Structure-Preserving Certificateless Encryption and Its Application

Tao Zhang, Huangting Wu, and Sherman S. M. Chow(✉)

Department of Information Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong
sherman@ie.cuhk.edu.hk

Abstract. Certificateless encryption (CLE) combines the advantages of public-key encryption (PKE) and identity-based encryption (IBE) by removing the certificate management of PKE and the key escrow problem of IBE. In this paper, we propose structure-preserving CLE schemes. Structure preservation enables efficient non-interactive proof of certain ciphertext properties, thus supporting efficient modular constructions of advanced cryptographic protocols with a simple design.

As an illustration, we propose a structure-preserving group signature scheme with certified limited (CL) opening from structure-preserving CLE. CL opening allows a master certifier to certify openers. The opener who is the designated one for a group signature can open it (i.e., revoke its anonymity). Neither the certifier nor any non-designated openers can perform the opening. The structure-preserving property of our scheme can also hide who is the designated opener among a list of possibilities.

Keywords: Structure-preserving cryptography · Certificateless encryption

1 Introduction

Structure-preserving cryptography is a promising paradigm which enables modular designs of advanced cryptographic protocols, due to its compatibility with efficient non-interactive zero-knowledge proof over the same structure, such as Groth-Sahai proof [21]. Abe *et al.* [3] constructed structure-preserving signature (SPS) schemes which sign on a vector of group elements. They also used SPS to design concurrently-secure group signatures among other applications. Camenisch *et al.* [10] proposed the first CCA-secure structure-preserving encryption (SPE) scheme. Specifically, their integrity check before the final step in the decryption algorithm does not hash the ciphertext, which is often required in other CCA-secure scheme and its presence may hinder its compatibility with

S. S. M. Chow—Supported by General Research Funds (CUHK 14210217) of the Research Grants Council, Hong Kong.

Groth-Sahai proof. SPE found applications in joint computation of ciphertext [10].

Many studies have been carried out on basic primitives which are structure-preserving; yet, despite the numerous applications of identity-based encryption (IBE), (fully) structure-preserving IBE (SP-IBE) has never been studied. SP-IBE requires the public parameters, the plaintext, the ciphertext, and the user identity, consists of only group elements. The user identity is of a particular interest. For existing pairing-based IBE schemes, the user identity ID is not a group element, but consists of integers or bits. Usually, these schemes hash ID to a group element or to an exponent, which kills the original structure of the identity. A notable exception is proposed by Libert and Joye [25], where everything except the user identity consists of only group element. Such a scheme found applications in group signature with message-dependent opening.

It is well known that any IBE construction implies an implicit signature scheme. One may wonder if any of the existing SPS schemes feature a signature which can be used as a decryption key for a certain SP-IBE scheme. In other words, any valid signature can recover the ephemeral session key in the SP-IBE scheme, by pairing up the signature (as a user decryption key) with the ciphertext. However, to the best of the authors' knowledge, existing SPS signatures cannot be used for this purpose. The reason is that the verification equation requires the computation of a pairing term where both of its input comes from the signature. From another perspective, if one is going to generate a ciphertext such that it is decryptable by such a decryption key, the pairing will involve an unknown term since the decryption key is unknown to the encryptor. In this paper, towards enriching the class of structure-preserving cryptographic schemes, we move our focus to structure-preserving certificateless encryption (SP-CLE).

Certificateless encryption (CLE), introduced by Al-Riyami and Paterson [5], strikes a balance between IBE and public-key encryption (PKE). In traditional PKE, an encryptor needs to verify a certificate which ensures that a given public key belongs to the recipient. This requires a public-key infrastructure to support the storage and distribution of the certificates. The sender also needs to verify the certificate before encrypting. To overcome this weakness of PKE, IBE provides another solution in which every identity string can be mapped to a public key via a publicly computable function. The corresponding private decryption key can only be generated by the key generation center (KGC). Such kind of key-escrow is inherent and introduces serious security concerns. CLE removes key-escrow by requiring both the partial decryption key from the KGC and a user secret in decryption. Yet, unlike PKE, CLE does not need any infrastructure to authenticate users' public keys. In contrast, implicit certification is ensured by the KGC since decryption would be impossible without the partial decryption key.

In the CLE formulation of Al-Riyami and Paterson [5], a user can compute and release its user public key before it obtains its partial decryption key from the KGC. Such formulation implies the existence of both PKE and IBE [18]. Indeed, CLE can be constructed generically from IBE and PKE. Baek, Safavi-Naini, and Susilo [6] formulated an alternative CLE notion in which a user must obtain its partial decryption key from the KGC before it can compute its user public key. Such formulation no longer implies IBE. Consequently, Baek *et al.* constructed CLE from Schnorr signatures and ElGamal PKE. This gives us hope in designing SP-CLE without first designing SP-IBE.

Another distinctive feature of CLE is its security under strong decryption [5]. A strong decryption oracle can provide correct decryption even when the public key of a user is replaced by the adversary, without requiring the adversary to surrender the decryption key corresponding to the replaced public key. This level of security has important applications in complete non-malleability [7,14]. Many CLE schemes, under either formulation [5,6], rely on the random oracle to simulate the strong decryption oracle. Dent *et al.* [19] proposed the first CLE scheme featuring strong decryption in the standard model. Yet, Groth-Sahai proof cannot prove about its ciphertext well-formedness due to the presence of a hash.

Our Contribution. We propose the first SP-CLE schemes over groups with bilinear map $e : \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$. We first present a construction encrypting plaintexts in $\mathbb{G}_T$ which is secure against chosen-plaintext attacks (CPA). Then, we extend it to support message space of $\mathbb{G}$ (or $\mathbb{H}$). Finally, we show how to extend it for security against replayable chosen-ciphertext attacks (RCCA). Our proofs do not rely on random oracles; yet, they are proven in the generic group model.

To illustrate the application of SP-CLE, we then build a (partially) structure-preserving group signature scheme with certified limited (CL) opening from our SP-CLE. We defer the relevant introduction and motivation to Sect. 5.

2 Preliminaries

2.1 Bilinear Group

For bilinear group context $\mathcal{G} = (\mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, p, g, h)$, $\mathbb{G}, \mathbb{H}$, and $\mathbb{G}_T$ are groups of prime order p, where g and h are random generators for $\mathbb{G}$ and $\mathbb{H}$ respectively. A bilinear map $e : \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$ is a non-trivial and efficiently computable pairing function such that, for all $u \in \mathbb{G}$, $v \in \mathbb{H}$, $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$. In Type-I groups, $\mathbb{G} = \mathbb{H}$. For Type II, there exists an efficient mapping from $\mathbb{G}$ to $\mathbb{H}$ but not the other way around. For Type III, there exists no efficient mapping between $\mathbb{G}$ and $\mathbb{H}$. This paper uses Type-III groups which is the most efficient.

2.2 Groth-Sahai Proof System

Groth and Sahai [21] proposed several instantiations for efficient NIZK proof of knowledge, for statements about group elements satisfying a pairing product equation. Their proof system (called Groth-Sahai proof hereinafter) consists of four algorithms $\mathcal{GS} = (\texttt{Setup}, \texttt{Prove}, \texttt{Verify}, \texttt{Extract})$. $\texttt{Setup}(1^\lambda)$ generates the common reference string $\texttt{crs}$ and the extraction key $\texttt{ek}$. $\texttt{Prove}()$ takes in a witness and a statement to generate a proof of the statement w.r.t. the witness. We use the notation PoK to refer to a proof. $\texttt{Verify}()$ outputs 1 on a valid proof.

$\mathcal{GS}$ uses a commitment scheme ($\mathtt{Commit}()$ with commitment key ck) as a building block, committing the witness to prepare for an NIZK proof of knowledge. The remaining algorithm $\mathtt{Extract}()$ extracts the hidden element from a proof with the extraction key ek. The commitment key ck is publicly accessible, and the extraction key ek is only accessible to a knowledge extractor.

2.3 Structure-Preserving Signature

A signature scheme is a tuple of four algorithms ($\mathtt{Setup}, \mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify}$). It is structure preserving [3] if the verification key, the messages, and the signatures consist of only group elements, and the verification algorithm only evaluates pairing product equations of the form $\prod_i \prod_j e(G_i, H_j)^{a_{ij}} = 1_{\mathbb{G}_T}$, where $G_i \in \mathbb{G}$ and $H_j \in \mathbb{H}$ are group elements forming the verification key, the message(s), and the public parameters, $a_{ij} \in \mathbb{Z}_p$ are constants, and the element $1_{\mathbb{G}_T}$ is the identity element in $\mathbb{G}_T$. An SPS is existentially unforgeable under chosen-message attack (EUF-CMA) if no probabilistic polynomial-time (PPT) adversary can output a valid forgery (M, σ), given the public parameters param, the verification key vk, and a signing oracle for adversarially chosen messages but M is never queried. If the signing oracle can only be queried once, the scheme is called one-time secure.

