Self-Generated-Certificate Public Key Cryptography and Certificateless Signature / Encryption Scheme in the Standard Model

Joseph K. Liu[1], Man Ho Au[2] and Willy Susilo[2]

Department of Computer Science
University of Bristol
Bristol, BS8 1UB, UK
liu@cs.bris.ac.uk
Centre for Information Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, Australia
mhaa456,wsusilo@uow.edu.au

Abstract. Certificateless Public Key Cryptography (CL-PKC) enjoys a number of features of Identity-Based Cryptography (IBC) while without having the problem of key escrow. However, it *does* suffer to an attack where the adversary, Carol, replaces Alice's public key by someone's public key so that Bob, who wants to send an encrypted message to Alice, uses Alice's identity and other's public key as the inputs to the encryption function. As a result, Alice cannot decrypt the message while Bob is unaware of this. We call it *Denial-of-Decryption (DoD) Attack* as its nature is similar to the well known Denial-of-Service (DoS) Attack. Based on CL-PKC, we propose a new paradigm called *Self-Generated-Certificate Public Key Cryptography (SGC-PKC)* that captures the DoD Attack. We also provide a generic construction of a self-generated-certificate public key encryption scheme in the standard model. Our generic construction uses certificateless signature and certificateless encryption as the building block.

In addition, we further propose a certificateless signature and a certificateless encryption scheme with concrete implementation that are all provably secure in the standard model, which are the first in the literature regardless of the generic constructions by Yum and Lee which may contain security weaknesses as pointed out by others. We believe these concrete implementations are of independent interest.

1 Introduction

In traditional public key cryptography (PKC), a user, Alice, selects a private key and computes the corresponding public key. The public key is published. Bob who wants to send an encrypted message to Alice needs to know her public key. However, an adversary Carol may replace Alice's public key by her own one so that Bob in fact encrypts the message using Carol's public key and he is unaware of this replacement attack, which lets Carol, the adversary, to decrypt the message designated for Alice. This attack may be defended if Alice's public key is authenticated by a trusted party. We usually refer it as the "Certification Authority (CA)" which signs Alice's public key and issues a digital certificate containing Alice's public key and her information (such as name, organization etc.) In order to send an encrypted message to Alice, her public key should be checked against the certificate. However, the realization of this authentication mechanism is not practical. The requisite explicit certificate and trusted authority are the main concerns and they are yet to be solved.

Identity-Based Cryptography (IBC). Identity-based cryptography (IBC), invented by Shamair [22] in 1984, solves this problem by using Alice's identity (or email address) which is an arbitrary string as her public key while the corresponding private key is a result of some mathematical operation that takes as input the user's identity and the master secret key of a trusted authority, referred as "Private Key Generator (PKG)". In this way, the certificate is implicitly provided and it is no longer necessary to explicitly authenticate public keys. The main disadvantage of identity-based cryptography is an unconditional trust to the PKG. This is even worse than traditional PKC since the secret key of every user is generated by the PKG, it can impersonate any user, or decrypt any ciphertext.

Certificateless Public Key Cryptography (CL-PKC). In order to solve for this problem, certificateless public key cryptography (CL-PKC) was proposed. It was first invented by Al-Riyami and Paterson [1] in 2003. It is a new paradigm which lies between identity-based cryptography and traditional public key cryptography. The concept was to eliminate the inherent key-escrow problem of identity-based cryptography. At the same time, it preserves the attractive advantage of IBC which is the absence of digital certificates (issued by Certificate Authority) and their important management overhead. Different from IBC, the user's public key is no longer an arbitrary string. Instead it is similar to PKC where the public key is generated by the user. A significant difference between them is that the public key in CL-PKC does not need to be explicitly certified by a trusted party (which is the CA in the case of PKC) as it has been generated by some "partial secret key" obtained from a trusted authority called "Key Generation Center (KGC)". More importantly, the KGC does *not* know the user's secret key since it contains some secret information which is only known by the user himself. Thus it solves the problem of key-escrow inherent in IBC.

Similar idea was introduced earlier by Girault [11] in 1991 and further developed in [19,20], as self-certified public keys. These schemes are structurally similar to CL-PKC. In a self-certified scheme, a user chooses his secret key and corresponding public key. He delivers his public key to a trusted party. The trusted party combines his public key with his identity to produce a witness. This witness may just be the trusted party's signature on the combination of the user's public key and identity [11], part of a signature [19] or the result of inverting a trapdoor one-way function based on the user's public key and identity [20]. Given this witness together with the public key of the trusted party and identity from the user, everyone can compute his public key. Certificate is provided implicitly inside the witness and the identity. They claim to be able to use the public key cryptography without traditional certificates. However, it can be regarded the witness as a shorten certificate. Moreover, Saeednia [21] pointed out that the scheme in [11] allows a cheating trusted party to extract users' private keys which suffers the same problem as IBC. Detailed comparison can be referred to [2].

Certificate Based Encryption(CBE). Independently of [1], Gentry [10] introduced a different but related concept, called Certificate Based Encryption (CBE). This approach is closer to the context of a traditional PKC model as it also involves a certification authority providing an efficient implicit certification service for users' public keys. However, in this model certificate is a part of secret key, so that certification is implicit. Furthermore, there is no key escrow since certificate itself is only a part of the secret key while the other part is only known to the user himself.

It seems that both CL-PKC and CBE can solve the problem of explicit certification. Nevertheless they both suffer the following attack. Suppose Alice wants to send an encrypted email to Bob. She takes Bob's public key and his identity (or personal information) as input to the encryption function. However, Carol, the adversary, has replaced Bob's public key by someone's public key (this maybe her own public key or a public key from other people). Alice is unaware of this replacement and continues to execute the encryption algorithm using Bob's identity and a public key not belonged to Bob. Although Carol cannot decrypt the ciphertext, nor Bob can do it. This is a kind of destructive behaviour. Carol cannot gain advantage by herself but she has also prevented Bob from getting the deserved information. We call it *Denial-of-Decryption (DoD) Attack*. (This is similar to Denial of Service (DoS) Attack in the way that the attacker cannot gain any secret information but precluding others from getting the normal service.) Under either CL-PKC or CBE model, Carol can succeed to launch the attack since there is no checking whether the public key is associated with the corresponding person or not. They cannot yet supersede the traditional PKC completely which is able to defend this kind of attack by the explicit certificate. Note that signature schemes are immune to this attack since the verification requires both the identity and public key of the signer. If either one of them is replaced by the adversary, the verification outputs invalid.

In parallel to this active attack, one may consider another daily life scenario. Again Alice wants to send an encrypted email to Bob. This time Alice cannot correctly identify which public key to use from a range that are made available to her, knowing that choosing the wrong one will result in her message not getting through. This is a fundamental problem in distributing certificateless public keys. This cannot be solved under either CL-PKC or CBE model.

This is the distribution problem for CL-PKC or CBE schemes. The problem is how to know which public key is correct for a user without a trust authority to vouch for it. This is one of the hugely important problems that are needed to be solved before a certificateless scheme can be used in practice.

Self-Generated-Certificate Public Key Cryptography (SGC-PKC). In this paper, we propose a new paradigm to solve the problem mentioned above while preserving all advantages of certificateless public key cryptography. Similar to CL-PKC, every user is given a partial secret key by the KGC and generates his own secret key and corresponding public key. In addition, he also needs to generate a certificate using his own secret key. The purpose of this self-generated certificate is similar to the one in traditional PKC. That is, to bind the identity (or personal information) and the pubic key together. The main difference is that, it can be verified by using the user's identity and public key only and does not require any trusted party. It is implicitly included in the user's public key. If Carol uses her public key to replace Alice's public key (or certificate), Bob can be aware of this and he may ask Alice to send him again her public key for the encryption.

It combines the advantages of traditional PKC and CL-PKC. It does not require any trusted authority as in PKC while it solves the distribution problem in CL-PKC.

Table 1 summarizes the comparison of the above cryptosystems.

	Implicit Certificates	Escrow Free	DoD Attack Free	Do not require trusted authority
Traditional PKC	X	√	√	X
Identity-based Cryptography	√	X	√	X
Certificateless PKC	√	√	X	√
Certificate-based PKC	√	√	X	√
Self-Generated-Certificate PKC	√	√	√	√

Table 1. Properties of related paradigms

Related Work. Since the introduction of Certificateless PKC [1] in 2003, in which the authors proposed a CL-encryption scheme and a CL-signature scheme and proved the security in the random oracle model, there are different variants or improvements proposed in the literature later on. Yum and Lee gave a generic construction on CL-encryption scheme [26], from any ID-based encryption scheme and any traditional public key encryption scheme. Libert and Quisquater [16] pointed out some security weakness of the generic construction in [26] and proposed a fix in the random oracle model. Independently, Bentahar et al. [4] proposed another generic construction of CL-encryption scheme which is also provable secure in the random oracle model. In addition, some concrete efficient implementations were proposed in [7,23,3,16]. The security of all these implementations relies on the random oracle model.

On the other hand, CL-signature was first proposed in the same paper as CL-encryption in [1]. The security weakness of this signature scheme was pointed out by Huang et al. [14]. They proposed a fix and proved its security in the random oracle model. A generic construction was proposed by Yum and Lee [27]. Hu et al. [13] showed that the Yum-Lee construction is insecure and proposed a fix in the standard model. A concrete implementation was proposed in [12] which is also provable secure in the random oracle model.

Some other certificateless cryptographic primitives are also proposed recently. Huang et al. [15] proposed a certificateless signature scheme with designated verifier. Chow et al. [8] proposed a Security-Mediated Certificateless Cryptography with revokation feature using a mediator. All these schemes rely on the random oracle model for proving security.

Contribution. In this paper, we propose a new paradigm called *Self-Generated-Certificate Public Key Cryptography (SGC-PKC)* which is the enhanced version of Certificateless Public Key Cryptography (CL-PKC). It captures the DoD attack mentioned above. We present a generic construction of an encryption scheme that is secure in the SGC-PKC model. Its security is proven in the standard model without relying on random oracles. It uses certificateless signature and certificateless encryption as the building block.

In addition, we propose the first certificateless signature scheme with concrete implementation in the standard mode as a primitive to our self-certified-certificate public key encryption scheme. Our security model is stronger than the generic construction given in [13].

We also propose a certificateless encryption scheme in the standard model which is the first in the literature regardless the generic construction from Yum and Lee [26].[1] They are of independent interest.

Organization. The rest of the paper is organized as follow. We give some definitions in Section 2. We propose a CL-signature and CL-encryption scheme in Section 3 and 4 respectively. The proposed Self-Generated-Certificate encryption scheme is presented in Section 5. Finally a concluding remark is given in Section 6.

2 Definition

2.1 Security of Certificateless Signature

We enhance the model of Hu et al. [13], which is the strongest among those in the literature (Al-Riyami and Paterson did not really develop a full security model for certificateless signature in [1]. It was later in [27,14] that more formalized and complete models were specified.)

Definition 1 (Definition of Certificateless Signature). *A certificateless signature scheme is a* 5*-tuple algorithms which are defined as follow:*

- **Setup:** is a probabilistic polynomial time (PPT) algorithm run by a Key Generation Centre (KGC), given a security parameter k' as input, outputs a randomly chosen master secret key mk and a list of public parameters param.
- **Partial-Secret-Key-Extract:** is PPT algorithm, run by the KGC, given a user's identity ID and the master secret key mk as inputs, outputs a partial-secret-key psk.
- **User-Key-Generation:** is PPT algorithm, run by the user, given a list of public parameters param as inputs, outputs a secret key sk and a public key pk.
- **Sign:** is a PPT algorithm, given list of parameters param, a user secret key sk, user partial secret key psk, a message m as inputs, outputs a signature σ.
- **Verify:** is a deterministic algorithm, given list of parameters param, a user identity ID, user public key pk, a message m and a signature σ as inputs, outputs either accept or reject.

For correctness, as usual we require that $\mathbf{Verify}(\mathsf{param}, \mathsf{m}, \sigma, \mathsf{ID}, \mathsf{pk}) = \mathsf{accept}$ whenever for all $k \in \mathbb{N}$, $m \in \{0,1\}^*$, $\mathsf{ID} \in \{0,1\}^*$, $(\mathsf{param}, \mathsf{mk}) \leftarrow \mathbf{Setup}(k)$, $\mathsf{psk} \leftarrow \mathbf{Partial\text{-}Secret\text{-}Key\text{-}Extract}(\mathsf{param}, \mathsf{mk}, \mathsf{ID})$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathbf{User\text{-}Key\text{-}Generation}\ (\mathsf{param})$, $\sigma = \mathbf{Sign}(\mathsf{param}, \mathsf{sk}, \mathsf{psk}, \mathsf{m})$.

Security Model. According to the original scheme in [1], there are two types of adversaries. Type I adversary does not have the KGC's master secret key but it can replace public keys of arbitrary identities with other public keys of its own choices. It can also obtain partial and full secret keys of arbitrary identities.

Type II adversary knows the master secret key (hence it can compute partial secret key by itself). It is still allowed to obtain full secret key for arbitrary identities but is not allowed to replace public keys at any time. Same as [13], we also assume that the KGC generates the master secret key according to the scheme specification.

