, c'_{a_1a_2,2}, c'_{a_1a_2,3} \cdot \zeta^{\varrho_3}, g^{\varrho_3}, g^{b\varrho_3})$.
10. Sets $ct_J := (c_\alpha, c_{\alpha a_1}, c_{a_1a_2})$.
11. Outputs circuit $(\Gamma, vk, apk, pk_e, \mathsf{Ct}_J)$.

We consider the following distributions.

$\mathsf{Real\text{-}C}(\mathcal{D}, \lambda, \mathsf{z})$:

$\Gamma \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$; $(pk_e, dk) \xleftarrow{\mathsf{R}} \mathsf{L.Gen}(\Gamma)$; $(vk, sk) \xleftarrow{\mathsf{R}} \mathsf{W.Gen}(\Gamma)$;

$\beta \xleftarrow{\mathsf{U}} \mathbb{Z}_p$; $(apk, ask) := (g^\beta, \beta)$;

$(c'_{\alpha,1}, c'_{\alpha,2}, c'_{\alpha,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{-\alpha})$; $c_\alpha := (c'_{\alpha,1}, c'_{\alpha,2}, c'_{\alpha,3} \cdot \zeta^{\varrho_1}, g^{\varrho_2}, g^{ba_1\varrho_1})$;

$(c'_{\alpha a_1,1}, c'_{\alpha a_1,2}, c'_{\alpha a_1,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{\alpha a_1})$;

$c_{\alpha a_1} := (c'_{\alpha a_1,1}, c'_{\alpha a_1,2}, c'_{\alpha a_1,3} \cdot \zeta^{\varrho_2}, g^{\varrho_2}, g^{b\varrho_2})$;

$(c'_{a_1a_2,1}, c'_{a_1a_2,2}, c'_{a_1a_2,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{-a_1a_2})$;

$c_{a_1a_2} := (c'_{a_1a_2,1}, c'_{a_1a_2,2}, c'_{a_1a_2,3} \cdot \zeta^{\varrho_3}, g^{\varrho_3}, g^{b\varrho_3})$;

$ct_{sk} := (c_\alpha, c_{\alpha a_1}, c_{a_1a_2})$;

$b \xleftarrow{\mathsf{R}} \mathcal{D}^C(\Gamma, vk, pk_e, apk, ct_{sk}, \mathsf{z})$.

$\mathsf{Sim\text{-}C}(\mathcal{D}, \lambda, \mathsf{z})$:

$\Gamma \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bmp}}(1^\lambda)$; $(pk_e, dk) \xleftarrow{\mathsf{R}} \mathsf{L.Gen}(\Gamma)$; $(vk, sk) \xleftarrow{\mathsf{R}} \mathsf{W.Gen}(\Gamma)$;

$\beta \xleftarrow{\mathsf{U}} \mathbb{Z}_p$; $(apk, ask) := (g^\beta, \beta)$;

$(c'_{\alpha,1}, c'_{\alpha,2}, c'_{\alpha,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, J_\alpha)$; $c_\alpha := (c'_{\alpha,1}, c'_{\alpha,2}, c'_{\alpha,3} \cdot \zeta^{\varrho_1}, g^{\varrho_1}, g^{ba_1\varrho_1})$;

$(c'_{\alpha a_1,1}, c'_{\alpha a_1,2}, c'_{\alpha a_1,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, J_{\alpha a_1})$;

$c_{\alpha a_1} := (c'_{\alpha a_1,1}, c'_{\alpha a_1,2}, c'_{\alpha a_1,3} \cdot \zeta^{\varrho_2}, g^{\varrho_2}, g^{b\varrho_2})$;

$(c'_{a_1a_2,1}, c'_{a_1a_2,2}, c'_{a_1a_2,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, J_{a_1a_2})$;

$c_{a_1a_2} := (c'_{a_1a_2,1}, c'_{a_1a_2,2}, c'_{a_1a_2,3} \cdot \zeta^{\varrho_3}, g^{\varrho_3}, g^{b\varrho_3})$;

$ct_J := (c_\alpha, c_{\alpha a_1}, c_{a_1a_2})$;

$b \xleftarrow{\mathsf{R}} \mathcal{D}^C(\Gamma, vk, pk_e, apk, ct_J, \mathsf{z})$.