3 Definitions of Certificateless Encryption

We follow Baek *et al.*'s formulation [6,31], where the user public key can only be generated after the user has interacted with the KGC. We add one algorithm $\mathtt{SetUserSec}()$ which is executed by a user and include a partial user public key ppk as part of the input of $\mathtt{Issue}$, an algorithm executed by the KGC for the user. These changes have been discussed in the seminal work [5]. A benefit is that the CLE scheme can reach trust level 3 named by Girault [20] as a traditional PKI. The CLE definition in this paper consists of seven algorithms ($\mathtt{Setup}$, $\mathtt{MKeyGen}$, $\mathtt{SetUserSec}, \mathtt{Issue}, \mathtt{UKeyGen}, \mathtt{Enc}, \mathtt{Dec}$):

– $\mathtt{Setup}(1^\lambda) \rightarrow \mathsf{param}$. This algorithm takes in a security parameter 1^λ and outputs the parameter param. We assume param is an implicit input to all other algorithms.
– $\mathtt{MKeyGen}() \rightarrow (\mathsf{mpk}, \mathsf{msk})$. The KGC runs this algorithm. It generates the master public-private key pair. The KGC publishes the master public key mpk and keeps the master secret key msk in private.
– $\mathtt{SetUserSec}(\mathsf{mpk}, \mathsf{ID}) \rightarrow (\mathsf{ppk}, \mathsf{uk})$. A user takes as input the master public key mpk and its own identity ID, and outputs a partial user public key ppk and a user secret value uk.
– $\mathtt{Issue}(\mathsf{msk}, \mathsf{mpk}, \mathsf{ID}, \mathsf{ppk}) \rightarrow \mathsf{psk}$. The KGC takes in the master public-private key pair, a user identity ID, and a user partial public key ppk to generate the user partial secret key psk for ID.
– $\mathtt{UKeyGen}(\mathsf{mpk}, \mathsf{ppk}, \mathsf{psk}, \mathsf{uk}) \rightarrow (\mathsf{upk}, \mathsf{usk})$. With respect to the master public key mpk and a partial public key ppk, the user uses its partial secret key psk

and user secret value uk to generate the user public-private key pair $(\mathsf{upk}, \mathsf{usk})$. The user publishes the user public key upk and keeps the full private key usk in private.
- $\mathtt{Enc}(\mathsf{mpk}, \mathsf{upk}, \mathsf{ID}, M) \rightarrow C$. This algorithm takes in the master public key mpk, the user public key upk, and an identity ID, to encrypt a plaintext M.
- $\mathtt{Dec}(\mathsf{mpk}, \mathsf{upk}, \mathsf{usk}, C) \rightarrow M$. This deterministic algorithm takes in the master public key, the user public-private key pair, and a ciphertext to recover the plaintext M, or the error symbol $\perp$ when C is invalid.

A CLE scheme is said to be correct if for any integer λ, $\mathsf{param} \leftarrow \mathtt{Setup}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathtt{MKeyGen}(\mathsf{param})$, any string ID, $(\mathsf{ppk}, \mathsf{uk}) \leftarrow \mathtt{SetUserSec}(\mathsf{mpk}, \mathsf{ID})$, $\mathsf{psk} \leftarrow \mathtt{Issue}(\mathsf{msk}, \mathsf{mpk}, \mathsf{ID}, \mathsf{ppk})$, $(\mathsf{upk}, \mathsf{usk}) \leftarrow \mathtt{UKeyGen}(\mathsf{mpk}, \mathsf{ppk}, \mathsf{psk}, \mathsf{uk})$, any message M, and $C \leftarrow \mathtt{Enc}(\mathsf{mpk}, \mathsf{upk}, \mathsf{ID}, M)$, we have $M \leftarrow \mathtt{Dec}(\mathsf{mpk}, \mathsf{upk}, \mathsf{usk}, C)$.

A CLE scheme is said to be structure-preserving if the encryption and decryption algorithms only operate on group elements. In other words, all elements in mpk, upk, and usk, the identity ID, the message M to encrypt, and the ciphertext C to be produced, are all group elements. We call a CLE scheme to be partially structure-preserving if some elements in ID, M, or C are not group elements, e.g., ID in Libert and Joye [25] and M and C in our basic scheme.

We consider two kinds of adversaries. Type-I adversary $\mathcal{A}_I$ models the malicious users who can replace the public key of a victim user to other "unauthenticated" public keys since there is no certificate. Type-II adversary $\mathcal{A}_{II}$ models an honest-but-curious KGC who can obtain partial decryption keys for the users, but cannot replace the user public key for any user. Obviously, these two types of adversaries cannot collude. We first describe the oracles available to $\mathcal{A}_I/\mathcal{A}_{II}$:

- **Replace Public Key.** The adversary submits ID and a user public key upk' to this oracle, which replaces the previous user public key of ID to upk'.
- **Extract Partial Secret Key.** The adversary submits an identity ID to this oracle. This oracle returns the partial secret key psk generated for ID.
- **Extract Full Private Key.** The adversary supplies an identity ID to this oracle. This oracle returns the full private key usk generated for ID.
- **Strong Decrypt.** The adversary supplies an identity ID and a ciphertext C. This oracle creates a full private key usk for ID if it is not previously generated, decrypts C with usk even if upk of ID used in C has been replaced, and sends the plaintext to the adversary.
- **Weak SV Decrypt.** The adversary supplies an identity ID, a user secret uk', and a ciphertext C to this oracle. This oracle creates usk' for ID with the real psk and uk', and decrypts C. The oracle returns the plaintext result.

Definition 1 (IND-CPA security against Type-I adversary). *A CLE scheme is indistinguishable under chosen-plaintext attacks (IND-CPA secure) against Type-I adversary if $Adv_{\mathcal{A}_I}^{IND\text{-}CPA}$ is negligible.*

Setup. The challenger $\mathcal{C}$ executes $\mathtt{Setup}()$ and publishes param.

Master Key Generation. $\mathcal{C}$ runs $\mathtt{MKeyGen}()$, sends mpk to $\mathcal{A}_I$, and keeps msk private.

Query Phase. The adversary $\mathcal{A}_I$ first makes registration queries for a polynomial number of identities $\{\mathsf{ID}_i\}_{i=1}^{q}$. $\mathcal{C}$ runs $\mathsf{psk}_i \leftarrow \mathtt{Issue}(\mathsf{msk}, \mathsf{mpk}, \mathsf{ID}_i, \mathsf{ppk}_i)$ and $(\mathsf{upk}_i, \mathsf{usk}_i) \leftarrow \mathtt{UKeyGen}(\mathsf{mpk}, \mathsf{psk}_i)$, and publishes upk_i for $i \in [1, q]$. Then, $\mathcal{A}_I$ can make **Replace Public Key**, **Extract Partial Secret Key**, and **Extract Full Private Key** queries on any registered identity, but $\mathcal{A}_I$ cannot request for the partial or full private key of an identity ID after replacing its upk.

Challenge. $\mathcal{A}_I$ submits an identity ID^* and two messages M_0, M_1 to $\mathcal{C}$. $\mathcal{C}$ aborts this game if any of the following events happen.

- $\mathcal{A}_I$ made **Extract Full Private Key** query on ID^*.
- $\mathcal{A}_I$ made both **Replace Public Key** query and **Extract Partial Secret Key** query on ID^*.

$\mathcal{C}$ then randomly picks $b \xleftarrow{\$} \{0, 1\}$ and gives $C^* = \mathtt{Enc}(\mathsf{mpk}, \mathsf{upk}^*, \mathsf{ID}^*, M_b)$ to $\mathcal{A}_I$.

Guess. $\mathcal{A}_I$ receives C^* and outputs a bit b'. If $b' = b$, $\mathcal{A}_I$ wins the game. The advantage of $\mathcal{A}_I$ in this game is $Adv_{\mathcal{A}_I}^{\text{IND-CPA}} = \Pr[b' = b] - \frac{1}{2}$.

Definition 2 (IND-CPA security against Type-II adversary). *A CLE scheme is IND-CPA secure against Type-II adversary if* $Adv_{\mathcal{A}_{II}}^{IND\text{-}CPA}$ *defined below is negligible.*

Setup. The challenger $\mathcal{C}$ executes $\mathtt{Setup}()$ and publishes param.

Master Key Generation. The challenger $\mathcal{C}$ runs the algorithm $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathtt{MKeyGen}(\mathsf{param})$, publishes mpk, and sends msk to $\mathcal{A}_{II}$.

Query Phase. $\mathcal{A}_{II}$ and $\mathcal{C}$ interact in the same way as in the experiment in Definition 1 except for the following differences. First, $\mathcal{C}$ sends psk to $\mathcal{A}_{II}$. Second, $\mathcal{A}_{II}$ can create new psk_i for ID_i by itself. Third, $\mathcal{A}_{II}$ can only make **Extract Full Private Key** queries in this game.

Challenge and **Guess.** These two phases are the same as in the experiment in Definition 1. The advantage of $\mathcal{A}_{II}$ in this game is $Adv_{\mathcal{A}_{II}}^{\text{IND-CPA}} = \Pr[b' = b] - \frac{1}{2}$.

The indistinguishability under chosen-ciphertext attacks (IND-CCA security) games for SP-CLE against *Strong Type-I* and *Strong Type-II* adversaries are similar to the experiments in Definitions 1 and 2 respectively, except that in Query Phase, the adversaries can make **Strong Decrypt** and **Weak SV Decrypt** queries on ciphertexts of its choice except C^*. The advantage of $\mathcal{A}_I$ and $\mathcal{A}_{II}$ in IND-CCA game are defined as $Adv_{\mathcal{A}_I}^{\text{IND-CCA}}$ and $Adv_{\mathcal{A}_{II}}^{\text{IND-CCA}}$ respectively. For replayable CCA (RCCA) security [11], decryption oracle returns $\mathtt{replay}$ if the decryption result is M_0 or M_1 after the challenge phase.