Definition 2 (Existential Unforgeability). *A certificateless signature scheme is existential unforgeable against chosen message attack if no PPT adversary $\mathcal{A}$ of Type I or Type II has a non-negligible advantage in the following game played against the challenger:*

1. The challenger takes a security parameter k' and runs the **Setup** algorithm. It gives $\mathcal{A}$ the resulting system parameters param. If $\mathcal{A}$ is of Type I, the challenger keeps the master secret key mk to itself, otherwise, it gives mk to $\mathcal{A}$.
2. $\mathcal{A}$ is given access to the following oracles:
   - Public-Key-Broadcast-Oracle: on input an identity, it outputs the matching public key.

[1] The generic construction from Libert and Quisquater [16] and Bentahar et al. [4] relies on the random oracle model.

- Partial-Secret-Key-Extract-Oracle: on input an identity, it outputs partial secret key associated with the user's identity. (Note that it is only useful to Type I adversary.)
- Secret-Key-Extract-Oracle: on input an identity, it outputs secret key associated with the user's identity. It outputs $\perp$ if the user's public key has been replaced (in the case of Type I adversary).
- Public-Key-Replace-Oracle: (For Type I adversary only) on input an identity and a valid public key, it replaces the associated user's public key with the new one.
- Signing-Oracle: on input an identity and a message, it outputs a valid signature σ no matter whether the public key of the identity has not been replaced or not if it is Type I adversary. [2]

3. $\mathcal{A}$ outputs $(\mathsf{ID}^*, \mathsf{m}^*, \sigma^*)$. It wins if $\mathbf{Verify}(\mathsf{param}, \mathsf{ID}^*, \mathsf{pk}_{\mathsf{ID}^*}, \mathsf{m}^*, \sigma^*) = \mathsf{valid}$ and fulfills the following conditions:
   - $(\mathsf{ID}^*, \mathsf{m}^*)$ has not been submitted to Signing-Oracle.
   - If it is Type I, ID^* has not been submitted to both Partial-Secret-Key-Extract-Oracle and, Public-Key-Replace-Oracle or Secret-Key-Extract-Oracle.
   - If it is Type II, ID^* has not been submitted to Secret-Key-Extract-Oracle.

Define the advantage of $\mathcal{A}$ as: $\mathsf{Adv}^{EF}_{CLS}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$

Note that Type I adversary is allowed to make Secret-Key-Extract-Oracle queries and gets the user secret key $\mathsf{sk}_{\mathsf{ID}^*}$ or queries Public-Key-Replace-Oracle to replace the public key of ID^* before generating a forgery in Step (3). This is to capture the attack where the signature should still be secure even if $\mathsf{sk}_{\mathsf{ID}^*}$ is compromised provided that $\mathsf{psk}_{\mathsf{ID}^*}$ is not. For details please refer to [13].

We remark that, in [13], they allow the signing oracle to output invalid signature if the public key of the corresponding identity has been replaced. We require in a stronger sense that even the public key has been replaced, the signing oracle *should* output a valid signature.

2.2 Security of Certificateless Encryption

In the security model of certificateless encryption scheme, we slightly modify the original one from Al-Riyami and Paterson [1]. Instead we use the simplification as used in [13] in CL-signature. The discussion of the main difference can be referred to [13]. We skip it here due to page limitation.

Definition 3 (Definition of Certificateless Encryption). *A certificateless encryption scheme is a* 5*-tuple algorithms which are defined as follow:* [3]

- **Setup**, **Partial-Secret-Key-Extract**, **User-Key-Generation**: Same as Definition 1.
- **Encrypt:** is a PPT algorithm, given a plaintext m, list of parameters param, a receiver's identity ID and his public key pk as inputs, outputs either a ciphertext $C = \mathbf{Encrypt}(\mathsf{param}, \mathsf{m}, \mathsf{ID}, \mathsf{pk})$ or $\perp$ meaning encryption failure. This will occur if in the event that pk does not have the correct form.
- **Decrypt:** is a deterministic algorithm, given a ciphertext C, a list of public parameters param, a user secret key sk and a user partial secret key psk as inputs, outputs either a plaintext m or $\perp$ meaning decryption failure.

For correctness, as usual we require that $\mathbf{Decrypt}(\mathsf{param}, C, \mathsf{sk}, \mathsf{psk}) = \mathsf{m}$ whenever $C = \mathbf{Encrypt}(\mathsf{param}, \mathsf{m}, \mathsf{ID}, \mathsf{pk})$.

Security Model. Same as above in the case of signature, there are two types of adversary. Type I adversary does not have the KGC's master secret key but it can replace public keys of arbitrary identities with other public keys of its own choices. It can also obtain partial and full secret keys of arbitrary identities.

Type II adversary knows the master secret key (hence it can compute partial secret key by itself). It is still allowed to obtain full secret key for arbitrary identities but is not allowed to replace public keys at any time.

[2] Note that it is even stronger than the model of Hu et al. [13] in which they allow the signing oracle to output some invalid signatures if the public key of the corresponding identity has been replaced.

[3] In the original model in [1], it is a 7-tuple algorithms. We use the same simplification as in [13] to reduce it as a 5-tuple algorithms.

In considering the IND-CCA secure scenario, the strongest model in [1] does expect the challenger to be able to correctly respond to decryption queries made on identities for which the Type I adversary has replaced the public keys. That is, the decryption oracle should be able to output consistent answers even for identities whose public keys have been replaced and for which they do not know the corresponding private keys. This is a very strong notion of security. Several schemes [4,7,26] have weaken this definition to that the challenger is not forced to attempt to decrypt ciphertexts for which the public key has been replaced, if the corresponding secret key is not known. It is known as Type I^- adversary.

In our scheme, we adopt the security against Type I^- and Type II adversary.

Definition 4 (IND-CCA$^-$ Security). *A certificateless encryption scheme is IND-CCA$^-$ secure if no PPT adversary $\mathcal{A}$ of Type I^- or Type II has a non-negligible advantage in the following game played against the challenger:*

1. The challenger takes a security parameter k' and runs the **Setup** algorithm. It gives $\mathcal{A}$ the resulting system parameters param. If $\mathcal{A}$ is of Type I^-, the challenger keeps the master secret key mk to itself, otherwise, it gives mk to $\mathcal{A}$.
2. $\mathcal{A}$ is given access to the following oracles:
   - Public-Key-Broadcast-Oracle: on input identity, it outputs the matching public key.
   - Partial-Secret-Key-Extract-Oracle: on input identity, it outputs partial secret key associated with the user's identity. (Note that it is only useful to Type I^- adversary.)
   - Secret-Key-Extract-Oracle: on input identity, it outputs secret key associated with the user's identity. It outputs $\perp$ if the user's public key has been replaced (in the case of Type I^- adversary).
   - Decryption-Oracle: on input a ciphertext and an identity, returns the decrypted plaintext using the secret key corresponding to the current value of the public key associated with the identity of the user. If the user's public key has been replaced, it requires an additional input of the corresponding secret key for the decryption. If it is not given this secret key, it outputs $\perp$ (in the case of Type I^- adversary).
   - Public-Key-Replace-Oracle: (For Type I^- adversary only) on input identity and a valid public key, it replaces the associated user's public key with the new one.
3. After making oracle queries a polynomial times, $\mathcal{A}$ outputs and submits two messages m_0, m_1, together with an identity ID^* of uncorrupted secret key to the challenger. The challenger picks a random bit $b \in \{0,1\}$ and computes C^*, the encryption of m_b under the current public key pk^* for ID^*. If the output of the encryption is $\perp$, then $\mathcal{A}$ immediately looses the game. Otherwise C^* is delivered to $\mathcal{A}$.
4. $\mathcal{A}$ makes a new sequence of queries.
5. $\mathcal{A}$ outputs a bit b'. It wins if $b' = b$ and fulfills the following conditions:
   - In Step (4), C^* has not been submitted to Decryption-Oracle for the combination (ID^*, pk^*) under which m_b was encrypted.
   - If it is Type I^-, ID^* has not been submitted to both Partial-Secret-Key-Extract-Oracle at some step and, Public-Key-Replace-Oracle or Secret-Key-Extract-Oracle before Step (3).
   - If it is Type II, ID^* has not been submitted to Secret-Key-Extract-Oracle.

Define the advantage of $\mathcal{A}$ as: $\mathsf{Adv}_{CLE}^{IND-CCA^-}(\mathcal{A}) = 2\Pr[\mathcal{A} \text{ wins}] - 1$

Note that the only difference between a Type I and Type I^- Adversary is on the Decryption-Oracle. Type I Adversary requires Decryption-Oracle to output a valid plaintext without any additional input even in the case that the corresponding public key has been replaced. It is denoted as IND-CCA Secure.

2.3 Security of Self-Generated-Certificate (SGC) Encryption

The definition of SGC Encryption is the same as the definition of CL Encryption given in Definition 3, except for **User-Key-Generation** which also needs the partial secret key and the identity of the user as the input. That is, we require **Partial-Secret-Key-Extract** to be executed before **User-Key-Generation**.

For security, in addition to IND-CCA (or IND-CCA$^-$), we require the scheme to be DoD-Free, which is formally defined as follow as a game played between the challenger and a PPT adversary (DoD Adversary), which has the same power of a Type I (or Type I^-) adversary defined in CL Encryption.

Definition 5 (DoD-Free Security). *A SGC encryption scheme is DoD-Free secure if no PPT adversary $\mathcal{A}$ has a non-negligible advantage in the following game played against the challenger:*

1. The challenger takes a security parameter k' and runs the **Setup** algorithm. It gives $\mathcal{A}$ the resulting system parameters param. The challenger keeps the master secret key mk to itself.
2. $\mathcal{A}$ is given access to the following oracles:
   - Public-Key-Broadcast-Oracle: on input identity, it outputs the matching public key.
   - Partial-Secret-Key-Extract-Oracle: on input identity, it outputs partial secret key associated with the user's identity.
   - Secret-Key-Extract-Oracle: on input identity and partial secret key, it outputs secret key associated with the user's identity. It outputs $\perp$ if the user's public key has been replaced.
   - Decryption-Oracle: on input a ciphertext and an identity, returns the decrypted plaintext using the secret key corresponding to the current value of the public key associated with the identity of the user. If the user's public key has been replaced, it requires an additional input of the corresponding secret key for the decryption. If it is not given this secret key, it outputs $\perp$.
   - Public-Key-Replace-Oracle: on input identity and a valid public key, it replaces the associated user's public key with the new one.
3. After making oracle queries a polynomial times, $\mathcal{A}$ outputs a message m^*, together with an identity ID^* to the challenger. The challenger computes C^*, the encryption of m^* under the current public key pk^* for ID^*.
   If the output of the encryption is $\perp$, then $\mathcal{A}$ immediately looses the game. Otherwise it outputs C^*.
4. $\mathcal{A}$ wins if the following conditions are fulfilled:
   - The output of the encryption in Step (3) is not $\perp$.
   - $\mathbf{Decrypt}(C^*, \mathsf{sk}^*, \mathsf{psk}^*) \neq m^*$.
   - At any time, ID^* has not been submitted to Partial-Secret-Key-Extract-Oracle, and thus Secret-Key-Extract-Oracle (since the query of Secret-Key-Extract-Oracle requires the knowledge of partial secret key. Yet $\mathcal{A}$ is allowed to query Public-Key-Replace-Oracle at any time for any identity.)

Define the advantage of $\mathcal{A}$ as: $\mathsf{Adv}_{SGCE}^{DoD-Free}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$

Definition 6. *A SGC encryption scheme is secure if it is DoD-Free secure and IND-CCA (or IND-CCA$^-$) secure.*

3 A Certificateless Signature Scheme in the Standard Model

3.1 Construction

Our scheme is motivated from the identity-based signature scheme from Paterson and Schuldt [18]. Let $H_u : \{0,1\}^* \to \{0,1\}^{n_u}$ and $H_m : \{0,1\}^* \to \{0,1\}^{n_m}$ be two collision-resistant cryptographic hash functions for some $n_u, n_m \in \mathbb{Z}$. They are used to create identities and messages of the desired length respectively.

Setup. Select a pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ where the order of $\mathbb{G}_1$ is p. Let g be a generator of $\mathbb{G}_1$. Randomly select $\alpha \in_R \mathbb{Z}_p$, $g_2 \in_R \mathbb{G}_1$ and compute $g_1 = g^\alpha$. Also select randomly the following elements:

$$u', m' \in_R \mathbb{G}_1 \qquad \hat{u}_i \in_R \mathbb{G}_1 \text{ for } i = 1, \ldots, n_u \qquad \hat{m}_i \in_R \mathbb{G}_1 \text{ for } i = 1, \ldots, n_m \qquad \text{Let } \hat{U} = \{\hat{u}_i\}, \hat{M} = \{\hat{m}_i\}$$

The public parameters param are $(e, \mathbb{G}_1, \mathbb{G}_2, g, g_1, g_2, u', \hat{U}, m', \hat{M})$ and the master secret key is g_2^α.

Partial-Secret-Key (PSK)-Extract. Let $\mathfrak{u} = H_u(ID)$ for user with identity ID. Let $\mathfrak{u}[i]$ be the i-th bit of $\mathfrak{u}$. Define $\mathcal{U} \subset \{1, \ldots, n_u\}$ to be the set of indicies such that $\mathfrak{u}[i] = 1$.