It holds that $\mathsf{Real\text{-}C}(\mathcal{D}, \lambda, \mathsf{z}) = \mathcal{D}^C(\mathsf{Obf}(C), \mathsf{z})$ and $\mathsf{Sim\text{-}C}(\mathcal{D}, \lambda, \mathsf{z}) = \mathcal{D}^C(\mathcal{S}^C(1^\lambda, \mathsf{z}), \mathsf{z})$. We let $J_1 := J_\alpha$, $J_2 := J_{\alpha a_1}$, $J_3 := J_{a_1 a_2}$. We consider hybrid experiments Hyb_i^C, where $i \in [3]$ for $j = 1, \ldots, i$, $(c'_{1,1}, c'_{1,2}, c'_{1,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, J_j)$, and for $j = i+1, \ldots, 3$, $(c'_{1,1}, c'_{1,2}, c'_{1,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, sk[j])$, where $(sk[1], sk[2], sk[3]) := (g^{-\alpha}, g^{\alpha a_1}, g^{-a_1 a_2})$. It holds that $\mathsf{Hyb}_0^C = \mathsf{Real\text{-}C}(\mathcal{D}, \lambda, \mathbf{v}, \mathsf{z}) = \mathcal{D}^C(\mathsf{Obf}(C), \mathsf{z})$ and $\mathsf{Hyb}_3^C = \mathsf{Sim\text{-}C}(\mathcal{D}, \lambda, \mathsf{z}) = \mathcal{D}^C(\mathcal{S}^C(1^\lambda, \mathsf{z}), \mathsf{z})$. We can show $\mathsf{Hyb}_{i-1}^C \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_i^C$. If there exists distinguisher $\mathcal{D}$ that can distinguish these two variables, then we can construct adversary $\mathcal{A}$ which breaks IND-CPA security of the linear encryption scheme. $\mathcal{A}$ takes as input common parameter Γ, public key pk_e, and auxiliary input z. It selects $\beta \xleftarrow{\mathsf{U}} \mathbb{Z}_p$, sets $apk := g^\beta$ and $ask := \beta$, generates $(vk, sk) \xleftarrow{\mathsf{R}} \mathsf{sWd.Gen}(1^\lambda)$, selects $J_i \xleftarrow{\mathsf{U}} \mathbb{G}$, sets $m_0 := sk[i]$ and $m_1 := J_i$, and returns (m_0, m_1, vk). If $\mathcal{A}$ receives target ciphertext c^*, then $\mathcal{A}$ uses $\mathcal{D}$ as follows.

1. Parses $c^* = (c_1^*, c_2^*, c_3^*)$.
2. For $j = 1, \ldots, i-1$, selects $\varrho_j \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and sets $(c'_{i,1}, c'_{i,2}, c'_{i,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, J_j)$ and $c_{sk[j]} := (c'_{i,1}, c'_{i,2}, c'_{i,3} \cdot \zeta^{\varrho_j}, g^{\varrho_j}, \tilde{g}_j^{\varrho_j})$.
3. For $j = i$, selects $\varrho \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and sets $c_{sk[j]} := (c_1^*, c_2^*, c_3^* \cdot \zeta^{\varrho}, g^{\varrho}, \tilde{g}_j^{\varrho})$.
4. For $j = i+1, \ldots, 3$, selects $\varrho_j \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and sets $(c'_{i,1}, c'_{i,2}, c'_{i,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, sk[j])$ and $c_{sk[j]} := (c'_{i,1}, c'_{i,2}, c'_{i,3} \cdot \zeta^{\varrho_j}, g^{\varrho_j}, \tilde{g}_j^{\varrho_j})$.
5. Sets $ct_{sk} := (c_{sk[1]}, c_{sk[2]}, c_{sk[3]})$.
6. Gives $(\Gamma, vk, pk_e, apk, ct_{sk})$ as inputs.
7. Simulates $\mathcal{D}^{\mathcal{O}}(\Gamma, vk, pk_e, apk, ct_{sk}, \mathsf{z})$, where oracle queries can be perfectly simulated by using $sk = (g^\alpha, g^{\alpha a_1}, g^{a_1 a_2})$, $apk = \zeta$, and pk_e.
8. Outputs the output of $\mathcal{D}$.