4 A Specific Construction of SP-CLE

4.1 Intuition

Instead of using an SPE generically to perform encryption, we rely on the pairings computed in the SPS verification for encryption or decryption. In our scheme, a receiver generates and sends his partial public key ppk to the KGC. The KGC creates a structure-preserving signature on the receiver identity together with the partial public key. The receiver then publishes *a part* of the signature together with his partial public key while keeping the remaining signature parts.

A general verification algorithm of an SPS consists of a series of pairing product equations of the form $\prod_{i=1}^{m} \prod_{j=1}^{n} e(G_i, H_j)^{a_{ij}} = 1_{\mathbb{G}_T}$, where $G_i \in \mathbb{G}$ for $i \in [1, m]$, $H_j \in \mathbb{H}$ for $j \in [1, n]$, and $a_{ij} \in \{-1, 0, 1\}$. The group elements G_i and H_i are from the verification key of SPS, the signature being verified, or the message. The exponents a_{ij} indicate whether they should be on the left or the right side of the equation (1 or -1), or should not appear at all (0).

We divide the set $\{(G_i, H_j)\}_{(i,j)}$ into two indices sets: $\mathbf{K}$ which contains the pairings used in encryption by the sender to construct a session key (or for hiding the plaintext); and $\overline{\mathbf{K}}$ which contains the rest of the pairing that are used in decryption to recover the session key. To encrypt a plaintext M, the pairings $e(G_i, H_i)$ for $(i, j) \in \mathbf{K}$ and some randomness $r_{ij} \xleftarrow{\$} \mathbb{Z}_p$ together form a session key as $\prod_{(i,j)\in\mathbf{K}} e(G_i, H_j)^{a_{ij}\cdot r_{ij}}$. The ciphertext also contains elements exponentiated with the randomness r_{ij} ($\{x, y, z\}$ in our concrete scheme below). The remaining pairings in set $\overline{\mathbf{K}}$ can be used in the decryption algorithm to pair up the ciphertext elements and the decryption key to recover the session key.

Whether a pairing should be put in the session key, included in the other ciphertext elements, or used in decryption privately as part of the decryption key, depends on whether the input of a pairing function is public or not.

We start with the basics. To make our exposition concrete, we consider the SPS scheme due to Abe *et al.* [4]. We chose to build our SP-CLE based on this SPS for its optimality. The verification key of the SPS scheme is the master public key which should be public. This contains $(g, h, U, \tilde{V}_1, \tilde{V}_2, W_1, W_2)$. The message vector signed by SPS contains a user identity and a (partial) user public key D_α. Both elements are public. The signature $(\tilde{R}, \tilde{S}, T)$ contributes to the only parts which can be private. Now, we classify the pairings in the SPS verification. A similar classification has also been done in the literature [32] for a different purpose (delegating computations of pairings).

(1) *Both* elements in a pairing are *public*: This type of pairing includes public key-public key pairs and message-public key pairs. The involved elements are available to the encryptor, so we use *all* of them in the *session key*. In our scheme, these include $e(W_1, h)$, $e(\mathsf{ID}, \tilde{V}_1)$, $e(D_\alpha, \tilde{V}_2)$, and $e(g, h)$, where D_α is a user-chosen public key. Our scheme also includes an additional term $e(D_\alpha, h)$ to ensure that only the user but not the KGC (who can recreate the SPS signature) can decrypt. Looking ahead, our scheme publishes $\tilde{R}$ from the signature, so $e(W_2, \tilde{R})$ and $e(U, \tilde{R})$ eventually belong to this type (see "both private" below).

(2) *One* of the elements in a pairing is *public*: This type of pairing includes public key-signature pairs and message-signature pairs. In our scheme, that is $e(g, \tilde{S})$. The public element can be used to embed randomness r in the ciphertext in the form of G_i^r or H_j^r. In our scheme, such elements include g (and $\tilde{R}$ below).

(3) *Both* elements in a pairing are *private*: The private elements (from the SPS signature) are part of the user private key. This type of pairing includes only signature-signature pairs. In our scheme, $e(T, \tilde{R})$ "originally" belongs to this type. As both of the elements are private, the encryptor has no way to know what is the SPS signature (i.e., user private key) obtained by the intended decryptor. We thus publish $\tilde{R}$ as part of the user public key (which is not allowed in the IBE setting). We remark that such treatment is not possible for IBE since the user public key in IBE should be purely derived from the identity instead of any random choice made by the KGC during user private key generation.

Such a choice (over T) is due to multiple reasons. Firstly, $\tilde{R}$ is created as a random term which by itself does not relate to the private signing key in any way. It is intuitively safer to publish it instead of T which is a term created from the private signing key on top of some public information like identity. Moreover, $\tilde{R}$ is the term which "glues up" two equations in the SPS verification. If the adversary chose to manipulate this term, it needs to deal with two equations. From the efficiency perspective, publishing $\tilde{R}$ minimizes the number of public-private pairings, which reduces the ciphertext size.

With $\tilde{R}$ published in our scheme, this makes $e(T, \tilde{R})$ becomes the type of "one being public". As discussed, the ciphertext in our scheme thus includes the term $\tilde{R}$ to embed the ciphertext randomness. Also, $e(W_2, \tilde{R})$ and $e(U, \tilde{R})$ in the pairing-product equations become the type of "both being public", and hence these pairing terms appear in the session key.

4.2 CPA-Secure SP-CLE Scheme

We construct our CPA-secure SP-CLE scheme called $\mathcal{CLE}_0$ based on an existing structure-preserving signature scheme of Abe *et al.* [4].

$\mathtt{Setup}(1^\lambda) \to \mathsf{param}$. Choose a bilinear group context $\mathcal{G} = (\mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, p, g, h)$, and output $\mathsf{param} = \mathcal{G}$.

$\mathtt{MKeyGen}(\mathsf{param}) \to (\mathsf{mpk}, \mathsf{msk})$. The KGC randomly picks $u, v_1, v_2, w_1, w_2 \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ where $u \neq -w_2$, and computes $U = g^u$, $\tilde{V}_1 = h^{v_1}$, $\tilde{V}_2 = h^{v_2}$, $W_1 = g^{w_1}$, and $W_2 = g^{w_2}$. The master key pair is

$$\mathsf{mpk} = (U, \tilde{V}_1, \tilde{V}_2, W_1, W_2), \qquad \mathsf{msk} = (u, v_1, v_2, w_1, w_2).$$

This key pair is just the one for the SPS scheme by Abe *et al.* [4] with the message space of $\mathbb{G}^2 \times \mathbb{H}$. Specifically, U is for the $\mathbb{H}$ part of the message space, and $(\tilde{V}_1, \tilde{V}_2)$ is for $\mathbb{G}^2$. Note that $e(g, h)$ and $e(W_1, h)$ can be pre-computed, especially when W_1 is never used as is except in $e(W_1, h)$.

$\texttt{SetUserSec}(\mathsf{mpk}) \rightarrow (\mathsf{ppk}, \mathsf{uk})$. A user randomly picks $\alpha \xleftarrow{\$} \mathbb{Z}_p$, computes $D_\alpha = g^\alpha$ and $\tilde{D}_\alpha = h^\alpha$, and sets $\mathsf{ppk} = D_\alpha$ and $\mathsf{uk} = \tilde{D}_\alpha$.

$\texttt{Issue}(\mathsf{msk}, \mathsf{mpk}, \mathsf{ID}, \mathsf{ppk}) \rightarrow \mathsf{psk}$. For $\mathsf{ID} \in \mathbb{G}$ and $\mathsf{ppk} = D_\alpha \in \mathbb{G}$, the KGC randomly chooses $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes

$$\tilde{R} = h^r, \qquad \tilde{S} = h^{w_1 - r \cdot w_2} \cdot \tilde{R}^{-u}, \qquad T = (g \cdot \mathsf{ID}^{-v_1} \cdot D_\alpha^{-v_2})^{\frac{1}{r}},$$

Output $\mathsf{psk} = (\tilde{R}, \tilde{S}, T)$ as the partial secret key.

We remark that $(\tilde{R}, \tilde{S}, T)$ forms a signature on $(\mathsf{ID}, D_\alpha, \tilde{R}) \in \mathbb{G}^2 \times \mathbb{H}$ for the SPS scheme by Abe *et al.* [4] which can be verified with the equations below:

$$e(W_2, \tilde{R})e(g, \tilde{S})e(U, \tilde{R}) = e(W_1, h), \qquad e(T, \tilde{R})e(\mathsf{ID}, \tilde{V}_1)e(D_\alpha, \tilde{V}_2) = e(g, h).$$

Note that the first equation can be simplified to $e(W_2 \cdot U, \tilde{R})e(g, \tilde{S}) = e(W_1, h)$.

Different from the underlying signature scheme, we expect the signature to sign on an element $\tilde{R}$ of itself. This remains secure in the generic group model.

$\texttt{UKeyGen}(\mathsf{mpk}, \mathsf{ppk}, \mathsf{psk}, \mathsf{uk}) \rightarrow (\mathsf{upk}, \mathsf{usk})$. A user parses psk as $(\tilde{R}, \tilde{S}, T)$ and set the key pair as

$$\mathsf{upk} = (D_\alpha, \tilde{R}), \qquad \mathsf{usk} = (\tilde{D}_\alpha, \tilde{S}, T) \qquad \text{(recall: } \mathsf{ppk} = D_\alpha \text{ and } \mathsf{uk} = \tilde{D}_\alpha).$$

As $\tilde{R}$ is a part of upk, it can be replaced by an adversary. Our scheme thus also requires the KGC to "implicitly certify" $\tilde{R}$ during partial secret key generation.