To construct the PSK of identity ID, the master randomly selects $r_u \in_R \mathbb{Z}_p$ and compute

$$\left(g_2^\alpha \left(U\right)^{r_u}, g^{r_u}\right) = (\mathsf{psk}^{(1)}, \mathsf{psk}^{(2)}) \qquad \text{where } U = u' \prod_{i \in \mathcal{U}} \hat{u}_i$$

User-Key-Generation. User selects a secret value $x \in \mathbb{Z}_p$ as his secret key sk, and computes his public key as $(g^x, g_1^x) = (\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)})$

Sign. To sign a message $m \in \{0,1\}^*$, the signer with identity ID, partial secret key $(\mathsf{psk}^{(1)}, \mathsf{psk}^{(2)})$ and secret key sk and compute $\mathfrak{m} = H_m(m)$. Let $\mathfrak{m}[i]$ be the i-th bit of $\mathfrak{m}$ and $\mathcal{M} \subset \{1, \ldots, n_m\}$ be the set of indicies i such that $\mathfrak{m}[i] = 1$. Randomly select $r_\pi, r_\mathfrak{m} \in_R \mathbb{Z}_p$, compute $U = u' \prod_{i \in \mathcal{U}} \hat{u}_i$ and

$$\sigma = \left(\left(\mathsf{psk}^{(1)}\right)^{\mathsf{sk}} \left(U\right)^{r_\pi} \left(m' \prod_{i \in \mathcal{M}} \hat{m}_i\right)^{r_\mathfrak{m}} \ , \ \left(\mathsf{psk}^{(2)}\right)^{\mathsf{sk}} g^{r_\pi} \ , \ g^{r_\mathfrak{m}} \right) \ = (V, R_\pi, R_\mathfrak{m})$$

Verify. Given a signature $\sigma = (V, R_\pi, R_\mathfrak{m})$ for an identity ID and public key $(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)})$ on a message m, a verifier first computes $\mathfrak{m} = H_m(m)$, $U = u' \prod_{i \in \mathcal{U}} \hat{u}_i$ and checks whether

$$e(\mathsf{pk}^{(1)}, g_1) \stackrel{?}{=} e(\mathsf{pk}^{(2)}, g) \qquad \text{and} \qquad e(V, g) \stackrel{?}{=} e(g_2, \mathsf{pk}^{(2)}) \ e(U, R_\pi) \ e(m' \prod_{i \in \mathcal{M}} \hat{m}_i, R_\mathfrak{m})$$

Output valid if both equalities hold. Otherwise output $\mathsf{invalid}$.

3.2 Security Analysis

Correctness. It is easy to see that the signature scheme is correct, as shown in following:

$$\begin{aligned} e(V, g) &= e\Big(g_2^{\alpha x} \ U^{r_u x} \ U^{r_\pi} \ \Big(m' \prod_{i \in \mathcal{M}} \hat{m}_i\Big)^{r_m} \ , \ g \Big) = e(g_2^x, g^\alpha) \ e(U^{r_u x + r_\pi}, g) \ e\Big(\big(m' \prod_{i \in \mathcal{M}} \hat{m}_i\big)^{r_m}, g \Big) \\ &= e(g_2, g_1^x) \ e(U, g^{r_u x + r_\pi}) \ e(m' \prod_{i \in \mathcal{M}} \hat{m}_i, g^{r_\mathfrak{m}}) = e(g_2, \mathsf{pk}^{(2)}) \ e(U, R_\pi) \ e(m' \prod_{i \in \mathcal{M}} \hat{m}_i, R_\mathfrak{m}) \end{aligned}$$

Theorem 1 (Type I Existential Unforgeability). *The CL-signature scheme proposed in Section 3 is (ϵ, t)-existential unforgeable against Type I adversary (defined in Section 2) with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-NGBDH assumption[4] holds in $\mathbb{G}_1$, where $\epsilon' \geq \frac{\epsilon}{16(q_e+q_s)(n_u+1)q_s(n_m+1)}$ and $t' = t + O\Big(\big(q_e n_u + q_s(n_u + n_m)\big)\rho + (q_k + q_e + q_s)\tau\Big)$ where q_e is the number of queries made to the Partial-Secret-Key-Extract-Oracle, q_s is the number of queries made to the Signing-Oracle, q_k is the number of queries made to the Public-Key-Broadcast-Oracle and Secret-Key-Extract-Oracle altogether, and ρ and τ are the time for a multiplication and an exponentiation in $\mathbb{G}_1$ respectively.*

Proof. Assume there is a Type I adversary $\mathcal{A}$ exists. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the NGBDH problem with probability at least ϵ' and in time at most t'. We use a similar approach as in [18].

$\mathcal{B}$ is given a problem instance as follow: Given a group $\mathbb{G}_1$, a generator $g \in \mathbb{G}_1$, two elements $g^a, g^b \in \mathbb{G}_1$. It is asked to output two elements $g^{abc}, g^c \in \mathbb{G}_1$. In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulates a challenger and all oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. Let $l_u = 2(q_e + q_s)$ and $l_m = 2q_s$. $\mathcal{B}$ randomly selects two integers k_u and k_m such that $0 \leq k_u \leq n_u$ and $0 \leq k_m \leq n_m$. Also assume that $l_u(n_u+1) < p$ and $l_m(n_m+1) < p$ for the given values of q_e, q_s, q_k, n_u and n_m. It randomly selects the following integers:

$$x' \in_R \mathbb{Z}_{l_u} \ ; \ z' \in_R \mathbb{Z}_{l_m} \ ; \ y', w' \in_R \mathbb{Z}_p \qquad \hat{x}_i \in_R \mathbb{Z}_{l_u}, \text{ for } i = 1, \ldots, n_u \qquad \hat{z}_i \in_R \mathbb{Z}_{l_m}, \text{ for } i = 1, \ldots, n_m$$
$$\hat{y}_i \in_R \mathbb{Z}_p, \text{ for } i = 1, \ldots, n_u \qquad \hat{w}_i \in_R \mathbb{Z}_p, \text{ for } i = 1, \ldots, n_m \quad \text{Let } \hat{X} = \{\hat{x}_i\}, \hat{Z} = \{\hat{z}_i\}, \hat{Y} = \{\hat{y}_i\}, \hat{W} = \{\hat{w}_i\}$$

We further define the following functions for binary strings $\mathfrak{u}$ and $\mathfrak{m}$ where $\mathfrak{u} = H_u(ID)$ for an identity ID and $\mathfrak{m} = H_m(m)$ for a message m, as follow:

$$F(\mathfrak{u}) = x' + \sum_{i \in \mathcal{U}_j} \hat{x}_i - l_u k_u \qquad J(\mathfrak{u}) = y' + \sum_{i \in \mathcal{U}_j} \hat{y}_i \qquad K(\mathfrak{m}) = z' + \sum_{i \in \mathcal{M}} \hat{z}_i - l_m k_m \qquad L(\mathfrak{m}) = w' + \sum_{i \in \mathcal{M}} \hat{w}_i$$

[4] The definition of complexity assumptions are given in Appendix A.

$\mathcal{B}$ constructs a set of public parameters as follow:

$$g_1 = g^a, \qquad g_2 = g^b \qquad u' = g_2^{-l_u k_u + x'} g^{y'}, \qquad \hat{u}_i = g_2^{\hat{x}_i} g^{\hat{y}_i} \text{ for } 1 \le i \le n_u$$
$$m' = g_2^{-l_m k_m + z'} g^{w'}, \qquad \hat{m}_i = g_2^{\hat{z}_i} g^{\hat{w}_i} \text{ for } 1 \le i \le n_m$$

Note that the master secret will be $g_2^{\alpha} = g_2^a = g^{ab}$ and we have the following equations:

$$U = u' \prod_{i \in \mathcal{U}} \hat{u}_i = g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})} \qquad \text{and} \qquad m' \prod_{i \in \mathcal{M}} \hat{m}_i = g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})}$$

All public parameters are passed to $\mathcal{A}$.

Oracles Simulation. $\mathcal{B}$ simulates all oracles as follow:

(Public-Key-Broadcast-Oracle.) $\mathcal{B}$ keeps the database DB of user secret-public key. Upon receiving a query for a public key of an identity ID, $\mathcal{B}$ looks up its database DB to find out the corresponding entry. If it does not exits, $\mathcal{B}$ runs **User-Key-Generation** to generate a secret and public key pair. It stores the key pair in its database and returns the public key as the query output.

(Secret-Key-Extract-Oracle.) Upon receiving a query for a public key of an identity ID, $\mathcal{B}$ looks up its database DB to find out the corresponding entry. If it does not exits, $\mathcal{B}$ runs **User-Key-Generation** to generate a secret and public key pair. It stores the key pair in its database and returns the secret key as the query output.

(Partial-Secret-Key-Extract-Oracle.) Upon receiving a query for a partial secret key of an identity ID, B compute $\mathfrak{u} = H_u(ID)$. Although $\mathcal{B}$ does not know the master secret, it still can construct the private key by assuming $F(\mathfrak{u}) \neq 0 \bmod p$. It randomly chooses $r_u \in_R \mathbb{Z}_p$ and computes the private key as

$$(\mathsf{psk}^{(1)}, \mathsf{psk}^{(2)}) = \left(g_1^{-\frac{J(\mathfrak{u})}{F(\mathfrak{u})}} (U)^{r_u}, g_1^{-\frac{1}{F(\mathfrak{u})}} g^{r_u} \right)$$

By letting $\tilde{r}_u = r_u - \frac{a}{F(\mathfrak{u})}$, it can be verifier that psk is a valid partial secret key, shown as follow:

$$\begin{aligned}
\mathsf{psk}^{(1)} &= g_1^{-\frac{J(\mathfrak{u})}{F(\mathfrak{u})}} (U)^{r_u} = g_1^{-\frac{J(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u} = g^{-\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u} \\
&= g^{-\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\frac{a}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{-\frac{a}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u} \\
&= g^{-\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} g^{ab} g^{\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\tilde{r}_u} = g^{ab} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\tilde{r}_u} = g_2^a (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\tilde{r}_u} = g_2^a (U)^{\tilde{r}_u}
\end{aligned}$$

and $\mathsf{psk}^{(2)} = g_1^{-\frac{1}{F(\mathfrak{u})}} g^{r_u} = g^{r_u - \frac{a}{F(\mathfrak{u})}} = g^{\tilde{r}_u}$. To the adversary, all partial secret keys given by $\mathcal{B}$ are indistinguishable from the keys generated by the true challenger.

If $F(\mathfrak{u}) = 0 \bmod p$, since the above computation cannot be performed (division by 0), the simulator aborts. To make it simple, the simulator will abort if $F(\mathfrak{u}) = 0 \bmod l_u$. The equivalency can be observed as follow. From the assumption $l_u(n_u + 1) < p$, it implies $0 \le l_u k_u < p$ and $0 \le x' + \sum_{i \in \mathcal{U}_j} \hat{x}_i < p$ ($\because x' < l_u, \hat{x}_i < l_u, |\mathcal{U}| \le n_u$). We have $-p < F(\mathfrak{u}) < p$ which implies if $F(\mathfrak{u}) = 0 \bmod p$ then $F(\mathfrak{u}) \bmod l_u$. Hence, $F(\mathfrak{u}) \neq 0 \bmod l_u$ implies $F(\mathfrak{u}) \neq 0 \bmod p$. Thus the former condition will be sufficient to ensure that a private key can be computed without abort.

(Public-Key-Replace-Oracle.) Upon receiving a query for a public key replace oracle request of an identity ID, $\mathcal{B}$ looks up its database DB to replace the corresponding entry. If it does not exits, $\mathcal{B}$ creats a new entry for this identity.

(Signing-Oracle.) For a given query of a signature on an identity ID and a message m, $\mathcal{B}$ first checks from DB that whether the public key of ID has been replaced or not. If it has been replaced with public key $(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)})$, it computes the signature in the following way. Assume $K(\mathfrak{m}) \neq 0 \bmod l_m$. Using the argument mentioned above, it implies $K(\mathfrak{m}) \neq 0 \bmod p$ provided that $l_m(n_m + 1) < p$. The signature can be constructed by first randomly selecting $r_\pi, r_\mathfrak{m} \in_R \mathbb{Z}_p$, and computing

$$\begin{aligned}
\sigma &= \left((U)^{r_\pi} (\mathsf{pk}^{(2)})^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}} \left(m' \prod_{i \in \mathcal{M}} \hat{m}_i \right)^{r_\mathfrak{m}}, g^{r_\pi}, (\mathsf{pk}^{(2)})^{-\frac{1}{K(\mathfrak{m})}} g^{r_\mathfrak{m}} \right) \\
&= \left(g_2^{ax} (U)^{r_\pi} \left(m' \prod_{i \in \mathcal{M}} \hat{m}_i \right)^{\tilde{r}_m}, g^{r_\pi}, g^{\tilde{r}_m} \right) = (V, R_\pi, R_m)
\end{aligned}$$

where $\tilde{r}_m = r_{\mathfrak{m}} - \frac{ax}{K(\mathfrak{m})}$. If $K(\mathfrak{m}) = 0 \bmod l_m$, the simulator aborts.