Here, we set $\tilde{g}_1 := g^{ba_1}$, $\tilde{g}_2 := g^b$, $\tilde{g}_3 := g^b$.

If c^* is an encryption of $m_0 = sk[i]$, then the distribution is the same as Hyb_{i-1}^C. If c^* is an encryption of $m_1 = J_i$, then the distribution is the same as Hyb_i^C. Thus, if $\mathcal{D}$ can distinguish Hyb_{i-1}^C from Hyb_i^C, then $\mathcal{A}$ can break semantic security of the linear encryption scheme. By transitivity, it holds that $\mathsf{Real\text{-}C}(\mathcal{D}, \lambda, \mathsf{z}) = \mathsf{Hyb}_1^C \overset{\mathsf{c}}{\approx} \mathsf{Hyb}_3^C = \mathsf{Sim\text{-}C}(\mathcal{D}, \lambda, \mathsf{z})$ and the theorem follows. □

5.3.2 *Obfuscator for ES*

A naive construction of an ES is the sequential composition of a signature scheme and a encryption scheme (sign-then-encrypt), as Hada proposed [41]. We define the family of circuits as

$$\mathsf{ES}_\lambda := \{C_{\Gamma, sk, pk_e} | (vk, sk) \xleftarrow{\mathsf{R}} \mathsf{SIG.Gen}(\Gamma), (pk_e, dk) \xleftarrow{\mathsf{R}} \mathsf{PKE.Gen}(\Gamma)\},$$

where each circuit $C_{sk, pk_e} \in \mathsf{ES}_\lambda$ is a probabilistic circuit that consists of the following two algorithms: $\mathsf{ES}_{sk, pk_e}(M)$ takes M as input, generates $(\sigma_0, \sigma_1, \ldots, \sigma_7, \mathsf{stag}, \varphi) \xleftarrow{\mathsf{R}} \mathsf{sWd.Sign}(sk, M)$, computes $C_i \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, \sigma_i)$, $C_t \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, \mathsf{stag})$, $C_\varphi \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, \varphi)$, and outputs $C_\sigma := (C_0, \ldots, C_7, C_t, C_\varphi)$; $\mathsf{Keys}_{sk, pk_e}(\mathsf{keys})$ takes keys as input and outputs (Γ, vk, pk_e).

Our obfuscator for ES is easily obtained from our obfuscator for EVES and descried in Fig. 2.

Our obfuscator $\mathsf{Obf}_{\mathsf{ES}}$ takes C_{sk,pk_e} as input and proceeds as follows.