$\texttt{Enc}(\mathsf{mpk}, \mathsf{upk}, \mathsf{ID}, M) \rightarrow C$. To encrypt $M \in \mathbb{G}_T$, the sender randomly picks $x, y, z \xleftarrow{\$} \mathbb{Z}_p$, and computes

$$K = \{e(W_2, \tilde{R})e(U, \tilde{R})/e(W_1, h)\}^x \{e(\mathsf{ID}, \tilde{V}_1)e(D_\alpha, \tilde{V}_2)/e(g, h)\}^y / e(D_\alpha, h)^z,$$
$$C_0 = M \cdot K, \quad C_g = g^x, \quad C_R = \tilde{R}^y, \quad C_z = g^z.$$

Output the ciphertext $C = (C_0, C_g, C_R, C_z)$.

(Note that $K = \{e(W_2 U, \tilde{R})/e(W_1, h)\}^x \{e(\mathsf{ID}, \tilde{V}_1)/e(g, h)\}^y e(D_\alpha, \tilde{V}_2^y / h^z)$.)

$\texttt{Dec}(\mathsf{mpk}, \mathsf{upk}, \mathsf{usk}, C) \rightarrow M/\bot$. Parse C as (C_0, C_g, C_R, C_z). Output

$$M = C_0 \cdot e(C_g, \tilde{S})e(T, C_R)e(C_z, \tilde{D}_\alpha).$$

Analysis. **Correctness.** Recall that $D_\alpha = g^\alpha$, $\tilde{D}_\alpha = h^\alpha$, $C_0 = M \cdot K$, and

$$K = e(W_2, \tilde{R})^x e(U, \tilde{R})^x e(W_1, h)^{-x} \cdot e(\mathsf{ID}, \tilde{V}_1)^y e(D_\alpha, \tilde{V}_2)^y e(g, h)^{-y} \cdot e(D_\alpha, h)^{-z}.$$

Hence, the decryption algorithm proceeds as below.

$$\begin{aligned}
&C_0 \cdot e(C_g, \tilde{S}) e(T, C_R) e(C_z, \tilde{D}_\alpha) \\
=&M \cdot K \cdot e(C_g, \tilde{S}) e(T, C_R) e(C_z, \tilde{D}_\alpha) \\
=&M \cdot e(W_2, \tilde{R})^x e(U, \tilde{R})^x e(W_1, h)^{-x} e(\mathsf{ID}, \tilde{V}_1)^y e(D_\alpha, \tilde{V}_2)^y e(g, h)^{-y} e(D_\alpha, h)^{-z} \\
&e(C_g, \tilde{S}) e(T, C_R) e(C_z, \tilde{D}_\alpha) \\
=&M \cdot e(W_2, \tilde{R})^x e(U, \tilde{R})^x e(W_1, h)^{-x} e(C_g, \tilde{S}) \\
&e(\mathsf{ID}, \tilde{V}_1)^y e(D_\alpha, \tilde{V}_2)^y e(g, h)^{-y} e(T, C_R) \cdot e(D_\alpha, h)^{-z} e(C_z, \tilde{D}_\alpha) \\
=&M \cdot e(W_2, \tilde{R})^x e(U, \tilde{R})^x e(W_1, h)^{-x} e(g, \tilde{S})^x \\
&e(\mathsf{ID}, \tilde{V}_1)^y e(D_\alpha, \tilde{V}_2)^y e(g, h)^{-y} e(T, \tilde{R})^y \cdot e(g^\alpha, h)^{-z} e(g^z, h^\alpha) \\
=&M \cdot (e(W_2, \tilde{R}) e(g, \tilde{S}) e(U, \tilde{R}) e(W_1, h)^{-1})^x \\
&(e(T, \tilde{R}) e(\mathsf{ID}, \tilde{V}_1) e(D_\alpha, \tilde{V}_2) e(g, h)^{-1})^y = M.
\end{aligned}$$

The second last equality holds because $(\tilde{R}, \tilde{S}, T)$ is a signature which satisfies the verification equations mentioned when we describe `Issue`().

Efficiency. We first start with some basic observations of our scheme. The user private key consists of 3 elements in base groups. The ciphertext consists of 3 group elements in base groups and 1 group element in the target group. The decryption algorithm needs 3 pairings and 4 multiplications in the target group.

Comparison with the Generic Approach. It is mandatory to compare the performance of our proposed scheme with the folklore approach of building a CLE scheme "with certificate" [12]. Specifically, one can build a CLE scheme from any SPS and SPE schemes in the following way. A user publishes an SPE public key with an SPS signature on it as his public key. An encryptor encrypts to the user using the SPE public key only if the SPS signature is verified successfully.

Instantiating this idea with the SPS due to Abe *et al.* [4] used in our concrete construction, we can see that the user public key will then consists of at least 3 elements from the SPS (and at least 1 element from the SPE public key as the CLE partial user public key). In contrast, for our concrete construction, the user public key consists of only 2 elements in base groups, which is much shorter.

The explicit certificate verification step in the folklore approach using the same SPS scheme as ours will require 3 multiplications in the target group and 5 pairings. While the complexity of the actual encryption steps depends on which SPE scheme is used to instantiate this idea, the number of pairings involved is already larger than what our proposed scheme requires. Our encryption algorithm takes 5 exponentiations and 2 multiplications in base groups, 2 exponentiations and 4 multiplications in the target group, and 3 pairing computations.

Theorem 1. *$\mathcal{CLE}_0$ is CPA-secure against Type-I and Type-II adversaries in the generic group model (without any isomorphism between the two base groups).*

To prove that $\mathcal{CLE}_0$ is CPA-secure against Type-I and Type-II adversaries, we replace the challenge ciphertext component C_0^* with a random element in $\mathbb{G}_T$

and show that the adversaries cannot distinguish this simulation with the real scheme in the generic group model. The detailed proof is in the full version.

4.3 A Variant CLE Scheme for $M \in \mathbb{G}$

This part proposes an SP-CLE scheme $\mathcal{CLE}_1$ encrypting $M \in \mathbb{G}$ building on top of $\mathcal{CLE}_0$. Based on the technique of encrypting group elements in the partially structure-preserving IBE scheme [25], we present a generic way to transform a scheme encrypting plaintexts in $\mathbb{G}_T$ to a scheme encrypting plaintexts in $\mathbb{G}$ or $\mathbb{H}$.

$\texttt{Setup}(1^\lambda) \rightarrow \mathsf{param}$. The KGC runs $\mathsf{param}_0 \leftarrow \mathcal{CLE}_0.\texttt{Setup}(1^\lambda)$, picks $G_i \xleftarrow{\$} \mathbb{G}$ for $i \in [1, l]$ where l is suitably large[1], and outputs $\mathsf{param} = (\mathsf{param}_0, \{G_i\}_{i=1}^{l})$.

$\texttt{MKeyGen}() \rightarrow (\mathsf{mpk}, \mathsf{msk})$. The KGC runs $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathcal{CLE}_0.\texttt{MKeyGen}(\mathsf{param}_0)$ and outputs the master key pair $\mathsf{mpk} = (\mathsf{mpk}_0, \{G_i\}_{i=1}^{l})$, $\mathsf{msk} = \mathsf{msk}_0$.

$\texttt{SetUserSec}(\mathsf{mpk}) \rightarrow (\mathsf{ppk}, \mathsf{uk})$. A user runs $(\mathsf{ppk}, \mathsf{uk}) \leftarrow \mathcal{CLE}_0.\texttt{SetUserSec}(\mathsf{mpk}_0)$, and sets $\mathsf{ppk}, \mathsf{uk}$ as its partial public key and the user secret value respectively.

$\texttt{Issue}(\mathsf{msk}, \mathsf{mpk}, \mathsf{ID}, \mathsf{ppk}) \rightarrow \mathsf{psk}$. For a user $\mathsf{ID} \in \mathbb{H}$, the KGC runs $\mathsf{psk}_0 \leftarrow \mathcal{CLE}_0.\texttt{Issue}(\mathsf{msk}_0, \mathsf{mpk}_0, \mathsf{ID}, \mathsf{ppk})$ and outputs the partial secret key $\mathsf{psk} = \mathsf{psk}_0$.

$\texttt{UKeyGen}(\mathsf{mpk}, \mathsf{ppk}, \mathsf{psk}, \mathsf{uk}) \rightarrow (\mathsf{upk}, \mathsf{usk})$. The user computes its own user public-private key pair as $(\mathsf{upk}, \mathsf{usk}) \leftarrow \mathcal{CLE}_0.\texttt{UKeyGen}(\mathsf{mpk}_0, \mathsf{psk}_0, \mathsf{ppk}, \mathsf{uk})$.

$\texttt{Enc}(\mathsf{mpk}, \mathsf{upk}, \mathsf{ID}, M) \rightarrow C$. To encrypt $M \in \mathbb{G}$, randomly choose $\tau_k \in \{0, 1\}$ for $k = 1, 2, \cdots, l$, and compute

$$C_0 = M \cdot \prod_{j=1}^{l} G_j^{\tau_j}, C_{k,M} \leftarrow \mathcal{CLE}_0.\texttt{Enc}(\mathsf{mpk}_0, \mathsf{upk}, \mathsf{ID}, e(G_k, h)^{\tau_k})\ \forall k \in \{1, 2, \cdots, l\}.$$

Output $C = (C_0, \{C_{k,M}\}_{k=1}^{l})$ as the ciphertext (where $\{C_{k,M}\}$ are still in $\mathbb{G}_T$).