The correctness can be shown as follow:

$$V = \big(U\big)^{r_\pi}\big(\mathsf{pk}^{(2)}\big)^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}}\Big(m' \prod_{i\in\mathcal{M}} \hat{m}_i\Big)^{r_m} = \big(U\big)^{r_\pi} g_1^{-\frac{L(\mathfrak{m})x}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{r_m} = \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{r_m}$$
$$= \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\frac{ax}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{-\frac{ax}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{r_m}$$
$$= \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}} g^{abx} g^{\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\tilde{r}_m} = \big(U\big)^{r_\pi} g^{abx}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\tilde{r}_m}$$
$$= \big(U\big)^{r_\pi} g_2^{ax}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\tilde{r}_m} = \big(U\big)^{r_\pi}\big(g_2^a\big)^x\Big(m' \prod_{i\in\mathcal{M}} \hat{m}_i\Big)^{\tilde{r}_m}$$

and $R_m = \big(\mathsf{pk}^{(2)}\big)^{-\frac{1}{K(\mathfrak{m})}} g^{r_m} = g^{r_m - \frac{ax}{K(\mathfrak{m})}} = g^{\tilde{r}_m}$. The signature generated in this way is indistinguishable to the real one.

If the public key has not been replaced, it computes $\mathfrak{u} = H_u(ID)$ and $\mathfrak{m} = H_m(m)$.

If $F(\mathfrak{u}) \neq 0 \bmod l_u$, $\mathcal{B}$ just construct a partial-secret key as in the Partial-Secret-Key-Extract-Oracle, then it checks from DB whether the secret key of ID has been created or not. If it has not been created, run the **User-Key-Generation** algorithm and stores the secret / public key pair in DB. If it has been created, it just use the **Sign** algorithm to create a signature on ID and m.

If $F(\mathfrak{u}) = 0 \bmod l_u$, $\mathcal{B}$ tries to construct the signature in a similar way as above (the case that the public key has been replaced). Assume $K(\mathfrak{m}) \neq 0 \bmod l_m$. Using the argument mentioned above, it implies $K(\mathfrak{m}) \neq 0 \bmod p$ provided that $l_m(n_m + 1) < p$. The signature can be constructed by first randomly selecting $r_\pi, r_{\mathfrak{m}} \in_R \mathbb{Z}_p$, getting the secret key x from DB (if it has not been created, run **User-Key-Generation** algorithm first) and computing

$$\sigma = \left(\big(U\big)^{r_\pi} g_1^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}x}\Big(m' \prod_{i\in\mathcal{M}} \hat{m}_i\Big)^{r_{\mathfrak{m}}x}, g^{r_\pi}, g_1^{-\frac{x}{K(\mathfrak{m})}} g^{r_{\mathfrak{m}}x}\right) = \left(g_2^{ax}\big(U\big)^{r_\pi}\Big(m' \prod_{i\in\mathcal{M}} \hat{m}_i\Big)^{\tilde{r}_m}, g^{r_\pi}, g^{\tilde{r}_m}\right)$$

where $\tilde{r}_m = r_{\mathfrak{m}}x - \frac{a}{K(\mathfrak{m})}x$. If $K(\mathfrak{m}) = 0 \bmod l_m$, the simulator aborts.

Output Calculation. If $\mathcal{B}$ does not abort, $\mathcal{A}$ will return an identity ID^* and a message m^* with a forged signature $\sigma^* = (V, R_\pi, R_{\mathfrak{m}})$ on ID^*, the current public key $\mathsf{pk}_{\mathsf{ID}^*}$ and m^* with probability at least ϵ. $\mathcal{B}$ checks whether the following conditions are fulfilled:

1. $F(\mathfrak{u}^*) = 0 \bmod p$, where $\mathfrak{u}^* = H_u(ID^*)$.
2. $K(\mathfrak{m}^*) = 0 \bmod p$, where $\mathfrak{m}^* = H_m(m^*)$.

If not all the above conditions are fulfilled, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ computes and outputs

$$\frac{V}{R_\pi^{J(\mathfrak{u}^*)} R_{\mathfrak{m}}^{L(\mathfrak{m}^*)}} = \frac{g_2^{ax}\big(U\big)^{r_\pi}\Big(m' \prod_{i\in\mathcal{M}} \hat{m}_i\Big)^{r_{\mathfrak{m}}}}{g^{J(\mathfrak{u}^*)r_\pi} g^{L(\mathfrak{m}^*)r_{\mathfrak{m}}}} = \frac{g_2^{ax}\Big(g_2^{F(\mathfrak{u}^*)}g^{J(\mathfrak{u}^*)}\Big)^{r_\pi}\Big(g_2^{K(\mathfrak{m}^*)}g^{L(\mathfrak{m}^*)}\Big)^{r_{\mathfrak{m}}}}{g^{J(\mathfrak{u}^*)r_\pi} g^{L(\mathfrak{m}^*)r_{\mathfrak{m}}}} = g_2^{ax} = g^{abx}$$

$\mathcal{B}$ outputs $(g^{abx}, \mathsf{pk}^{(1)}_{\mathsf{ID}^*}) = (g^{abx}, g^x)$ as the solution to the NGBDH problem instance.

Probability Analysis. For the simulation to complete without aborting, we require the following conditions fulfilled:

1. Partial-Secret-Key-Extract-Oracle queries on an identity ID have $F(\mathfrak{u}) \neq 0 \bmod l_u$, where $\mathfrak{u} = H_u(ID)$.
2. Signing-Oracle queries (ID, m) will either have $F(\mathfrak{u}) \neq 0 \bmod l_u$, or $K(\mathfrak{m}) \neq 0 \bmod l_m$ where $\mathfrak{m} = H_m(m)$, if the public key of ID has not been replaced. Otherwise, it requires $K(\mathfrak{m}) \neq 0 \bmod l_m$.
3. $F(\mathfrak{u}^*) = 0 \bmod l_u$ and $K(\mathfrak{m}^*) = 0 \bmod l_m$.

In order to make the analysis more simple, we will bound the probability of a subcase of this event.

Let $\mathfrak{u}_1, \ldots, \mathfrak{u}_{q_I}$ be the output of the hash function H_u appearing in either Partial-Secret-Key-Extract-Oracle queries or in Signing-Oracle queries not involving any of the challenge identity ID^*, and let

$\mathfrak{m}_1, \ldots, \mathfrak{m}_{q_M}$ be the output of the hash function H_m in the sign queries involving the challenge list. We have $q_I \leq q_e + q_s$ and $q_M \leq q_s$. We also define the events A_i, A^*, B_ℓ, B^* as follow:

$$A_i : F(\mathfrak{u}_i) \neq 0 \bmod l_u \qquad \text{where } i = 1, \ldots, q_I \qquad A^* : F(\mathfrak{u}^*) = 0 \bmod p$$
$$B_\ell : K(\mathfrak{m}_\ell) \neq 0 \bmod l_m \qquad \text{where } \ell = 1, \ldots, q_M \qquad B^* : K(\mathfrak{m}^*) = 0 \bmod p$$

The probability of $\mathcal{B}$ not aborting is: $\Pr[\mathsf{not\ abort}] \geq \Pr\left[\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right) \wedge \left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)\right]$

Note that the events $\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right)$ and $\left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)$ are independent.

The assumption $l_u(n_u + 1) < p$ implies if $F(\mathfrak{u}) = 0 \bmod p$ then $F(\mathfrak{u}) = 0 \bmod l_u$. In addition, it also implies that if $F(\mathfrak{u}) = 0 \bmod l_u$, there will be a unique choice of k_u with $0 \leq k_u \leq n_u$ such that $F(\mathfrak{u}) = 0 \bmod p$. Since k_u, x' and $\hat{X}$ are randomly chosen,

$$\begin{aligned}\Pr[A^*] &= \Pr[F(\mathfrak{u}^*) = 0 \bmod p \wedge F(\mathfrak{u}^*) = 0 \bmod l_u] \\ &= \Pr[F(\mathfrak{u}^*) = 0 \bmod l_u] \Pr[F(\mathfrak{u}^*) = 0 \bmod p \mid F(\mathfrak{u}^*) = 0 \bmod l_u] = \frac{1}{l_u}\frac{1}{n_u + 1}\end{aligned}$$

On the other hand, we have: $\Pr\left[\bigwedge_{i=1}^{q_I} A_i | A^*\right] = 1 - \Pr\left[\bigvee_{i=1}^{q_I} \overline{A_i} \mid A^*\right] \geq 1 - \sum_{i=1}^{q_I} \Pr[\overline{A_i} \mid A^*]$ where $\overline{A_i}$ denote the event $F(\mathfrak{u}_i) = 0 \bmod l_u$.

Also note that the events $F(\mathfrak{u}_{i_1}) = 0 \bmod l_u$ and $F(\mathfrak{u}_{i_2}) = 0 \bmod l_u$ are independent, where $i_1 \neq i_2$, since the outputs of $F(\mathfrak{u}_{i_1})$ and $F(\mathfrak{u}_{i_2})$ will differ in at least one randomly chosen value. Also since the events A_i and A^* are independent for any i, we have $\Pr[\overline{A_i} | A^*] = 1/l_u$ and

$$\begin{aligned}\Pr\left[\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right] &= \Pr[A^*] \Pr\left[\bigwedge_{i=1}^{q_I} A_i | A^*\right] = \frac{1}{l_u(n_u+1)}\left(1 - \frac{q_I}{l_u}\right) \geq \frac{1}{l_u(n_u+1)}\left(1 - \frac{q_e + q_s}{l_u}\right) \\ &= \frac{1}{2(q_e+q_s)(n_u+1)}\left(1 - \frac{1}{2}\right)(\text{ by setting } l_u = 2(q_e + q_s)\ ) = \frac{1}{4(q_e+q_S)(n_u+1)}\end{aligned}$$

Using similar analysis technique for signing queries we can have: $\Pr\left[\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right] \geq \frac{1}{4q_s(n_m+1)}$

By combining the above result, we have

$$\Pr[\mathsf{not\ abort}] \geq \Pr\left[\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right) \wedge \left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)\right] \geq \frac{1}{16(q_e+q_s)(n_u+1)q_s(n_m+1)}$$

If the simulation does not abort, $\mathcal{A}$ will produce a forged signature with probability at least ϵ. Thus $\mathcal{B}$ can solve for the NGBDH problem instance with probability $\epsilon' \geq \frac{\epsilon}{16(q_e+q_s)(n_u+1)q_s(n_m+1)}$

Note that the Public-Key-Broadcast-Oracle query, Secret-Key-Extract-Oracle query and Public-Key-Replace-Oracle query will not cause the simulation abort. Thus they are excluded in the probability analysis.

Time Complexity Analysis. The time complexity of $\mathcal{B}$ is dominated by the exponentiation and multiplication operations for large values of n_u and n_m performed in the partial secret key extraction and signing queries.

There are $O(n_u)$ and $O(n_u + n_m)$ multiplications and $O(1)$ and $O(1)$ exponentiations in the partial secret key extraction and singing stage respectively. There is $O(1)$ exponentiation in the public and secret key queries. The time complexity of $\mathcal{B}$ is $t + O\Big(\big(q_e n_u + q_s(n_u + n_m)\big)\rho + (q_k + q_e + q_s)\tau\Big)$ □

Theorem 2 (Type II Existential Unforgeability). *The CL-signature scheme proposed in Section 3 is (ϵ, t)-existential unforgeable against Type II adversary (defined in Section 2) with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-Many-DH assumption*[5] *holds in $\mathbb{G}_1$, where $\epsilon' \geq \frac{\epsilon}{16(q_e+q_s)(n_u+1)q_s(n_m+1)q_k}$ and $t' = t + O\Big(\big(q_s(n_u + n_m)\big)\rho + (q_k + q_s)\tau\Big)$ where q_s is the number of queries*

[5] The definition of complexity assumptions are given in Appendix A.

made to the Signing-Oracle, q_k is the number of queries made to the Public-Key-Broadcast-Oracle and Secret-Key-Extract-Oracle altogether, and ρ and τ are the time for a multiplication and an exponentiation in $\mathbb{G}_1$ respectively.

Proof. Assume there is a Type II adversary $\mathcal{A}$ exists. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the Many-DH problem with probability at least ϵ' and in time at most t'.