1. Reads (Γ, sk, pk_e) from C_{sk,pk_e}, obtains vk using Keys, and parses $sk = (g^{\alpha}, g^{\alpha a_1})$.
2. Computes $(c_{\alpha,1}, c_{\alpha,2}, c_{\alpha,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{-\alpha})$, $(c_{\alpha a_1,1}, c_{\alpha a_1,2}, c_{\alpha a_1,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{\alpha a_1})$, and $(c_{a_1 a_2,1}, c_{a_1 a_2,2}, c_{a_1 a_2,3}) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{-a_1 a_2})$.
3. Sets $\mathsf{Ct}_{sk} := (c_{\alpha}, c_{\alpha a_1} c_{a_1 a_2})$.
4. Generates an obfuscated circuit as follows. (1) On input keys, outputs (Γ, vk, pk_e). (2) On input a message M,
   (a) Selects $r_1, r_2, \gamma, z_1, z_2, \mathsf{stag}, \varphi \xleftarrow{\mathsf{U}} \mathbb{Z}_p$ and sets $r := r_1 + r_2$.
   (b) Computes $\Psi_1 := c_{\alpha a_1,3} \cdot v^r \cdot c_{a_1 a_2,3}^{\gamma}$, $c_{1,1} := c_{\alpha a_1,1} \cdot c_{a_1 a_2,1}^{\gamma}$, $c_{1,2} := c_{\alpha a_1,2} \cdot c_{a_1 a_2,2}^{\gamma}$, $C_1 := (c'_{1,1}, c'_{1,2}, c'_{1,3}) \xleftarrow{\mathsf{R}} \mathsf{ReRand}(pk_e, (c_{1,1}, c_{1,2}, \Psi_1))$.
   (c) Computes $\Psi_2 := c_{\alpha,3} \cdot v_1{}^{r} \cdot g^{z_1} \cdot g^{a_2 \gamma}$, $c_{2,1} := c_{\alpha,1}$, $c_{2,2} := c_{\alpha,2}$, $C_2 := (c'_{2,1}, c'_{2,2}, c'_{2,3}) \xleftarrow{\mathsf{R}} \mathsf{ReRand}(pk_e, (c_{2,1}, c_{2,2}, \Psi_2))$.
   (d) Computes $\sigma_3 := (g^b)^{-z_1}$, $\sigma_4 := v_2^r g^{z_2} g^{a_1 \gamma}$, $\sigma_5 := (g^b)^{-z_2}$, $\sigma_6 := (g^b)^{r_2}$, $\sigma_7 := g^{r_1}$, $\vartheta := H_k(M \parallel \Sigma_2)$ where $\Sigma_2 := (\sigma_3, \ldots, \sigma_7, \mathsf{stag})$, $m := H_k(g^{\vartheta} h^{\varphi})$, $\sigma_0 := (u^m w^{\mathsf{stag}} h)^{r_1}$.
   (e) Computes $C_i \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, \sigma_i)$ $(i = 0, 3, 4, 5, 6, 7)$, $C_t \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, \mathsf{stag})$, $C_{\varphi} \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, \varphi)$, and outputs $(C_0, \ldots, C_7, C_t, C_{\varphi})$.

Fig. 2 Obfuscator for ES

It holds that

$$(c_{1,1}, c_{1,2}, \Psi_1) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{\alpha a_1} \cdot v^r \cdot g^{-a_1 a_2 \gamma})$$
$$(c_{2,1}, c_{2,2}, \Psi_2) \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, g^{-\alpha} \cdot v_1^r \cdot g^{z_1} \cdot g^{a_2 \gamma})$$

by simple calculation. Thus, we can write $C_i \xleftarrow{\mathsf{R}} \mathsf{L.Enc}(pk_e, \sigma_i)$ for not only $i = 0, 3, 4, 5, 6, 7$ but also $i = 1, 2$. We can easily verify that $\mathsf{ES}_{sk,pk_e}(M)$ and $\mathsf{Obf}_{\mathsf{ES}}(M)$ are identically distributed.

Theorem 10 *Let $T(C_{sk,pk_e})$ be $\mathcal{SO}(sk, \cdot)$. If the DLIN assumption holds, then $\mathsf{Obf}_{\mathsf{ES}}$ satisfies the ACVBP w.r.t. dependent oracle set T.*

We can prove Theorem 10 by using the same proof technique as that of Theorem 9.