$\texttt{Dec}(\mathsf{mpk}, \mathsf{upk}, \mathsf{usk}, C) \rightarrow M/\perp$. Parse C as $(C_0, \{C_{k,M}\}_{k=1}^{l})$. For $k = 1, 2, \cdots, l$, compute $M_k = \mathcal{CLE}_0.\texttt{Dec}(\mathsf{mpk}_0, \mathsf{upk}, \mathsf{usk}, C_{k,M})$ and find τ_k such that $M_k = e(G_k, h)^{\tau_k}$. Output $M = \frac{C_0}{\prod_{k=1}^{l} G_k^{\tau_k}}$ as the plaintext.

The scheme $\mathcal{CLE}_1$ also supports plaintexts from $\mathbb{H}$. If we choose $\tilde{H}_k \in \mathbb{H}$ for integer $k \in [1, l]$ as part of the master public key, and encrypt the plaintext as $M \cdot \prod_{k=1}^{l} \tilde{H}_k^{\tau_k}$, we can then encrypt plaintext in $\mathbb{H}$.

Correctness. The correctness of $\mathcal{CLE}_1$ follows from the correctness of $\mathcal{CLE}_0$, which ensures that M_k can be calculated correctly. Thus, there is at most one series $\{\tau_k\}_{k=1}^{l}$ such that $M_k = e(G_k, h)^{\tau_k}$ for all $k \in [1, l]$, and this series can cancel the term $\prod_{k=1}^{l} G_k^{\tau_k}$ in C_0 to obtain the plaintext M. More details can be seen from the correctness analysis in our CCA-secure extension presented below, which also encrypts messages in the base group ($\mathbb{H}$).

[1] In the partially structure-preserving IBE scheme [25], this represents the bit-length of the identity. In our scheme, ID is a group element, so l belongs to $\mathsf{poly}(\lambda)$.

Theorem 2. *The SP-CLE scheme $\mathcal{CLE}_1$ is IND-CPA secure if $\mathcal{CLE}_0$ is IND-CPA secure.*

The proof is deferred to the full version.

4.4 RCCA-Secure Extension

Now we propose an RCCA-secure SP-CLE scheme $\mathcal{CLE}_2$ with message space $\mathbb{H}$, which uses a one-time SPS scheme $\mathcal{OTS}$ and a simulation-sound NIZK proof system $\mathcal{GS}$ as building blocks, following the idea of transforming CPA-secure IBE to CCA-secure PKE [9]. We use the SPS scheme proposed by Abe *et al.* [2] as $\mathcal{OTS}$ (which is also used in an CCA-secure SPE scheme by Libert *et al.* [27]).

Our RCCA-secure SP-CLE is derived from $\mathcal{CLE}_1$. Intuitively, the encryptor generates an $\mathcal{OTS}$ key pair $(\mathsf{ovk}, \mathsf{osk})$, binds ovk with the session key, provides extra elements computed from osk (which can be simulated without osk with the "trapdoor" in param), and proves everything is faithfully constructed using osk. We add a Groth-Sahai proof of the validity of the ciphertext embedding the plaintext as a witness. When simulating **Strong Decrypt** oracle, the challenger can extract the plaintext even for an identity with replaced user public key.

$\mathtt{Setup}(1^\lambda) \to \mathsf{param}$. Run the two algorithms $\mathsf{param}_1 \leftarrow \mathcal{CLE}_1.\mathtt{Setup}(1^\lambda)$ and $\mathsf{param}_{OTS} \leftarrow \mathcal{OTS}.\mathtt{Setup}(1^\lambda, 1)$, and set up $\mathcal{GS}$ to generate a common reference string crs. Randomly choose $u_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [1,4]$ to compute $U_i = g^{u_i}$, $\tilde{H}_i = h^{u_i}$, and output the public parameter $\mathsf{param} = (\mathsf{param}_1, \mathsf{param}_{OTS}, \mathsf{crs}, \{U_i, \tilde{H}_i\}_{i=1}^4)$.

$\mathtt{MKeyGen}(\mathsf{param}) \to (\mathsf{mpk}, \mathsf{msk})$. The KGC runs the algorithm $(\mathsf{mpk}_1,\ \mathsf{msk}_1) \leftarrow \mathcal{CLE}_1.\mathtt{MKeyGen}(\mathsf{param}_1)$, and outputs the master public-private key pair as $\mathsf{mpk} = (\mathsf{mpk}_1, \{U_i, \tilde{H}_i\}_{i=1}^4)$, $\mathsf{msk} = \mathsf{msk}_1$. The one-time public key ovk for $\mathcal{OTS}$ of our choice [2] consists of 4 group elements in $\mathbb{H}$. The elements $\{U_i, \tilde{H}_i\}_{i=1}^4$ are for binding ovk with a ciphertext. Generally, i can be in the range $[1,k]$ where k is the number of elements contained in ovk of the one-time SPS scheme.

$\mathtt{SetUserSec}(\mathsf{mpk}) \to (\mathsf{ppk}, \mathsf{uk})$. A user runs $(\mathsf{ppk}, \mathsf{uk}) \leftarrow \mathcal{CLE}_1.\mathtt{SetUserSec}$ (mpk_1), and sets $(\mathsf{ppk}, \mathsf{uk})$ as its partial public key and the user secret value respectively.

$\mathtt{Issue}(\mathsf{msk}, \mathsf{mpk}, \mathsf{ID}, \mathsf{ppk}) \to \mathsf{psk}$. For a user with identity $\mathsf{ID} \in \mathbb{H}$, the KGC outputs the partial secret key $\mathsf{psk} \leftarrow \mathcal{CLE}_1.\mathtt{Issue}(\mathsf{msk}_1, \mathsf{mpk}_1, \mathsf{ID}, \mathsf{ppk})$.

$\mathtt{UKeyGen}(\mathsf{mpk}, \mathsf{psk}, \mathsf{ppk}, \mathsf{uk}) \to (\mathsf{upk}, \mathsf{usk})$. The user computes its own user public-private key pair as $(\mathsf{upk}, \mathsf{usk}) \leftarrow \mathcal{CLE}_1.\mathtt{UKeyGen}(\mathsf{mpk}, \mathsf{psk}, \mathsf{ppk}, \mathsf{uk})$.

$\mathtt{Enc}(\mathsf{mpk}, \mathsf{upk}, \mathsf{ID}, M) \to C$. To encrypt $M \in \mathbb{G}$, the sender randomly picks $\tau_k \xleftarrow{\$} \{0,1\}$ and $x_k, y_k, z_k \xleftarrow{\$} \mathbb{Z}_p$ for $k \in [1,l]$. The set $\{x_k, y_k, z_k, \tau_k\}$ will be used as the internal randomness for $\mathcal{CLE}_1.\mathtt{Enc}()$. The sender also runs $(\mathsf{ovk}, \mathsf{osk}) \leftarrow \mathcal{OTS}.\mathtt{KeyGen}(\mathsf{param}_{OTS})$ of Abe *et al.*'s one-time SPS scheme [2] which the exponent $\{a_i\}$ for $i \in [1,4]$ such that $\mathsf{ovk} = (h^{a_1}, h^{a_2}, h^{a_3}, h^{a_4})$ are available. For the ease of presentation, we use $(\tilde{A}_1, \tilde{A}_2, \tilde{A}_3, \tilde{A}_4)$ to represent ovk.

Finally, the sender computes

$$\begin{aligned}
&(C_0, \{C_{k,M}\}_{k=1}^{l}) \leftarrow \mathcal{CLE}_1.\mathtt{Enc}(\mathsf{mpk}_1, \mathsf{upk}, \mathsf{ID}, M; \{x_k, y_k, z_k, \tau_k\}),\\
&(C'_{k,0}, C_{k,g}, C_{k,R}, C_{k,z}) \leftarrow C_{k,M},\\
&C_{k,0} = C'_{k,0} \cdot \prod_{i=1}^{4} e(U_i, \tilde{A}_i)^{-x_k} \text{ for } k \in [1, l],\\
&C_{a,i} = \tilde{H}_i^{a_i} \text{ for } i \in [1, 4],\\
&\pi = PoK\{(M, \{x_k, y_k, z_k, \tau_k\}_{k=1}^{l}, \{a_i\}_{i=1}^{4}) :\\
&\qquad (C_0, \{(C'_{k,0}, C_{k,g}, C_{k,R}, C_{k,z})\}_{k=1}^{l})\\
&\qquad \leftarrow \mathcal{CLE}_1.\mathtt{Enc}(\mathsf{mpk}_1, \mathsf{upk}, \mathsf{ID}, M; \{x_k, y_k, z_k, \tau_k\}_{k=1}^{l})\\
&\qquad \wedge_{k=1}^{l} C_0 = M \cdot \prod_{j=1}^{l} G_j^{\tau_j} \wedge_{i=1}^{4} C_{a,i} = \tilde{H}_i^{a_i}\\
&\qquad \wedge_{k=1}^{l} C_{k,0} = C'_{k,0} \cdot \prod_{i=1}^{4} e(U_i, \tilde{A}_i)^{-x_k}\},\\
&\sigma \leftarrow \mathcal{OTS}.\mathtt{Sign}(\mathsf{osk}, C_0).
\end{aligned}$$

Output $(C_0,\ \{\tilde{A}_i, C_{a,i}\}_{i=1}^{4},\ \{C_{k,0},\ C_{k,g},\ C_{k,R},\ C_{k,z}\}_{k=1}^{l},\ \pi,\ \sigma)$ as the ciphertext.