$\mathcal{B}$ is given a problem instance as follow: Given a group $\mathbb{G}_1$, a generator $g \in \mathbb{G}_1$, 6 elements $g^a, g^b, g^x, g^{ab}, g^{ax}, g^{bx} \in \mathbb{G}_1$. It is asked to output an element $g^{abx} \in \mathbb{G}_1$. In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulates a challenger and all oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. Let $l_u = 2(q_e+q_s)$ and $l_m = 2q_s$. $\mathcal{B}$ randomly selects two integers k_u and k_m such that $0 \leq k_u \leq n_u$ and $0 \leq k_m \leq n_m$. Also assume that $l_u(n_u+1) < p$ and $l_m(n_m+1) < p$ for the given values of q_s, q_k, n_u and n_m. It randomly selects the following integers:

$$x' \in_R \mathbb{Z}_{l_u} \ ; \ z' \in_R \mathbb{Z}_{l_m} \ ; \ y', w' \in_R \mathbb{Z}_p \qquad \hat{x}_i \in_R \mathbb{Z}_{l_u}, \text{ for } i = 1, \ldots, n_u \qquad \hat{z}_i \in_R \mathbb{Z}_{l_m}, \text{ for } i = 1, \ldots, n_m$$
$$\hat{y}_i \in_R \mathbb{Z}_p, \text{ for } i = 1, \ldots, n_u \qquad \hat{w}_i \in_R \mathbb{Z}_p, \text{ for } i = 1, \ldots, n_m \text{ Let } \hat{X} = \{\hat{x}_i\}, \hat{Z} = \{\hat{z}_i\}, \hat{Y} = \{\hat{y}_i\}, \hat{W} = \{\hat{w}_i\}$$

We further define the following functions for binary strings $\mathfrak{u}$ and $\mathfrak{m}$ where $\mathfrak{u} = H_u(ID)$ for an identity ID and $\mathfrak{m} = H_m(m)$ for a message m, as follow:

$$F(\mathfrak{u}) = x' + \sum_{i \in \mathcal{U}_j} \hat{x}_i - l_u k_u \qquad J(\mathfrak{u}) = y' + \sum_{i \in \mathcal{U}_j} \hat{y}_i \qquad K(\mathfrak{m}) = z' + \sum_{i \in \mathcal{M}} \hat{z}_i - l_m k_m \qquad L(\mathfrak{m}) = w' + \sum_{i \in \mathcal{M}} \hat{w}_i$$

$\mathcal{B}$ constructs a set of public parameters as follow:

$$g_1 = g^a, \qquad g_2 = g^b, \qquad (g_2)^\alpha = g^{ab}, \qquad \mathsf{pk}^{(1)}_{\mathsf{ID}^*} = g^x, \qquad \mathsf{pk}^{(2)}_{\mathsf{ID}^*} = g^{ax}, \qquad u' = g_2^{-l_u k_u + x'} g^{y'}$$
$$\hat{u}_i = g_2^{\hat{x}_i} g^{\hat{y}_i} \text{ for } 1 \leq i \leq n_u, \qquad m' = g_2^{-l_m k_m + z'} g^{w'}, \qquad \hat{m}_i = g_2^{\hat{z}_i} g^{\hat{w}_i} \text{ for } 1 \leq i \leq n_m$$

for a randomly chosen identity ID^*, and we have the following equations:

$$U = u' \prod_{i \in \mathcal{U}} \hat{u}_i = g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})} \qquad \text{and} \qquad m' \prod_{i \in \mathcal{M}} \hat{m}_i = g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})}$$

All public parameters and master secret $g_2^\alpha = g^{ab}$ are passed to $\mathcal{A}$.

Oracles Simulation. $\mathcal{B}$ simulates all oracles as follow:

(Public-Key-Broadcast-Oracle.) $\mathcal{B}$ keeps the database DB of user secret-public key. It first put the public key of the identity ID^* into DB. Upon receiving a query for a public key of an identity ID, $\mathcal{B}$ looks up its database DB to find out the corresponding entry. If it does not exits, $\mathcal{B}$ runs **User-Key-Generation** to generate a secret and public key pair. It stores the key pair in its database and returns the public key as the query output.

(Secret-Key-Extract-Oracle.) Upon receiving a query for a public key of an identity ID, $\mathcal{B}$ looks up its database DB to find out the corresponding entry. If it does not exits, $\mathcal{B}$ runs **User-Key-Generation** to generate a secret and public key pair. It stores the key pair in its database and returns the secret key as the query output. If the secret key of identity ID^* is queries, it just aborts.

(Signing-Oracle.) For a given query of a signature on an identity ID and a message m, $\mathcal{B}$ first checks if the identity is equal to ID^*. If yes, it computes the signature in the following way. Assume $K(\mathfrak{m}) \neq 0 \bmod l_m$. Using the argument mentioned above, it implies $K(\mathfrak{m}) \neq 0 \bmod p$ provided that $l_m(n_m+1) < p$. The signature can be constructed by first randomly selecting $r_\pi, r_\mathfrak{m} \in_R \mathbb{Z}_p$, and computing

$$\sigma = \left((U)^{r_\pi} \left(\mathsf{pk}^{(2)}_{\mathsf{ID}^*}\right)^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}} \left(m' \prod_{i \in \mathcal{M}} \hat{m}_i \right)^{r_\mathfrak{m}}, g^{r_\pi}, \left(\mathsf{pk}^{(2)}_{\mathsf{ID}^*}\right)^{-\frac{1}{K(\mathfrak{m})}} g^{r_\mathfrak{m}} \right)$$
$$= \left(g_2^{ax} (U)^{r_\pi} \left(m' \prod_{i \in \mathcal{M}} \hat{m}_i \right)^{\tilde{r}_m}, g^{r_\pi}, g^{\tilde{r}_m} \right) = (V, R_\pi, R_m)$$

where $\tilde{r}_m = r_{\mathfrak{m}} - \frac{ax}{K(\mathfrak{m})}$. If $K(\mathfrak{m}) = 0 \bmod l_m$, the simulator aborts.

The correctness can be shown as follow:

$$V = \big(U\big)^{r_\pi}\big(\mathsf{pk}^{(2)}\big)^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}}\Big(m'\prod_{i\in\mathcal{M}}\hat{m}_i\Big)^{r_m} = \big(U\big)^{r_\pi} g_1^{-\frac{L(\mathfrak{m})x}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{r_m} = \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{r_m}$$
$$= \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\frac{ax}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{-\frac{ax}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{r_m}$$
$$= \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}} g^{abx} g^{\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\tilde{r}_m} = \big(U\big)^{r_\pi} g^{abx}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\tilde{r}_m}$$
$$= \big(U\big)^{r_\pi} g_2^{ax}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\tilde{r}_m} = \big(U\big)^{r_\pi}\big(g_2^a\big)^x\Big(m'\prod_{i\in\mathcal{M}}\hat{m}_i\Big)^{\tilde{r}_m}$$

and $R_m = \big(\mathsf{pk}^{(2)}_{\mathsf{ID}^*}\big)^{-\frac{1}{K(\mathfrak{m})}} g^{r_m} = g^{r_m - \frac{ax}{K(\mathfrak{m})}} = g^{\tilde{r}_m}$. The signature generated in this way is indistinguishable to the real one.

If it is not equal to ID^*, it computes $\mathfrak{u} = H_u(ID)$ and $\mathfrak{m} = H_m(m)$.

If $F(\mathfrak{u}) \neq 0 \bmod l_u$, $\mathcal{B}$ just construct a partial-secret key as in the Partial-Secret-Key-Extract-Oracle in the proof of Type I Adversary, then it checks from DB whether the secret key of ID has been created or not. If it has not been created, run the **User-Key-Generation** algorithm and stores the secret / public key pair in DB. If it has been created, it just use the **Sign** algorithm to create a signature on ID and m.

If $F(\mathfrak{u}) = 0 \bmod l_u$, $\mathcal{B}$ tries to construct the signature in a similar way as above (the case that the public key has been replaced). Assume $K(\mathfrak{m}) \neq 0 \bmod l_m$. Using the argument mentioned above, it implies $K(\mathfrak{m}) \neq 0 \bmod p$ provided that $l_m(n_m + 1) < p$. The signature can be constructed by first randomly selecting $r_\pi, r_{\mathfrak{m}} \in_R \mathbb{Z}_p$, getting the secret key x from DB (if it has not been created, run **User-Key-Generation** algorithm first) and computing

$$\sigma = \left(\big(U\big)^{r_\pi} g_1^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}x}\Big(m'\prod_{i\in\mathcal{M}}\hat{m}_i\Big)^{r_{\mathfrak{m}}x}, g^{r_\pi}, g_1^{-\frac{x}{K(\mathfrak{m})}} g^{r_{\mathfrak{m}}x}\right) = \left(g_2^{ax}\big(U\big)^{r_\pi}\Big(m'\prod_{i\in\mathcal{M}}\hat{m}_i\Big)^{\tilde{r}_m}, g^{r_\pi}, g^{\tilde{r}_m}\right)$$

where $\tilde{r}_m = r_{\mathfrak{m}}x - \frac{a}{K(\mathfrak{m})}x$. If $K(\mathfrak{m}) = 0 \bmod l_m$, the simulator aborts.

Output Calculation. If $\mathcal{B}$ does not abort, $\mathcal{A}$ will return an identity ID^* and a message m^* with a forged signature $\sigma^* = (V, R_\pi, R_{\mathfrak{m}})$ on ID^*, the current public key $\mathsf{pk}_{\mathsf{ID}^*}$ and m^* with probability at least ϵ. $\mathcal{B}$ checks whether the following conditions are fulfilled:

1. $F(\mathfrak{u}^*) = 0 \bmod p$, where $\mathfrak{u}^* = H_u(ID^*)$.
2. $K(\mathfrak{m}^*) = 0 \bmod p$, where $\mathfrak{m}^* = H_m(m^*)$.

If not all the above conditions are fulfilled, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ computes and outputs

$$\frac{V}{R_\pi^{J(\mathfrak{u}^*)} R_{\mathfrak{m}}^{L(\mathfrak{m}^*)}} = \frac{g_2^{ax}\big(U\big)^{r_\pi}\Big(m'\prod_{i\in\mathcal{M}}\hat{m}_i\Big)^{r_{\mathfrak{m}}}}{g^{J(\mathfrak{u}^*)r_\pi} g^{L(\mathfrak{m}^*)r_{\mathfrak{m}}}} = \frac{g_2^{ax}\Big(g_2^{F(\mathfrak{u}^*)}g^{J(\mathfrak{u}^*)}\Big)^{r_\pi}\Big(g_2^{K(\mathfrak{m}^*)}g^{L(\mathfrak{m}^*)}\Big)^{r_{\mathfrak{m}}}}{g^{J(\mathfrak{u}^*)r_\pi} g^{L(\mathfrak{m}^*)r_{\mathfrak{m}}}} = g_2^{ax} = g^{abx}$$

$\mathcal{B}$ outputs g^{abx} as the solution to the Many-DH problem instance.

Probability Analysis. For the simulation to complete without aborting, we require the following conditions fulfilled:

1. Signing-Oracle queries (ID, m) will either have $F(\mathfrak{u}) \neq 0 \bmod l_u$, or $K(\mathfrak{m}) \neq 0 \bmod l_m$ where $\mathfrak{m} = H_m(m)$, if $ID \neq ID^*$. Otherwise, it requires $K(\mathfrak{m}) \neq 0 \bmod l_m$.
2. $F(\mathfrak{u}^*) = 0 \bmod l_u$ and $K(\mathfrak{m}^*) = 0 \bmod l_m$.

In addition, in order to get the desired result, it is required that $\mathcal{A}$ has chosen ID^* for the signature forgery.

To make the analysis more simple, we will bound the probability of a subcase of this event.

Let $\mathfrak{u}_1, \ldots, \mathfrak{u}_{q_I}$ be the output of the hash function H_u appearing in Signing-Oracle queries not involving any of the challenge identity ID^*, and let $\mathfrak{m}_1, \ldots, \mathfrak{m}_{q_M}$ be the output of the hash function H_m in the sign

queries involving the challenge list. We have $q_I \leq q_s \leq q_e + q_s$ and $q_M \leq q_s$. We also define the events A_i, A^*, B_ℓ, B^* as follow:

$$A_i : F(\mathfrak{u}_i) \neq 0 \bmod l_u \qquad \text{where } i = 1, \ldots, q_I \qquad A^* : F(\mathfrak{u}^*) = 0 \bmod p$$
$$B_\ell : K(\mathfrak{m}_\ell) \neq 0 \bmod l_m \qquad \text{where } \ell = 1, \ldots, q_M \qquad B^* : K(\mathfrak{m}^*) = 0 \bmod p$$

The probability of $\mathcal{B}$ not aborting is: $\Pr[\mathsf{not\ abort}] \geq \Pr\left[\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right) \wedge \left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)\right]$. Note that the events $\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right)$ and $\left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)$ are independent.

The assumption $l_u(n_u + 1) < p$ implies if $F(\mathfrak{u} = 0 \bmod p$ then $F(\mathfrak{u}) = 0 \bmod l_u$. In addition, it also implies that if $F(\mathfrak{u}) = 0 \bmod l_u$, there will be a unique choice of k_u with $0 \leq k_u \leq n_u$ such that $F(\mathfrak{u}) = 0 \bmod p$. Since k_u, x' and $\hat{X}$ are randomly chosen,

$$\begin{aligned}\Pr[A^*] &= \Pr[F(\mathfrak{u}^*) = 0 \bmod p \wedge F(\mathfrak{u}^*) = 0 \bmod l_u] \\ &= \Pr[F(\mathfrak{u}^*) = 0 \bmod l_u] \Pr[F(\mathfrak{u}^*) = 0 \bmod p \mid F(\mathfrak{u}^*) = 0 \bmod l_u] = \frac{1}{l_u}\frac{1}{n_u + 1}\end{aligned}$$

On the other hand, we have: $\Pr\left[\bigwedge_{i=1}^{q_I} A_i | A^*\right] = 1 - \Pr\left[\bigvee_{i=1}^{q_I} \overline{A_i} \mid A^*\right] \geq 1 - \sum_{i=1}^{q_I} \Pr[\overline{A_i} \mid A^*]$ where $\overline{A_i}$ denote the event $F(\mathfrak{u}_i) = 0 \bmod l_u$.