Acknowledgments The authors would like to thank Mehdi Tibouchi for his useful comments on encodings between $\mathbb{Z}_p$ and $\mathbb{G}$. The authors would like to thank the anonymous reviewers of PKC 2012, 2013, and Designs, Codes and Cryptography for their useful comments and suggestions.

References

1. Abe M., Chase M., David B., Kohlweiss M., Nishimaki R., Ohkubo M.: Constant-size structure-preserving signatures: generic constructions and simple assumptions. In: ASIACRYPT'12. Lecture Notes in Computer Science, vol. 7658, pp. 4–24. Springer, Berlin (2012).
2. Asokan N., Shoup V., Waidner M.: Optimistic fair exchange of digital signatures (extended abstract). In: EUROCRYPT'98. Lecture Notes in Computer Science, vol. 1403, pp. 591–606. Springer, Berlin (1998).
3. Bao F., Deng R.H., Mao W.: Efficient and practical fair exchange protocols with off-line TTP. In: IEEE Symposium on Security and Privacy'98, pp. 77–85. IEEE Computer Society, Washington, DC (1998).
4. Barak B., Goldreich O., Impagliazzo R., Rudich S., Sahai A., Vadhan S.P., Yang K.: On the (im)possibility of obfuscating programs. J. ACM **59**(2), 6 (2012).
5. Belenkiy M., Camenisch J., Chase M., Kohlweiss M., Lysyanskaya A., Shacham H.: Randomizable proofs and delegatable anonymous credentials. In: CRYPTO'09. Lecture Notes in Computer Science, vol. 5677, pp. 108–125. Springer, Berlin (2009).