$\mathtt{Dec}(\mathsf{mpk}, \mathsf{upk}, \mathsf{usk}, C) \rightarrow M/\perp$. The decryptor first performs the following checks.

1. Parse the ciphertext C as specified in the output of the algorithm $\mathtt{Enc}()$.
2. Verify the equations $e(g, C_{a,i}) = e(U_i, \tilde{A}_i)$ for $i \in [1, 4]$.
3. Verify the signature σ using $\mathcal{OTS}.\mathtt{Verify}((\tilde{A}_1, \tilde{A}_2, \tilde{A}_3, \tilde{A}_4), C_0, \sigma)$.
4. Verify the proof π using the $\mathcal{GS}.\mathtt{Verify}()$ algorithm.

If any one of the four equations does not hold, or either σ or π does not pass the verification, output $\perp$. Otherwise, for $k \in [1, l]$, compute

$$M_k = C_{k,0} \cdot e(C_{k,g}, \tilde{S} \cdot \prod_{i=1}^{4} C_{a,i}) e(T, C_{k,R}) e(C_{k,z}, \tilde{D}_\alpha).$$

Find τ_k such that $M_k = e(G_k, h)^{\tau_k}$. Finally, output $M = \frac{C_0}{\prod_{i=1}^{l} G_i^{\tau_k}}$.

Correctness. For $k \in [1, l]$,

$$\begin{aligned}
&C_{k,0} \cdot e(C_{k,g}, \tilde{S} \cdot \prod_{i=1}^{4} C_{a,i}) e(T, C_{k,R}) e(C_{k,z}, \tilde{D}_\alpha)\\
= &M_k \cdot e(W_2, \tilde{R})^{x_k} e(U, \tilde{R})^{x_k} \cdot \prod_{i=1}^{4} e(U_i, \tilde{A}_i)^{-x_k} \cdot e(W_1, h)^{-x_k}\\
&\cdot e(\mathsf{ID}, \tilde{V}_1)^{y_k} e(D_\alpha, \tilde{V}_2)^{y_k} e(g, h)^{-y_k} e(D_\alpha, h)^{-z_k}\\
&\cdot e(C_{k,g}, \tilde{S} \cdot \prod_{i=1}^{4} C_{a,i}) e(T, C_{k,R}) e(C_{k,z}, \tilde{D}_\alpha)
\end{aligned}$$

$$= M_k \cdot e(W_2, \tilde{R})^{x_k} e(U, \tilde{R})^{x_k} \cdot \prod_{i=1}^{4} e(U_i, \tilde{A}_i)^{-x_k} \cdot e(W_1, h)^{-x_k} \cdot e(C_{k,g}, \tilde{S} \cdot \prod_{i=1}^{4} C_{a,i})$$
$$\cdot e(\mathsf{ID}, \tilde{V}_1)^{y_k} e(D_\alpha, \tilde{V}_2)^{y_k} e(g, h)^{-y_k} \cdot e(T, C_{k,R}) \cdot e(D_\alpha, h)^{-z_k} \cdot e(C_{k,z}, \tilde{D}_\alpha)$$
$$= M_k \cdot (e(W_2, \tilde{R}) e(U, \tilde{R}) \cdot \prod_{i=1}^{4} e(U_i, \tilde{A}_i)^{-1} \cdot e(W_1, h)^{-1} \cdot e(g, \tilde{S} \cdot \prod_{i=1}^{4} C_{a,i}))^{x_k}$$
$$\cdot (e(\mathsf{ID}, \tilde{V}_1) e(D_\alpha, \tilde{V}_2) e(g, h)^{-1} \cdot e(T, \tilde{R}))^{y_k} \cdot e(g^\alpha, h)^{-z_k} \cdot e(g^z, h^\alpha) = M_k.$$

With correct M_k, τ_k such that $M_k = e(G_k, h)^{\tau_k}$ can be correctly recovered. With all M_k for $k \in [1, l]$, $M = \frac{C_0}{\prod_{i=1}^{l} G_i^{\tau_k}}$ can be correctly recovered as in Sect. 4.3.

Theorem 3. *The SP-CLE scheme $\mathcal{CLE}_2$ is RCCA-secure against* Strong Type-I *and* Strong Type-II *adversaries if $\mathcal{CLE}_1$ is CPA-secure against Type-I and Type-II adversaries.*

The proof is deferred to the full version.

Remark. A fully structure-preserving CLE scheme would be an overkill for our application as it does not need to hide the ciphertext and prove about its validity. Also, our application will apply yet another signature on top of the CLE ciphertext (with other parts) such that any rerandomization of the CLE ciphertext will invalidate the signature, so $\mathcal{CLE}_2$ only aimed for RCCA-security.

Nevertheless, Appendix A outlines how to use the trick of Libert and Joye [25] for converting $\mathbb{G}_T$ values into base group elements in the ciphertext of our $\mathcal{CLE}_1$.

5 Group Signatures with Certified Limited Opening

We use our SP-CLE (in Sect. 4) as a building block to construct an example application, a group signature scheme with certified limited (CL) opening, a generalization of message-dependent opening [30]. Due to the page limit, we present the formal definitions in the full version.

Group signature is a privacy-oriented signature scheme where the verifier can be convinced that a given signature is signed by a group member, but not exactly whom. Since perfect anonymity may be abused, group signatures come with an opening mechanism such that the group manager, or in general, an opening authority (OA), can use a secret key to reveal the true signer of a signature.

When there is purported abuse, we want to identify the signer of the suspicious signatures. In traditional group signatures, it means *all* signatures must be opened, which is undesirable for honest users. The notion of traceable signatures (TS) [1,23] extends that of the group signatures to mitigate this problem. In TS, when a group member is classified as a misbehaving one. A user-specific tracing trapdoor can be generated (by the group manager or the OA). Every one with this user-specific trapdoor can check if a signature is actually signed by the misbehaving user, or trace [13] the signatures generated by the misbehaving user. TS can be regarded as a group signature scheme with signer-dependent opening. Subsequently, Sakai *et al.* [30] proposed the notion of group signature with message-dependent opening (GS-MDO). In GS-MDO, apart from the OA, there is another entity called the admitter. The admitter can generate a message-dependent opening key. The real signer of a group signature signing on a given message can be revealed only when both the master opening key (of the OA) and the message-dependent opening key (provided by the admitter) are used.

Difficulty in Construction. GS-MDO schemes are often constructed by IBE since GS-MDO implies its existence (or precisely, identity-based key encapsulation) [30]. Existing schemes not relying on the pairing-based Groth-Sahai proof are either not that efficient [26] or is proven secure in the random oracle model [28]; however, typical pairing-based IBE schemes encrypt messages in the target group, which are not compatible with Groth-Sahai proof that a correct message (the signer identity in the case of GS-MDO) has been encrypted.

Consequently, the original work of Sakai *et al.* [30] proposed to use k-resilient IBE to construct GS-MDO which remains secure only when adversary obtains no more than a predefined bound of k message-dependent opening keys. Later, Ohara *et al.* [28] proposed a GS-MDO scheme with unbounded MDO in the random oracle model. A subsequent work of Libert and Joye [25] describes an unbounded GS-MDO scheme in the standard model by proposing an IBE scheme which encrypts messages in the base group. This IBE scheme is partially structure preserving in the sense that the identity is still a bit-string instead of a group element. In an IBE-based GS-MDO scheme, the identity used in IBE is the same as the message to be signed. So this scheme [25] is not structure-preserving and cannot sign on group elements. Potential higher applications of GS-MDO thus cannot hide yet prove about the message with another Groth-Sahai proof.

Certified Limited Opening. We consider an alternative way of limiting the opening power which we call certified limited (CL) opening. CL opening features an entity called a *master certifier*, who certifies *openers* case by case depending on the context. For example, consider the application of group signatures for signing on votes in electronic voting. The government can be the master certifier, and the openers can be those overseeing different districts/counties/provinces/states. When issuing a group signature, the group member can designate an *opener* during the signing process. The opener who is the designated one for a group signature can open it (i.e., revoke the anonymity of the signature). Neither the certifier nor any non-designated openers can perform opening.

CL opening is a variant of MDO which removes the reliance of a single opening authority and minimizes the disturbance of honest users. Moreover, it decouples the criteria of opening from the message being signed. In many applications, the need for opening may not be originated from the message itself. We can assign the openers depending on the applications. Consider the e-voting scenario again, where the voting software in one of the voting booths could be compromised. We can set the opener to be the authorities overseeing different booths. If some anomaly happen with a particular booth, say, the candidate is set to be an adversarially-chosen set under the hood, independent of what is the vote cast by the voters; only the signatures in the concerned booth will be opened, and only the affected voters will be asked to cast a correct vote again.

CL opening also simplifies the opening process. The existing MDO functionality [25,30] requires the master opening key and the message-dependent key as inputs. That means the two parties holding the corresponding keys must cooperate in an honest manner. In our formulation, the master certifier and the opening authority interact once such that latter will get the opening key of limited power, instead of performing joint decryption in every opening. Dealing with a single key also allows an easier zero-knowledge proof for the opening correctness.

5.1 Our Group Signature Scheme with Certified Limited Opening

We build our group signature scheme with CL opening using SP-CLE. In a nutshell, the signing algorithm uses SP-CLE to encrypt the identity of the signer with respect to a SP-CLE user. In this way, we can realize new privacy-enhancing features easily thanks to the preserved structures. In particular, since the identity and the user public key in our SP-CLE scheme are both group elements, one can include an additional proof about them to preserve the opener privacy. For example, it can hide who is the designated opener among a list of possibilities.