Also note that the events $F(\mathfrak{u}_{i_1}) = 0 \bmod l_u$ and $F(\mathfrak{u}_{i_2}) = 0 \bmod l_u$ are independent, where $i_1 \neq i_2$, since the outputs of $F(\mathfrak{u}_{i_1})$ and $F(\mathfrak{u}_{i_2})$ will differ in at least one randomly chosen value. Also since the events A_i and A^* are independent for any i, we have $\Pr[\overline{A_i}|A^*] = 1/l_u$ and

$$\begin{aligned}\Pr\left[\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right] &= \Pr[A^*] \Pr\left[\bigwedge_{i=1}^{q_I} A_i | A^*\right] = \frac{1}{l_u(n_u+1)}\left(1 - \frac{q_I}{l_u}\right) \geq \frac{1}{l_u(n_u+1)}\left(1 - \frac{q_e + q_s}{l_u}\right) \\ &= \frac{1}{2(q_e+q_s)(n_u+1)}\left(1 - \frac{1}{2}\right) (\text{ by setting } l_u = 2(q_e + q_s)\ ) = \frac{1}{4(q_e+q_S)(n_u+1)}\end{aligned}$$

Using similar analysis technique for signing queries we can have: $\Pr\left[\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right] \geq \frac{1}{4q_s(n_m+1)}$. By combining the above result, we have

$$\Pr[\mathsf{not\ abort}] \geq \Pr\left[\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right) \wedge \left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)\right] \geq \frac{1}{16(q_e+q_s)(n_u+1)q_s(n_m+1)}$$

If the simulation does not abort, $\mathcal{A}$ will produce a forged signature with probability at least ϵ. In addition, $\mathcal{B}$ needs to guess which identity $\mathcal{A}$ is going to forge the signature, and assign the problem instance element as the public key of this identity. The probability of guessing correctly is $1/q_k$. Thus $\mathcal{B}$ can solve for the Many-DH problem instance with probability $\epsilon' \geq \frac{\epsilon}{16(q_e+q_s)(n_u+1)q_s(n_m+1)q_k}$

Time Complexity Analysis. It is similar to the proof of Type I Adversary except the removal of the partial secret key extract query in Type II Adversary. We skill here.

□

4 A Certificateless Encryption Scheme in the Standard Model

4.1 Building Blocks

The following two building blocks are needed to construct the certificateless encryption scheme in the standard model. (One-time signature with strong unforgeability can also be used, but it is less efficient).

Message Authentication. One of the building blocks of our system is Message Authentication scheme. Following the notions in [5], a message authentication code is a pair of PPT algorithms (Mac, Vrfy) such that Mac takes as input a key sk and a message m to produce a tag tag. The algorithm Vrfy takes as input a key sk, a message m and tag and outputs either `accept` or `reject`. It is required that for all sk and m, $\mathsf{Vrfy}(sk, m, \mathsf{Mac}(sk, m)) = \mathtt{accept}$. Loosely speaking, ($\mathsf{Mac}$, Vrfy) is secure against *one-time chosen-message attack* if no adversary can produce tag', m' such that the following holds:

– The adversary chooses a message m, and is given tag such that $\mathsf{Vrfy}(sk, m, tag) = \texttt{accept}$ for a randomly selected key sk unknown to adversary.
– $\mathsf{Vrfy}(sk, m', tag') = \texttt{accept}$.
– $m \neq m'$ or $tag \neq tag'$.

Encapsulation. Another building block of our system is an Encapsulation scheme, introduced in [5]. Roughly speaking, it is a weak variant of commitment and is defined by a triple of PPT algorithms (Init,S,R) as follow. On input security parameter k', Init outputs some public parameters pub. On input k' and pub, S outputs com, dec on some appropriate range and a string $r \in \{0,1\}^{k'}$. On input pub, com and dec, R outputs r. It is required that for all pub output by Init and for all (r, com, dec) output by $\mathsf{S}(k', pub)$, we have $\mathsf{R}(pub, com, dec) = r$. In addition, an encapsulation scheme must satisfy *binding* and *hiding*. Informally speaking, binding means that an honestly generated com can be opened to a single value of r only while hiding means that even given pub and com, the string r should be indistinguishable from random. Very efficient construction (based only on hash function) is given in [5].

4.2 Construction

Our scheme modifies from Waters identity-based encryption scheme [25], although in a non-trivial way, as follow. Let $H_u : \{0,1\}^* \to \{0,1\}^n$ be a collision-resistant cryptographic hash function for some $n \in \mathbb{Z}$. It is used to create identities of the desired length.

Setup. Select a pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ where the order of $\mathbb{G}_1$ is p. Let g be a generator of $\mathbb{G}_1$. Randomly select $\alpha \in_R \mathbb{Z}_p$, $g_2 \in_R \mathbb{G}_1$ and compute $g_1 = g^\alpha$. Compute $pub = \mathsf{Init}(k')$. Also randomly select $u', g_1', h_1 \in_R \mathbb{G}_1$ and $\hat{u}_i \in_R \mathbb{G}_1$ for $i = 1, \ldots, n$. Let $\hat{U} = \{\hat{u}_i\}$. The public parameters param are $(e, \mathbb{G}_1, \mathbb{G}_2, g, g_1, g_2, u', g_1', h_1, \hat{U}, pub)$ and the master secret key is g_2^α.

Partial-Secret-Key (PSK) Extract. Let $\mathfrak{u} = H_u(ID)$ for user with identity ID. Let $\mathfrak{u}[i]$ be the i-th bit of $\mathfrak{u}$. Define $\mathcal{U} \subset \{1, \ldots, n\}$ to be the set of indices such that $\mathfrak{u}[i] = 1$.

To construct the PSK of identity ID, the master randomly selects $r_u \in_R \mathbb{Z}_p$ and compute

$$\left(g_2^\alpha (U)^{r_u}, g^{r_u}\right) = (\mathsf{psk}^{(1)}, \mathsf{psk}^{(2)}) \qquad \text{where } U = u' \prod_{i \in \mathcal{U}} \hat{u}_i$$

User-Key-Generation. User selects a secret value $x \in \mathbb{Z}_p$ as his secret key sk and computes his public key as $(g^x, g_1^x) = (\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)})$

Encrypt. To encrypt a message $m \in \{0,1\}^{\kappa'}$, where $\kappa' = \lfloor \frac{\kappa - 1}{2} \rfloor$ and κ is the number of bit representing $\mathbb{G}_2$, for an identity ID and public key $(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)})$, first check whether $\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)} \in \mathbb{G}_1$ and $e(\mathsf{pk}^{(1)}, g_1) = e(\mathsf{pk}^{(2)}, g)$. If not, output reject and abort encryption. Otherwise, run $\mathsf{S}(pub)$ to obtain $(r \in \{0,1\}^{k'}, com \in \mathbb{Z}_p, dec \in \{0,1\}^{\kappa'})$ and set $M = m || dec$. Randomly select $t \in_R \mathbb{Z}_p$, compute $U = u' \prod_{i \in \mathcal{U}} \hat{u}_i$ and

$$C_1 = e(\mathsf{pk}^{(2)}, g_2)^t M \qquad C_2 = g^t \qquad C_3 = U^t \qquad C_4 = (g_1'^{com} h_1)^t$$

Let $\hat{C} = (C_1, C_2, C_3, C_4)$. Compute $tag = \mathsf{Mac}(r, \hat{C})$. The ciphertext $\texttt{CTXT} = (\hat{C}, com, tag)$.

Decrypt. On receiving CTXT, compute $M = C_1 \left(\frac{e(\mathsf{psk}^{(2)}, C_3) e(g, C_4)}{e(\mathsf{psk}^{(1)} g_1'^{com} h_1, C_2)} \right)^{\mathsf{sk}}$ and obtain m and dec. Compute $r = \mathsf{R}(pub, com, dec)$. If $\mathsf{Vrfy}(r, \hat{C}, tag) = \texttt{accept}$, then the plaintext is m, else output invalid ciphertext.

4.3 Security Analysis

Correctness.

$$\begin{aligned} C_1 \left(\frac{e(\mathsf{psk}^{(2)}, C_3) e(g, C_4)}{e(\mathsf{psk}^{(1)} g_1'^{com} h_1, C_2)} \right)^{\mathsf{sk}} &= e(g_1^x, g_2)^t M \left(\frac{e(g^{r_u}, U^t)^x e(g, (g_1'^{com} h_1)^{tx})}{e(g_2^\alpha U^{r_u} g_1'^{com} h_1, g^t)^x} \right) \\ &= e(g_1^x, g_2)^t M \left(\frac{e(g, U^{t r_u x}) e(g, (g_1'^{com} h_1)^{tx})}{e(g, g_2^\alpha U^{r_u} g_1'^{com} h_1)^{tx}} \right) \end{aligned}$$

$$= e(g_1^x, g_2)^t M \left(\frac{e(g, U^{tr_u x} (g_1'^{com} h_1)^{tx})}{e(g, g_2^\alpha)^{tx} e(g, U^{r_u} g_1'^{com} h_1)^{tx}} \right) = e(g_1^x, g_2)^t M \frac{1}{e(g^{\alpha x}, g_2)^t}$$
$$= e(g_1^x, g_2)^t M \frac{1}{e(g_1^x, g_2)^t} = M$$

Theorem 3 (Type I$^-$ IND-CCA$^-$ Secure). *The CL-encryption scheme proposed in Section 4 is $(\tilde{t}, q, \epsilon)$ secure against Type I$^-$ adversary (defined in Section 2) with advantage at most ϵ, runs in time at most $\tilde{t}$ and making at most q Partial-Secret-Key-Extract-Oracle queries, assuming that the $(\epsilon', \tilde{t}')$-DBDH-1 assumption[6] holds in $\mathbb{G}_1$, where $\epsilon' \geq \frac{\epsilon}{16(n+1)q}$ and $\tilde{t}' = \tilde{t} + O\Big(\big(qn + q_d n\big)\rho + (q_k + q + q_d)\tau\Big)$ where q_d is the number of queries made to the Decryption-Oracle, q_k is the number of queries made to the Public-Key-Broadcast-Oracle and Secret-Key-Extract-Oracle altogether, and ρ and τ are the time for a multiplication and an exponentiation in $\mathbb{G}_1$ respectively.*

Proof. Assume there exists a $(\tilde{t}, q, \epsilon)$ type I$^-$ adversary $\mathcal{A}$ against our scheme. We construct a PPT simulator $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the DBDH-1 problem with probability at least ϵ' and in time at most $\tilde{t}'$. $\mathcal{B}$ is given a DBDH-1 problem instance $(g, A = g^a, B = g^b, C = g^c, Z)$ and is to decide if $Z = g^{abc}$. In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulates a challenger and the oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. On input a security parameter k', $\mathcal{B}$ computes $pub = \mathsf{Init}(k')$. $\mathcal{B}$ sets an integer $m' = 4q$ and chooses an integer $k \in_R \{0, \cdots, n\}$. It also chooses a random n-length vector, $\boldsymbol{x} = (x_i)$, and a value x' such that each x_i and x' are random integers between 0 and $m' - 1$. Let X^* denote $(x', \boldsymbol{x})$. It also chooses an n-length vector, $\boldsymbol{y} = (y_i)$, together with a value y' such that each y_i and y' are random elements in $\mathbb{Z}_p$. $\mathcal{B}$ runs $\mathsf{S}(pub)$ and obtain r^*, com^*, dec^*. $\mathcal{B}$ constructs a set of public parameters as follow:

$$g_1 = g^a, \qquad g_2 = g^b \qquad u' = g_2^{p - m'k + x'} g^{y'}, \qquad u_i = g_2^{x_i} g^{y_i} \quad \text{for} \quad 1 \leq i \leq n$$
$$g_1' = g_1^{\alpha_2}, \qquad h_1 = g_1'^{-com^*} g^{\alpha_1} \quad \text{for randomly generated} \quad \alpha_1, \alpha_2 \in_R \mathbb{Z}_p$$

Note that the master secret will be $g_2^\alpha = g_2^a = g^{ab}$ which is unknown to $\mathcal{B}$. All public parameters are passed to $\mathcal{A}$.

For the ease of presentation, for identity $\mathfrak{u}$, define $\mathcal{U} \subset \{1, \ldots, n\}$ to be the set of indicies such that $\mathfrak{u}[i] = 1$. Also define $F(\mathfrak{u}) = (p - m'k) + x' + \sum_{i \in \mathcal{U}} x_i$ and $J(\mathfrak{u}) = y' + \sum_{i \in \mathcal{U}} y_i$. Next we define $K(\mathfrak{u}) = x' + \sum_{i \in \mathcal{U}} x_i \bmod m'$. Also, define $G(t) = g_1'^{t - com^*} g^{\alpha_1}$.

Phase 1. $\mathcal{A}$ can issue partial secret key extract, secret key extract and decryption queries.