6. Bitansky N., Canetti R.: On strong simulation and composable point obfuscation. In: CRYPTO'10. Lecture Notes in Computer Science, vol. 6223, pp. 520–537 (2010).
7. Boneh D., Boyen X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptol. **21**(2), 149–177 (2008).
8. Boneh D., Franklin M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003).
9. Boneh D., Boyen X., Shacham H.: Short group signatures. In: CRYPTO'04. Lecture Notes in Computer Science, vol. 3152, pp. 41–55. Springer, Berlin (2004).
10. Boneh D., Lynn B., Shacham H.: Short signatures from the Weil pairing. J. Cryptol. **17**(4), 297–319 (2004).
11. Boneh D., Shen E., Waters B.: Strongly unforgeable signatures based on computational Diffie–Hellman. In: PKC'06. Lecture Notes in Computer Science, vol. 3958, pp. 229–240. Springer, Berlin (2006).
12. Boneh D., Gentry C., Lynn B., Shacham H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: EUROCRYPT'03. Lecture Notes in Computer Science, vol. 2656, pp. 416–432. Springer, Berlin (2003).
13. Brakerski Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: CRYPTO'12. Lecture Notes in Computer Science, vol. 7417, pp. 868–886. Springer, Berlin (2012).
14. Brakerski Z., Vaikuntanathan V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS, pp. 97–106. IEEE Press, New York, NY (2011).
15. Brakerski Z., Gentry C., Vaikuntanathan V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS, pp. 309–325. ACM Press, New york, NY (2012).
16. Canetti R.: Towards realizing random oracles: Hash functions that hide all partial information. In: CRYPTO'97. Lecture Notes in Computer Science, vol. 1294, pp. 455–469. Springer, Berlin (1997).
17. Canetti R., Dakdouk R.R.: Obfuscating point functions with multibit output. In: EUROCRYPT'08. Lecture Notes in Computer Science, vol. 4965, pp. 489–508. Springer, Berlin (2008).
18. Canetti R., Varia M.: Non-malleable obfuscation. In: TCC'09. Lecture Notes in Computer Science, vol. 5444, pp. 73–90. Springer, Berlin (2009).
19. Canetti R., Micciancio D., Reingold O.: Perfectly one-way probabilistic hash functions (preliminary version). In: STOC'98, pp. 131–140. ACM Press, New York, NY (1998).
20. Canetti R., Rothblum G.N., Varia M.: Obfuscation of hyperplane membership. In: TCC'10. Lecture Notes in Computer Science, vol. 5978, pp. 72–89. Springer, Berlin (2010).
21. Canetti R., Kalai Y.T., Varia M., Wichs D.: On symmetric encryption and point obfuscation. In: TCC'10. Lecture Notes in Computer Science, vol. 5978, pp. 52–71. Springer, Berlin (2010).
22. Chandran N., Chase M., Vaikuntanathan V.: Collusion resistant obfuscation and functional re-encryption. In: TCC'12. Lecture Notes in Computer Science, vol. 7194, pp. 404–421. Springer, Berlin (2012).
23. Cheng R., Zhang B., Zhang F.: Secure obfuscation of encrypted verifiable encrypted signatures. In: ProvSec'11. Lecture Notes in Computer Science, vol. 6980, pp. 188–203. Springer, Berlin (2011).
24. Cheon J.H., Coron J.S., Kim J., Lee M.S., Lepoint T., Tibouchi M., Yun A.: Batch fully homomorphic encryption over the integers. In: EUROCRYPT'13. Lecture Notes in Computer Science, vol. 7881, pp. 315–335. Springer, Berlin (2013).
25. Coron J.S., Naccache D.: Boneh et al'.s k-element aggregate extraction assumption is equivalent to the Diffie-Hellman assumption. In: ASIACRYPT'03. Lecture Notes in Computer Science, vol. 2894, pp. 392–397. Springer, Berlin (2003).
26. Coron J.S., Mandal A., Naccache D., Tibouchi M.: Fully homomorphic encryption over the integers with shorter public keys. In: CRYPTO'11. Lecture Notes in Computer Science, vol. 6841, pp. 487–504. Springer, Berlin (2011).
27. Dodis Y., Smith A.: Correcting errors without leaking partial information. In: STOC'05, pp. 654–663. ACM Press, New York, NY (2005).
28. Dodis Y., Lee P.J., Yum D.H.: Optimistic fair exchange in a multi-user setting. In: PKC'07. Lecture Notes in Computer Science, vol. 4450, pp. 118–133. Springer, Berlin (2007).
29. Fouque P.A., Joux A., Tibouchi M.: Injective encodings to elliptic curves. In: ACISP'13. Lecture Notes in Computer Science, vol. 7959, pp. 203–218. Springer, Berlin (2013).
30. Fuchsbauer G.: Commuting signatures and verifiable encryption. In: EUROCRYPT'11. Lecture Notes in Computer Science, vol. 6632, pp. 224–245. Springer, Berlin (2011).
31. Garg S., Gentry C., Halevi S.: Candidate multilinear maps from ideal lattices. In: EUROCRYPT'13. Lecture Notes in Computer Science, vol. 7881, pp. 1–17. Springer, Berlin (2013).
32. Garg S., Gentry C., Halevi S., Raykova M., Sahai A., Waters B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS'13. IEEE press, New York, NY (2013).
33. Gentry C.: Fully homomorphic encryption using ideal lattices. In: STOC'09, pp. 169–178. ACM Press, New York, NY (2009).