Due to our formulation of the underlying SP-CLE scheme, our resulting group signature scheme with CL opening can be considered as weaker than group signatures with MDO since the message in the latter does not require prior "certification" from any party. However, in case the message domain is small, one can obtain MDO from CL opening by assigning an opener for each possible message. Also, as argued above, we decouple the message to be signed from the context of the opening. More importantly, from the technical perspective, since SP-IBE does not exist, it is unclear how to "upgrade" the existing GS-MDO schemes such that we can sign on a group element, while retaining the MDO functionality. On the other hand, our group signature scheme with CL opening is partially structure-preserving, in the sense that it can sign on group element as a message (and the public-key and the identity of the opener are also group elements, due to our SP-CLE). It can then sign on an encryption of vote (for privacy) when the resulting ciphertext consists of only group elements, and further allow a zero-knowledge proof of the message being encrypted and signed. For example, the zero-knowledge proof can be proving that the vote is a valid choice among the possible candidates. With the group structure preserved, the encrypted votes can also be homomorphically-processed (when the underlying encryption is homomorphic) such that only the aggregate results will be revealed.

Finally, as a generic construction, future constructions of SP-CLE in the original formulation can be directly plugged into our proposed design.

5.2 Construction

Design Overview. We follow the two-level signature construction [8] and use two SPS instances and one SP-CLE instance. The group manager generates an SPS signature $\mathsf{cert}_{\mathsf{ID}}$ on an identity ID and a verification key $\mathsf{vk}_{\mathsf{ID}}$ for an SPS scheme as part of the user private key for ID. The user with identity ID generates another SPS signature σ' on a message M, then proves the relation of $(\mathsf{ID}, \mathsf{vk}_{\mathsf{ID}}, \mathsf{cert}_{\mathsf{ID}})$ and that of (M, σ') without revealing $\mathsf{ID}, \mathsf{vk}_{\mathsf{ID}}, \mathsf{certID}$, nor σ'.

To implement the certified limited opening feature using SP-CLE, the KGC (as the master certifier) interacts with an SP-CLE user (as an opener). After they interact in the SP-CLE key-issuing process, the opener obtains a public-private key pair. Suppose the identity of the opener is E, the user public key pk_E will be published, and the user private key osk_E will be kept secret. The signer uses pk_E to encrypt ID, then generates a proof showing that this ciphertext is well-formed. All the proofs and this ciphertext are output as the group signature. The party holding osk_E can decrypt the ciphertext to obtain ID.

Syntax. Our definition extends the one by Sakai *et al.* [30]. We replace the input of the `TrapGen` algorithm from a message M with an identifier E and an opener public key, and only require the output of `TrapGen` but not the "master" opening key in the `Open` algorithm. We also split the key generation into `Setup`, `MKeyGen`, and `Issue`. A detailed definition can be found in the full version.

Our Construction. We use an our CLE scheme for $M \in \mathbb{G}$ $\mathcal{CLE}$, two SPS schemes $\mathcal{SPS}_G$ and $\mathcal{SPS}$, and a GS-proof system $\mathcal{GS}$ as the building blocks to construct a structure-preserving group signature with certified limited opening. As Groth-Sahai proof is rerandomizable, we use a structure-preserving one-time signature $\mathcal{OTS}$ to enforce CCA-anonymity.

This scheme also achieves the "hidden identity" features as in hidden identity-based signatures [17,24] since its opening mechanism can directly recover the signer identity without relying on the existence of any membership database.

$\mathtt{Setup}(1^\lambda) \rightarrow \mathsf{param}$. Choose a Type III bilinear group $\mathcal{G} = (\mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, p, g, h)$ which is suitable for $\mathcal{CLE}$, $\mathcal{SPS}_G$, and $\mathcal{SPS}$. Generate the common reference string crs for $\mathcal{GS}$. Output $\mathsf{param} = (\mathcal{G}, \mathsf{crs})$.

$\mathtt{MKeyGen}() \rightarrow (\mathsf{mpk}, \mathsf{msk})$. Generate the key-pair for the underlying structure-preserving primitives as follows.

1. $(\mathsf{vk}_G, \mathsf{sk}_G) \leftarrow \mathcal{SPS}_G.\mathtt{KeyGen}()$.
2. $(\mathsf{mpk}_{\mathcal{CLE}}, \mathsf{msk}_{\mathcal{CLE}}) \leftarrow \mathcal{CLE}.\mathtt{MKeyGen}()$.

Output the master public-private key pair $\mathsf{mpk} = (\mathsf{vk}_G, \mathsf{mpk}_{\mathcal{CLE}})$, $\mathsf{msk} = \mathsf{sk}_G$ to the KGC, and output the master opening key $\mathsf{ok} = \mathsf{msk}_{\mathcal{CLE}}$ to the master certifier.

$\mathtt{Issue}(\mathsf{msk}, \mathsf{ID}) \rightarrow \mathsf{usk}_{\mathsf{ID}}$. A user with identity ID and the KGC interactively compute a certificate as part of the user secret key for the user.

1. The user runs $(\mathsf{vk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}) \leftarrow \mathcal{SPS}.\mathtt{KeyGen}()$, sends $(\mathsf{ID}, \mathsf{vk}_{\mathsf{ID}})$ to the KGC.
2. The KGC runs $\mathsf{cert}_{\mathsf{ID}} \leftarrow \mathcal{SPS}_G.\mathtt{Sign}(\mathsf{sk}_G, (\mathsf{ID}, \mathsf{vk}_{\mathsf{ID}}))$, sends $\mathsf{cert}_{\mathsf{ID}}$ to the user.

The user sets $\mathsf{usk}_{\mathsf{ID}} = (\mathsf{sk}_{\mathsf{ID}}, \mathsf{vk}_{\mathsf{ID}}, \mathsf{cert}_{\mathsf{ID}})$ as user private key.

$\mathtt{TrapGen}(\mathsf{mpk}, \mathsf{ok}, E) \rightarrow (\mathsf{pk}_E, \mathsf{osk}_E)$. The master certifier and an opener runs this protocol such that the opener will get an opening key for an identity $E \in \mathbb{H}$.

1. The opener first runs $(\mathsf{ppk}_E, \mathsf{uk}_E) \leftarrow \mathtt{SetUserSec}(\mathsf{mpk}_{\mathcal{CLE}}, E)$.
2. The master certifier runs $\mathsf{psk}_E \leftarrow \mathcal{CLE}.\mathtt{Issue}(\mathsf{msk}_{\mathcal{CLE}}, \mathsf{mpk}_{\mathcal{CLE}}, E, \mathsf{ppk}_E)$ and $(\mathsf{upk}_{E,\mathcal{CLE}}, \mathsf{usk}_{E,\mathcal{CLE}}) \leftarrow \mathcal{CLE}.\mathtt{UKeyGen}(\mathsf{mpk}_{\mathcal{CLE}}, \mathsf{ppk}_E, \mathsf{psk}_E, \mathsf{usk}_E)$, where ok is parsed as $\mathsf{msk}_{\mathcal{CLE}}$.
3. The master certifier outputs $\mathsf{usk}_{E,\mathcal{CLE}}$ as the certified limited opening key osk_E, and publishes $\mathsf{upk}_{E,\mathcal{CLE}}$ as pk_E for identity E.

$\mathtt{Sign}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{ID}}, \mathsf{pk}_E, E, M) \rightarrow \sigma$. The input E is the identity of the opener, and pk_E is the public key of the opener generated by the algorithm $\mathtt{TrapGen}$. To sign on a message $M \in \mathbb{H}$ by $\mathsf{usk}_{\mathsf{ID}}$, a user performs the following steps.

1. $(\mathsf{ovk}, \mathsf{osk}) \leftarrow \mathcal{OTS}.\mathtt{KeyGen}()$,
2. $\sigma' \leftarrow \mathcal{SPS}.\mathtt{Sign}(\mathsf{sk}_{\mathsf{ID}}, (M, E, \mathsf{ovk}))$.
3. $C \leftarrow \mathcal{CLE}.\mathtt{Enc}(\mathsf{mpk}_{\mathcal{CLE}}, \mathsf{pk}_E, E, \mathsf{ID})$.
4. Run $\mathcal{GS}.\mathtt{Prove}()$ to generate the proof

$$\begin{aligned}\pi = PoK\{&(\mathsf{vk}_{\mathsf{ID}}, \mathsf{cert}_{\mathsf{ID}}, \mathsf{ID}, \sigma') : 1 \leftarrow \mathcal{SPS}.\mathtt{Verify}(\mathsf{vk}_{\mathsf{ID}}, (M, E, \mathsf{ovk}), \sigma')\\ &\wedge 1 \leftarrow \mathcal{SPS}_G.\mathtt{Verify}(\mathsf{vk}_G, (\mathsf{ID}, \mathsf{vk}_{\mathsf{ID}}), \mathsf{cert}_{\mathsf{ID}})\\ &\wedge C \leftarrow \mathcal{CLE}.\mathtt{Enc}(\mathsf{mpk}_{\mathcal{CLE}}, \mathsf{pk}_E, E, \mathsf{ID})\}.\end{aligned}$$

5. $\sigma'' \leftarrow \mathcal{OTS}.\mathtt{Sign}(\mathsf{osk}, (C, \pi))$.

Output $\sigma = (\pi, C, E, \mathsf{ovk}, \sigma'')$ as the group signature.