(Public-Key-Broadcast-Oracle.) $\mathcal{B}$ keeps the database of user secret-public key. Upon receiving a query for a public key of an identity ID, $\mathcal{B}$ looks up its database to find out the corresponding entry. If it does not exits, $\mathcal{B}$ runs **User-Key-Generation** to generate a secret and public key pair. It stores the key pair in its database and returns the public key as the query output.

(Partial-Secret-Key-Extract-Oracle.) Upon receiving a query for a private secret of an identity ID_u, B compute $\mathfrak{u} = H_u(ID_u)$. Although $\mathcal{B}$ does not know the master secret, it still can construct the private key by assuming $K(\mathfrak{u}) \neq 0$. If $K(\mathfrak{u}) = 0$, $\mathcal{B}$ aborts. If $\mathcal{B}$ does not abort, it randomly chooses $r_u \in_R \mathbb{Z}_p$ and computes the private key as: $d_u = (\mathsf{psk}^{(1)}, \mathsf{psk}^{(2)}) = \left(g_1^{-\frac{J(\mathfrak{u})}{F(\mathfrak{u})}} \big(U\big)^{r_u}, g_1^{-\frac{1}{F(\mathfrak{u})}} g^{r_u} \right)$

By letting $\tilde{r}_u = r_u - \frac{a}{F(\mathfrak{u})}$, it can be verifier that d_u is a valid partial secret, shown as follow:

$$\begin{aligned}
\mathsf{psk}^{(1)} &= g_1^{-\frac{J(\mathfrak{u})}{F(\mathfrak{u})}} \big(U\big)^{r_u} = g_1^{-\frac{J(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u} = g^{-\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u} \\
&= g^{-\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\frac{a}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{-\frac{a}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u} \\
&= g^{-\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} g^{ab} g^{\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\tilde{r}_u} = g^{ab} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\tilde{r}_u} = g_2^a (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\tilde{r}_u} = g_2^a \big(U\big)^{\tilde{r}_u}
\end{aligned}$$

and $\mathsf{psk}^{(2)} = g_1^{-\frac{1}{F(\mathfrak{u})}} g^{r_u} = g^{r_u - \frac{a}{F(\mathfrak{u})}} = g^{\tilde{r}_u}$

[6] The definition of complexity assumptions are given in Appendix A.

(Secret-Key-Extract-Oracle.) For any given identity, the public key is generated by $\mathcal{B}$ and thus it can return to secret key to $\mathcal{A}$. If ID^* is to be corrupted, $\mathcal{B}$ aborts. If the corresponding public key has been replaced, $\mathcal{B}$ outputs $\perp$.

(Decryption-Oracle.) Since $\mathcal{A}$ is a type I^- adversary, it can only issue decryption query either to identities where it has not replaced the public keys or the corresponding secret keys are supplied. We can assume that $\mathcal{B}$ knows the discrete logarithm of $\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}$ in the decryption query since it is either generated by $\mathcal{B}$ (in the first case) or supplied by $\mathcal{A}$ (in the second case). Suppose the query is (ID, g^x, g_1^x) with $\mathtt{CTXT} = \{\hat{C}, com, tag\}$ such that $\hat{C} = \{C_1, C_2, C_3, C_4\}$. If $com = com^*$, $\mathcal{B}$ output invalid ciphertext. Otherwise, denote $\hat{t} = com - com^*$ and assume $\mathfrak{u} = H_u(ID)$. $\mathcal{B}$ then randomly chooses $r_0, r_1 \in_R \mathbb{Z}_p$ and computes $d_0 = g_2^{-\alpha_1/(\hat{t}\alpha_2)}\big(g_1'^{\hat{t}} g^{\alpha_1}\big)^{r_1}\big(U\big)^{r_0}$, $d_1 = g^{r_0}$, $d_2 = g_2^{-1/(\hat{t}\alpha_2)} g^{r_1}$. Similar to above, let $\tilde{r_1} = r_1 - b/\hat{t}\alpha_2$. We have

$$
\begin{aligned}
d_0 &= g_2^{-\frac{\alpha_1}{\hat{t}\alpha_2}}(g_1'^{\hat{t}} g^{\alpha_1})^{r_1}(U)^{r_0} = g^{-\frac{b\alpha_1}{\hat{t}\alpha_2}}(g_1'^{com-com^*} g^{\alpha_1})^{r_1}(U)^{r_0} = g^{\alpha_1(\tilde{r}_1 - r_1)}(g_1'^{com-com^*} g^{\alpha_1})^{\tilde{r}_1 + \frac{b}{\hat{t}\alpha_2}}(U)^{r_0} \\
&= g^{\alpha_1(\tilde{r}_1 - r_1)} G(com)^{\tilde{r}_1}(g^{\frac{a\alpha_2\hat{t}b}{\hat{t}\alpha_2}} g^{\frac{\alpha_1 b}{\hat{t}\alpha_2}})(U)^{r_0} = g^{\alpha_1(\tilde{r}_1 - r_1)}(g^{ab} g^{\frac{\alpha_1 b}{\hat{t}\alpha_2}}) G(com)^{\tilde{r}_1}(U)^{r_0} \\
&= g^{\alpha_1\tilde{r}_1 - \alpha_1 r_1 + ab + \frac{\alpha_1 b}{\hat{t}\alpha_2}} G(com)^{\tilde{r}_1}(U)^{r_0} = g^{\alpha_1\tilde{r}_1 - \alpha_1(r_1 - \frac{b}{\hat{t}\alpha_2}) + ab} G(com)^{\tilde{r}_1}(U)^{r_0} \\
&= g^{\alpha_1\tilde{r}_1 - \alpha_1\tilde{r}_1} g^{ab} G(com)^{\tilde{r}_1}(U)^{r_0} = g_2^a\big(U\big)^{r_0} G(com)^{\tilde{r_1}}
\end{aligned}
$$

and $d_2 = g_2^{-\frac{1}{\hat{t}\alpha_2}} g^{r_1} = g^{-\frac{b}{\hat{t}\alpha_2} + r_1} = g^{\tilde{r}_1}$. That is, (d_0, d_1, d_2) is of the form $(g_2^a\big(U\big)^{r_0} G(com)^{\tilde{r_1}}, g^{r_0}, g^{\tilde{r_1}})$.

$\mathcal{B}$ computes

$$
C_1\Big(\frac{e(d_1, C_3)e(d_2, C_4)}{e(d_0, C_2)}\Big)^x = e(g_1^x, g_2)^t M\left(\frac{e(g^{r_0}, U^t)e(g^{\tilde{r}_1}, (g_1'^{com} h_1)^t)}{e\Big(g_2^a (U)^{r_0} G(com)^{\tilde{r}_1}, g^t\Big)}\right)^x
$$

$$
= e(g,g)^{abxt} M\left(\frac{e(g, U)^{r_0 tx} e(g, g_1'^{com} h_1)^{\tilde{r}_1 tx}}{e(g,g)^{abxt} e(U, g)^{r_0 tx} e(g, g_1'^{com-com^*} g^{\alpha_1})^{\tilde{r}_1 tx}}\right) = M\left(\frac{e(g, g_1'^{com} h_1)^{\tilde{r}_1 tx}}{e(g, g_1'^{com} h_1)^{\tilde{r}_1 tx}}\right) = M
$$

and obtain m, dec. It then computes $r = \mathsf{R}(pub, com, dec)$. If $\mathsf{Vrfy}(r, \hat{C}, tag) = \mathtt{accept}$, then output m, else output `invalid ciphertext`.

(Public-Key-Replace-Oracle.) Upon receiving a query for a public key replace oracle request of an identity ID, $\mathcal{B}$ looks up its database to replace the corresponding entry. If it does not exits, $\mathcal{B}$ creats a new entry for this identity.

Challenge. $\mathcal{A}$ then outputs two messages m_0, m_1, a challenge identity ID^* with public key $\mathsf{pk}^{(1)*}, \mathsf{pk}^{(2)*}$ which may have been replaced. Let $H_u(ID^*) = \mathfrak{u}^*$. If $x' + \sum_{i \in \mathcal{U}^*} x_i \neq m'k$, the $\mathcal{B}$ aborts. Otherwise it flips a fair coin, $\gamma \in \{0, 1\}$, and constructs the ciphertext as follows. $M_\gamma = m_\gamma || dec^*$, $\hat{C}^* = \{e(Z, \mathsf{pk}^{(1)*}) M_\gamma, g^c, (g^c)^{J(\mathfrak{u}^*)}, (g^c)^{\alpha_1}\}$. Compute $tag^* = \mathsf{Mac}(r^*, \hat{C}^*)$ and return $\mathtt{CTXT}^* = \{\hat{C}^*, com^*, tag^*\}$. $\mathtt{CTXT}^*$ is a valid encryption of m_γ if Z is a BDH tuple. Otherwise, $\mathtt{CTXT}^*$ will give no information about $\mathcal{B}$'s choice of γ.

Phase 2. $\mathcal{B}$ repeats the same method it used in Phase 1.

Guess. Finally, $\mathcal{A}$ output a guess γ' of γ. If $\gamma' = \gamma$, then $\mathcal{B}$ output 1 and else it outputs 0.

Probability Analysis. For the simulation to be perfect, we require the following conditions fulfilled:

1. All partial secret extraction queries on an identity ID have $K(\mathfrak{u}) \neq 0$, where $\mathfrak{u} = H_u(ID)$.
2. $\mathcal{B}$ does not reject valid ciphertext during decryption oracle queries.
3. $x' + \sum_{i \in \mathcal{U}^*} x_i = m'k$ for the challenge identity ID^* such that $\mathfrak{u}^* = H_u(ID^*)$.

In order to make the analysis more simple, we will bound the probability of a subcase of this event. Event 2 happens with negligible probability assume the underlying commitment and message authentication scheme are secure. Thus, for any set of q queries on $\mathfrak{u}_i$ and challenge $\mathfrak{u}^*$ such that $\mathfrak{u}_i = H_u(ID_i)$ and $\mathfrak{u}^* = H_u(ID^*)$, $Pr[\overline{abort}] = Pr[(\bigwedge K(\mathfrak{u}_i) \neq 0) \wedge K(\mathfrak{u}^*) = 0]$. The lower bound of this probability is $\lambda = \frac{1}{8(n+1)q}$, as follows.

$$
\begin{aligned}
\Pr[\overline{abort}] &= \Pr\left[\bigwedge K(\mathfrak{u}_i) \neq 0 \wedge x' + \sum_{i \in \mathcal{U}^*} x_i = m'k\right] \\
&= \left(1 - \Pr\left[\bigvee K(\mathfrak{u})i) = 0\right]\right) \Pr\left[x' + \sum_{i \in \mathcal{U}^*} x_i = m'k | \bigwedge K(\mathfrak{u}_i) \neq 0\right] \\
&\geq \left(1 - \sum \Pr\left[K(\mathfrak{u})i) = 0\right]\right) \Pr\left[x' + \sum_{i \in \mathcal{U}^*} x_i = m'k | \bigwedge K(\mathfrak{u}_i) \neq 0\right] \\
&= (1 - \frac{q}{m'}) \Pr\left[x' + \sum_{i \in \mathcal{U}^*} x_i = m'k | \bigwedge K(\mathfrak{u}_i) \neq 0\right] = \frac{1}{n+1}(1 - \frac{q}{m'}) \Pr\left[K(\mathfrak{u}^*) = 0 | \bigwedge K(\mathfrak{u}_i) \neq 0\right] \\
&= \frac{1}{n+1}(1 - \frac{q}{m'}) \frac{\Pr[K(\mathfrak{u}^*) = 0]}{\Pr[\bigwedge K(\mathfrak{u}_i) \neq 0]} \Pr\left[\bigwedge K(\mathfrak{u}_i) \neq 0 | K(\mathfrak{u}^*) = 0\right] \\
&\geq \frac{1}{(n+1)m'}(1 - \frac{q}{m'}) \Pr\left[\bigwedge K(\mathfrak{u}_i) \neq 0 | K(\mathfrak{u}^*) = 0\right] \\
&= \frac{1}{(n+1)m'}(1 - \frac{q}{m'})(1 - \Pr\left[\bigwedge K(\mathfrak{u}_i) = 0 | K(\mathfrak{u}^*) = 0\right]) \\
&\geq \frac{1}{(n+1)m'}(1 - \frac{q}{m'})(1 - \sum \Pr\left[K(\mathfrak{u}_i) = 0 | K(\mathfrak{u}^*) = 0\right]) = \frac{1}{(n+1)m'}(1 - \frac{q}{m'})^2 \\
&\geq \frac{1}{(n+1)m'}(1 - 2\frac{q}{m'}) = \frac{1}{8q(n+1)}
\end{aligned}
$$

If the simulation does not abort, success probability of $\mathcal{A}$ is analyzed as follows. Suppose (A, B, C, Z) is not a valid DBDH tuple, then the challenge ciphertext contains no valid information about the message and thus probability of the adversary winning will be $\frac{1}{2}$. Suppose (A, B, C, Z) is a valid DBDH tuple, then by our assumption, adversary makes the correct guess with probability $\frac{1}{2} + \epsilon$. Thus, advantage of $\mathcal{B}$ is at least $\frac{\epsilon}{16(n+1)q}$.

Time Complexity Analysis. The time complexity of $\mathcal{B}$ is dominated by the exponentiation and multiplication operations for large values of n performed in the partial secret key extraction and decryption queries.