34. Gentry C., Halevi S.: Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In: FOCS'11, pp. 107–116. IEEE Press, New York, NY (2011).
35. Gentry C., Halevi S.: Implementing Gentry's fully-homomorphic encryption scheme. In: EUROCRYPT'11. Lecture Notes in Computer Science, vol. 6632, pp. 129–148. Springer, Berlin (2011).
36. Gentry C., Halevi S., Smart N.P.: Fully homomorphic encryption with polylog overhead. In: EUROCRYPT'12. Lecture Notes in Computer Science, vol. 7237, pp. 465–482. Springer, Berlin (2012).
37. Gentry C., Sahai A., Waters B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: CRYPTO'13 (1). Lecture Notes in Computer Science, vol. 8042, pp. 75–92. Springer, Berlin (2013).
38. Goldwasser S., Kalai Y.T.: On the impossibility of obfuscation with auxiliary input. In: FOCS'05, pp. 553–562. IEEE press, New York, NY (2005).
39. Goldwasser S., Rothblum G.N.: On best-possible obfuscation. In: TCC'07. Lecture Notes in Computer Science, vol. 4392, pp. 194–213. Springer, Berlin (2007).
40. Hada S.: Zero-knowledge and code obfuscation. In: ASIACRYPT'00. Lecture Notes in Computer Science, vol. 1976, pp. 443–457. Springer, Berlin (2000).
41. Hada S.: Secure obfuscation for encrypted signatures. In: EUROCRYPT'10. Lecture Notes in Computer Science, vol. 6110, pp. 92–112. Springer, Berlin (2010).
42. Hofheinz D., Malone-Lee J., Stam M.: Obfuscation for cryptographic purposes. J. Cryptol. **23**(1), 121–168 (2010).
43. Hohenberger S., Rothblum G.N., Shelat A., Vaikuntanathan V.: Securely obfuscating re-encryption. J. Cryptol. **24**(4), 694–719 (2011).
44. Lu S., Ostrovsky R., Sahai A., Shacham H., Waters B.: Sequential aggregate signatures and multisignatures without random oracles. In: EUROCRYPT'06. Lecture Notes in Computer Science, vol. 4004, pp. 465–485. Springer, Berlin (2006).
45. Lu S., Ostrovsky R., Sahai A., Shacham H., Waters B.: Sequential aggregate signatures, multisignatures, and verifiably encrypted signatures without random oracles. J. Cryptol. **26**(2), 340–373 (2013).
46. Lynn B., Prabhakaran M., Sahai A.: Positive results and techniques for obfuscation. In: EUROCRYPT'04. Lecture Notes in Computer Science, vol. 3027, pp. 20–39. Springer, Berlin (2004).
47. Rückert M.: Verifiably encrypted signatures from RSA without NIZKs. In: INDOCRYPT'09. Lecture Notes in Computer Science, vol. 5922, pp. 363–377. Springer, Berlin (2009).
48. Rückert M., Schröder D.: Security of verifiably encrypted signatures and a construction without random oracles. In: Pairing'09. Lecture Notes in Computer Science, vol. 5671, pp. 17–34. Springer, Berlin (2009).
49. Rückert M., Schneider M., Schröder D.: Generic constructions for verifiably encrypted signatures without random oracles or NIZKs. In: ACNS'10. Lecture Notes in Computer Science, vol. 6123, pp. 69–86 (2010).
50. Shoup V.: Lower bounds for discrete logarithms and related problems. In: EUROCRYPT'97, LNCS, vol. 1233, pp. 256–266 (1997).
51. van Dijk M., Gentry C., Halevi S., Vaikuntanathan V.: Fully homomorphic encryption over the integers. In: EUROCRYPT'10. Lecture Notes in Computer Science, vol. 6110, pp. 24–43. Springer, Berlin (2010).
52. Waters B.: Efficient identity-based encryption without random oracles. In: EUROCRYPT'05. Lecture Notes in Computer Science, vol. 3494, pp. 114–127. Springer, Berlin (2005).
53. Waters B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: CRYPTO'09. Lecture Notes in Computer Science, vol. 5677, pp. 619–636. Springer, Berlin (2009). Full version available from http://eprint.iacr.org/2009/385.
54. Wee H.: On obfuscating point functions. In: STOC'05, pp. 523–532. ACM Press, New York, NY (2005).
55. Zhang F., Safavi-Naini R., Susilo W.: Efficient verifiably encrypted signature and partially blind signature from bilinear pairings. In: INDOCRYPT'03. Lecture Notes in Computer Science, vol. 2904, pp. 191–204. Springer Berlin (2003).