$\mathtt{Verify}(\mathsf{mpk}, M, \sigma) \rightarrow 1/0$. The verifier parses σ as $(\pi, C, E, \mathsf{ovk}, \sigma'')$. If the algorithm $\mathcal{OTS}.\mathtt{Verify}(\mathsf{ovk}, (C, \pi), \sigma'')$ outputs 1 and $\mathcal{GS}.\mathtt{Verify}()$ outputs 1 for π (i.e., π is a valid proof), the verifier outputs 1 and accepts the group signature σ; Otherwise, the verifier outputs 0.

$\mathtt{Open}(\mathsf{mpk}, \mathsf{pk}_E, \mathsf{osk}_E, \sigma) \rightarrow \mathsf{ID}/\perp$. An opener parses mpk as $(\mathsf{vk}_G, \mathsf{mpk}_{\mathcal{CLE}})$ and σ as $(\pi, C, E, \mathsf{ovk}, \sigma'')$. It returns $\perp$ if $0 \leftarrow \mathtt{Verify}(\mathsf{mpk}, M, \sigma)$. Otherwise, it computes $\mathsf{ID} \leftarrow \mathcal{CLE}.\mathtt{Dec}(\mathsf{mpk}_{\mathcal{CLE}}, \mathsf{pk}_E, \mathsf{psk}_E, C)$ and outputs ID.

Theorem 4. *The proposed group signature scheme with certified limited opening provides traceability, anonymity, and is existentially unforgeable against adaptive chosen-message attack (EUF-CMA secure) if $\mathcal{GS}$ is an non-interactive zero-knowledge proof, $\mathcal{CLE}$ is CPA/CCA secure, $\mathcal{SPS}_G$ and $\mathcal{SPS}$ are both EUF-CMA secure, and $\mathcal{OTS}$ is one-time secure (only for CCA-anonymity).*

The proof is deferred to the full version.

Remarks. Two specific steps of $\texttt{Sign}()$, namely, $\sigma' \leftarrow \mathcal{SPS}.\texttt{Sign}(\mathsf{sk}_{\mathsf{ID}}, (M, E, \mathsf{ovk}))$ and $C \leftarrow \mathcal{CLE}.\texttt{Enc}(\mathsf{mpk}_{\mathcal{CLE}}, \mathsf{pk}_E, E, \mathsf{ID})$ merit more discussion. With the use of $\mathcal{SPS}$, our group signature scheme can sign on group element $M \in \mathbb{H}$. With our SP-CLE, pk_E and E are both group elements. It is thus easy to use Groth-Sahai proof to, say prove that the opener is among one of a known list of n openers.

6 Conclusion

We propose a series of structure-preserving certificateless encryption schemes by extending an existing structure-preserving signature scheme. We illustrate their applications in group signature with certified limited opening. We leave it as a future work to use our structure-preserving certificateless encryption scheme for other accountable privacy features, e.g., escrowed linkability [16] in which two anonymous signatures from the same signer can only be linked by the one who owns the private key (in our structure-preserving certificateless encryption).

Our scheme supports typical application of CLE except "encrypt to the future" [15,22,29]. We leave it as an open problem to devise an SP-CLE under the original formulation [5]. Another future work is to propose a generic way to construct SP-CLE from any SPS scheme, without any step verifying an SPS in the encryption algorithm. A challenge is to generically "upgrade" the complexity assumption required for the SPS to its decisional variant required by SP-CLE.

A Towards Removing $\mathbb{G}_T$ Elements from the Ciphertext

Recall that in our basic scheme (Sect. 4.2)

$$K = \{e(W_2, \tilde{R})e(U, \tilde{R})/e(W_1, h)\}^x \{e(\mathsf{ID}, \tilde{V_1})e(D_\alpha, \tilde{V_2})/e(g, h)\}^y / e(D_\alpha, h)^z.$$

We include the following terms in the ciphertext such that $\prod_{i=1}^{4}\{e(C_i, \tilde{C_i})\} = K$.

$$\begin{aligned} &C_1 = ((W_2 \cdot U)^x)^{r_1}, && \tilde{C}_1 = \tilde{R}^{1/r_1}, && C_2 = (\mathsf{ID}^y)^{r_2}, && \tilde{C}_2 = \tilde{V}_1^{1/r_2}, \\ &C_3 = ({D_\alpha}^y)^{r_3}, && \tilde{C}_3 = \tilde{V}_2^{1/r_3}, && C_4 = ({W_1}^x/g^y/{D_\alpha}^z)^{r_4}, && \tilde{C}_4 = h^{1/r_4}. \end{aligned}$$

K can be recovered by $e(C_g, \tilde{S})e(T, C_R)e(C_z, \tilde{D}_\alpha)$ as in the decryption algorithm.

The idea of encryption/decryption is still about encoding/recovering the bits $\{\tau_j\}$ in $C_0 = M \cdot \prod_{j=1}^{l} G_j^{\tau_j}$ (Sect. 4.3). Roughly, the trick [25] has two steps. First, we replicate K into l versions by different randomness. Second, we replicate the master public key and the private key into two versions based on different generators. To encode $\tau_j = 0$, both encryption and decryption should use the first version of the corresponding key. Similarly, $\tau_j = 1$ takes the second version.

References

1. Abe, M., Chow, S.S.M., Haralambiev, K., Ohkubo, M.: Double-trapdoor anonymous tags for traceable signatures. Int. J. Inf. Secur. **12**(1), 19–31 (2013)
2. Abe, M., David, B., Kohlweiss, M., Nishimaki, R., Ohkubo, M.: Tagged one-time signatures: tight security and optimal tag size. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 312–331. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_20
3. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_12
4. Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal structure-preserving signatures in asymmetric bilinear groups. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 649–666. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_37
5. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40061-5_29
6. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless public key encryption without pairing. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 134–148. Springer, Heidelberg (2005). https://doi.org/10.1007/11556992_10
7. Barbosa, M., Farshim, P.: Relations among notions of complete non-malleability: indistinguishability characterisation and efficient construction without random oracles. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 145–163. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14081-5_10
8. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_38
9. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. **36**(5), 1301–1328 (2007)
10. Camenisch, J., Haralambiev, K., Kohlweiss, M., Lapon, J., Naessens, V.: Structure preserving CCA secure encryption and applications. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 89–106. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_5
11. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_33
12. Chow, S.S.M.: Certificateless encryption. In: Identity-Based Cryptography. Cryptology and Information Security Series, vol. 2, pp. 135–155. IOS Press (2008)
13. Chow, S.S.M.: Real traceable signatures. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 92–107. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05445-7_6
14. Chow, S.S.M., Franklin, M.K., Zhang, H.: Practical dual-receiver encryption - soundness, complete non-malleability, and applications. In: The Cryptographer's Track at the RSA Conference (CT-RSA), pp. 85–105 (2014)

15. Chow, S.S.M., Roth, V., Rieffel, E.G.: General certificateless encryption and timed-release encryption. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 126–143. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85855-3_9
16. Chow, S.S.M., Susilo, W., Yuen, T.H.: Escrowed linkability of ring signatures and its applications. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 175–192. Springer, Heidelberg (2006). https://doi.org/10.1007/11958239_12
17. Chow, S.S.M., Zhang, H., Zhang, T.: Real hidden identity-based signatures. In: Financial Cryptography and Data Security (FC), pp. 21–38 (2017)
18. Dent, A.W.: A brief introduction to certificateless encryption schemes and their infrastructures. In: Martinelli, F., Preneel, B. (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 1–16. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16441-5_1
19. Dent, A.W., Libert, B., Paterson, K.G.: Certificateless encryption schemes strongly secure in the standard model. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 344–359. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78440-1_20
20. Girault, M.: Self-certified public keys. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 490–497. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_42
21. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. SIAM J. Comput. **41**(5), 1193–1232 (2012)
22. Kasamatsu, K., Matsuda, T., Emura, K., Attrapadung, N., Hanaoka, G., Imai, H.: Time-specific encryption from forward-secure encryption. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 184–204. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32928-9_11
23. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_34
24. Kiayias, A., Zhou, H.: Hidden identity-based signatures. IET Inf. Secur. **3**(3), 119–127 (2009)
25. Libert, B., Joye, M.: Group signatures with message-dependent opening in the standard model. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 286–306. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04852-9_15
26. Libert, B., Mouhartem, F., Nguyen, K.: A lattice-based group signature scheme with message-dependent opening. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 137–155. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_8
27. Libert, B., Peters, T., Qian, C.: Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10174, pp. 247–276. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54365-8_11
28. Ohara, K., Sakai, Y., Emura, K., Hanaoka, G.: A group signature scheme with unbounded message-dependent opening. In: ACM SIGSAC Symposium on Information, Computer and Communications Security (AsiaCCS), pp. 517–522. ACM (2013)
29. Paterson, K.G., Quaglia, E.A.: Time-specific encryption. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 1–16. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15317-4_1

30. Sakai, Y., Emura, K., Hanaoka, G., Kawai, Y., Matsuda, T., Omote, K.: Group signatures with message-dependent opening. In: Abdalla, M., Lange, T. (eds.) Pairing 2012. LNCS, vol. 7708, pp. 270–294. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36334-4_18
31. Sun, Y., Zhang, F., Baek, J.: Strongly secure certificateless public key encryption without pairing. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 194–208. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76969-9_13
32. Tsang, P.P., Chow, S.S.M., Smith, S.W.: Batch pairing delegation. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 74–90. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75651-4_6