There are $O(n)$ and $O(n)$ multiplications and $O(1)$ and $O(1)$ exponentiations in the partial secret key extraction and decryption stage respectively. There is $O(1)$ exponentiation in the public and secret key queries. The time complexity of $\mathcal{B}$ is $\tilde{t} + O\Big((qn + q_d n)\rho + (q_k + q + q_d)\tau\Big)$ □

We remark that if our proof is carried out in the Generic Group Model [24], it can be proven secure against Type I adversary by the following changes:

- In the Public-Key-Replace-Oracle, $\mathcal{B}$ verifies the validity of the public key, by checking whether $\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)} \in \mathbb{G}_1$ and $e(\mathsf{pk}^{(1)}, g_1) = e(\mathsf{pk}^{(2)}, g)$ which ensures $\mathcal{A}$ to carry out a group operation.
- $\mathcal{B}$ can extract the corresponding secret key from the group operation and decrypt the ciphertext for those identities whose public keys have been replaced.

Theorem 4 (Type II IND-CCA Secure). *The CL-encryption scheme proposed in Section 4 is $(\tilde{t}, q, \epsilon)$ secure against Type II adversary (defined in Section 2) with advantage at most ϵ, runs in time at most $\tilde{t}$ and making at most q Public-Key-Broadcast-Oracle queries, assuming that the $(\epsilon', \tilde{t}')$-DBDH-1 assumption holds in $\mathbb{G}_1$, where $\epsilon' \geq \frac{\epsilon}{2q}$ and $\tilde{t}' = \tilde{t} + O\Big((q_d n)\rho + (q + q_c + q_d)\tau\Big)$ where q_d is the number of queries made to the Decryption-Oracle, q_c is the number of queries made to the Secret-Key-Extract-Oracle, and ρ and τ are the time for a multiplication and an exponentiation in $\mathbb{G}_1$ respectively.*

Proof. Assume there exists a $(\tilde{t}, q, \epsilon)$ type II adversary $\mathcal{A}$ against our scheme. We construct a PPT simulator $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the DBDH-1 problem with probability at least ϵ' and in time at most $\tilde{t}'$. $\mathcal{B}$ is given a DBDH-1 problem instance $(g, A = g^a, B = g^b, C = g^c, Z)$ and is to decide if

$Z = g^{abc}$. In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulates a challenger and the oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. On input a security parameter k', $\mathcal{B}$ computes $pub = \mathsf{Init}(k')$. $\mathcal{B}$ sets an integer $m' = 4q$ and chooses an integer $k \in_R \{0, \cdots, n\}$. It also chooses a random n-length vector, $\boldsymbol{x} = (x_i)$, and a value x' such that each x_i and x' are random integers between 0 and $m'-1$. Let X^* denote $(x', \boldsymbol{x})$. It also chooses an n-length vector, $\boldsymbol{y} = (y_i)$, together with a value y' such that each y_i and y' are random elements in $\mathbb{Z}_p$. $\mathcal{B}$ runs $\mathsf{S}(pub)$ and obtain r^*, com^*, dec^*. Randomly chooses α as the master key. Randomly chooses an identity ID^* and relate it to the public key $(g^a, (g^a)^\alpha)$. $\mathcal{B}$ constructs a set of public parameters as follow:

$$g_1 = g^\alpha, \qquad g_2 = g^b, \qquad u' = g_2^{p-m'k+x'} g^{y'}, \qquad u_i = g_2^{x_i} g^{y_i} \quad \text{for} \quad 1 \le i \le n$$
$$g_1' = g^{a\alpha_2}, \qquad h_1 = g_1'^{\,-com^*} g^{\alpha_1} \quad \text{for randomly generated} \quad \alpha_1, \alpha_2 \in_R \mathbb{Z}_p$$

All public parameters, together with master key α are passed to $\mathcal{A}$.

For the ease of presentation, for identity $\mathfrak{u}$, define $\mathcal{U} \subset \{1, \ldots, n\}$ to be the set of indicies such that $\mathfrak{u}[i] = 1$. Also define $F(\mathfrak{u}) = (p - m'k) + x' + \sum_{i \in \mathcal{U}} x_i$ and $J(\mathfrak{u}) = y' + \sum_{i \in \mathcal{U}} y_i$. Next we define $K(\mathfrak{u}) = x' + \sum_{i \in \mathcal{U}} x_i \bmod m'$. Also, define $G(t) = g_1'^{\,t-com^*} g^{\alpha_1}$.

Phase 1. $\mathcal{A}$ can issue partial secret extract queries by itself and the following oracle queries.

(Public-Key-Broadcast-Oracle.) $\mathcal{B}$ keeps the database of user secret-public key. Upon receiving a query for a public key of an identity ID, $\mathcal{B}$ looks up its database to find out the corresponding entry. If it does not exits, $\mathcal{B}$ runs **User-Key-Generation** to generate a secret and public key pair. It stores the key pair in its database and returns the public key as the query output.

(Secret-Key-Extract-Oracle.) For any given identity, the public key is generated by $\mathcal{B}$ and thus it can return to secret key to $\mathcal{A}$. If ID^* is to be corrupted, $\mathcal{B}$ aborts.

(Decryption-Oracle.) For all decryption queries not involving ID^*, $\mathcal{B}$ is in possession of the user secret and can thus decrypt perfectly. If decryption query involves ID^*, $\mathcal{B}$ answers as follows. Suppose the query is (ID^*, g^a, g_1^a) with $\mathtt{CTXT} = \{\hat{C}, com, tag\}$ such that $\hat{C} = \{C_1, C_2, C_3, C_4\}$. If $com = com^*$, $\mathcal{B}$ output invalid ciphertext. Otherwise, denote $\hat{t} = com - com^*$ and assume $\mathfrak{u} = H_u(ID)$. $\mathcal{B}$ then randomly chooses $r_0, r_1 \in_R \mathbb{Z}_p$ and computes $d_0 = g_2^{-\alpha\alpha_1/(\hat{t}\alpha_2)} \big(g_1'^{\,\hat{t}} g^{\alpha_1}\big)^{r_1} \big(U\big)^{r_0}$, $d_1 = g^{r_0}$, $d_2 = g_2^{-1/(\hat{t}\alpha_2)} g^{r_1}$. Similar to above, let $\tilde{r_1} = r_1 - b/t\alpha_2$. Using the same argument of Type I$^-$ proof, it can be easily shown that (d_0, d_1, d_2) is of the form $(g_2^{a\alpha} \big(U\big)^{r_0} G(com)^{\tilde{r_1}}, g^{r_0}, g^{\tilde{r_1}})$. $\mathcal{B}$ computes

$$C_1\big(\frac{e(d_1, C_3)e(d_2, C_4)}{e(d_0, C_2)}\big) = e(g^{\alpha a}, g^b)^t M\left(\frac{e(g^{r_0}, U^t)e(g^{\tilde{r}_1}, (g_1'^{\,com} h_1)^t)}{e\big(g_2^{a\alpha}(U)^{r_0} G(com)^{\tilde{r}_1}, g^t\big)}\right)$$
$$= e(g,g)^{ab\alpha t} M\left(\frac{e(g, U)^{r_0 t} e(g, g_1'^{\,com} h_1)^{\tilde{r}_1 t}}{e(g,g)^{ab\alpha t} e(g, U)^{r_0 t} e(g, g_1'^{\,com} h_1)^{\tilde{r}_1 t}}\right) = M$$

and obtain m, dec. It then computes $r = \mathsf{R}(pub, com, dec)$. If $\mathsf{Vrfy}(r, \hat{C}, tag) = \mathtt{accept}$, then output m, else output `invalid ciphertext`.

Challenge. $\mathcal{A}$ then outputs two messages m_0, m_1, a challenge identity $\hat{ID}$ with public key $\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}$. If $\hat{ID} \neq ID^*$, $\mathcal{B}$ aborts. If it does not abort, assume $H_u(ID^*) = \mathfrak{u}^*$. If $x' + \sum_{i \in \mathcal{U}^*} x_i \neq m'k$, the $\mathcal{B}$ aborts. Otherwise it flips a fair coin, $\gamma \in \{0, 1\}$, and constructs the ciphertext as follows. $M_\gamma = m_\gamma || dec^*$, $\hat{C}^* = \{e(Z, \mathsf{pk}^{(1)*}) M_\gamma, g^c, (g^c)^{J(\mathfrak{u}^*)}, (g^c)^{\alpha_1}\}$. Compute $tag^* = \mathsf{Mac}(r^*, \hat{C}^*)$ and return $\mathtt{CTXT}^* = \{\hat{C}^*, com^*, tag^*\}$. $\mathtt{CTXT}^*$ is a valid encryption of m_γ if Z is a BDH tuple. Otherwise, $\mathtt{CTXT}^*$ will give no information about $\mathcal{B}$'s choice of γ.

Phase 2. $\mathcal{B}$ repeats the same method it used in Phase 1.

Guess. Finally, $\mathcal{A}$ output a guess γ' of γ. If $\gamma' = \gamma$, then $\mathcal{B}$ output 1 and else it outputs 0.

Probability Analysis. For the simulation not to abort, $\hat{ID} = ID^*$. This happens with probability $\frac{1}{q}$ since $\mathcal{A}$ can make q different Public-Key-Broadcast-Oracle queries. If the simulation does not abort, success probability of $\mathcal{A}$ is analyzed as follows. Suppose (A, B, C, Z) is not a valid BDH tuple, then the challenge ciphertext contains no valid information about the message and thus probability of the adversary winning

will be $\frac{1}{2}$. Suppose (A, B, C, Z) is a valid BDH tuple, then by our assumption, adversary makes the correct guess with probability $\frac{1}{2} + \epsilon$. Thus, advantage of $\mathcal{B}$ is at least $\frac{\epsilon}{2q}$.

Time Complexity Analysis. The time complexity of $\mathcal{B}$ is dominated by the exponentiation and multiplication operations for large values of n performed in the secret key extraction and decryption queries.

There are $O(0)$ and $O(n)$ multiplications and $O(1)$ and $O(1)$ exponentiations in the secret key extraction and decryption stage respectively (since the decryption oracle may need to construct the partial secret key in order to carry out the decryption, which may require up to $O(n)$ multiplications). There is $O(1)$ exponentiation in the public and secret key queries. The time complexity of $\mathcal{B}$ is $\tilde{t}+O\Big(\big(q_d n\big)\rho+(q+q_c+q_d)\tau\Big)$ □

5 A Self-Generated-Certificate Encryption Scheme

We give a generic construction of Self-Generated-Certificate (SGC) encryption scheme, building from a certificateless (CL) encryption scheme and a certificateless (CL) signature scheme that are using the same set of public parameters and key generation algorithm. (It maybe possible to use a CL-encryption and a CL-signature scheme that use different sets of public parameters and user key generation algorithms, although space and time complexity may be increased. For simplicity, we exclude this case in our paper (including the definition and construction of the scheme).)

In order to distinguish the algorithm of CL-encryption and CL-signature, we will add the prefix "CL." to the corresponding algorithms. For example, we use "**CL.Encrypt**" to denote the encryption algorithm of the underlying CL-encryption scheme. The proposed SGC-encryption scheme is described as follow:

Setup. Same as **CL.Setup**, outputs parameters param and master secret key mk.

$\mathsf{psk} \leftarrow$ **Partial-Secret-Key-Extract**$(\mathsf{param}, \mathsf{ID})$: To generate a partial sercret key with identity ID, it is defined as follow:

$$\begin{aligned}
\mathsf{CL.psk_{sign}} &\leftarrow \textbf{CL.Partial-Secret-Key-Extract}(\mathsf{param}, \mathsf{mk}, \langle \text{“SIGN”}||\mathsf{ID}\rangle)\\
\mathsf{CL.psk_{enc}} &\leftarrow \textbf{CL.Partial-Secret-Key-Extract}(\mathsf{param}, \mathsf{mk}, \langle \text{“ENC”}||\mathsf{ID}\rangle)\\
\text{Output } \mathsf{psk} &:= \langle\ \mathsf{CL.psk_{sign}} \ ||\ \mathsf{CL.psk_{enc}}\ \rangle
\end{aligned}$$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow$ **User-Key-Generation**$(\mathsf{param}, \mathsf{psk}, \mathsf{ID})$: To generate a user key with identity ID and partial secret key psk, it is defined as follow:

$$\begin{aligned}
\langle\ \mathsf{CL.psk_{sign}} \ ||\ \mathsf{CL.psk_{enc}}\ \rangle &:= \mathsf{psk}\\
(\mathsf{CL.sk}, \mathsf{CL.pk}) &\leftarrow \textbf{CL.User-Key-Generation}(\mathsf{param});\\
m' &:= \langle\ \mathsf{ID}\ ||\ \mathsf{CL.pk}\ \rangle\\
\sigma &\leftarrow \textbf{CL.Sign}(\mathsf{param}, \mathsf{CL.sk}, \mathsf{CL.psk_{sign}}, m')\\
\mathsf{pk}' &:= \langle\ \mathsf{CL.pk}\ ||\ \sigma\ \rangle\\
\text{Output } (\mathsf{sk}, \mathsf{pk}) &:= (\ \mathsf{CL.sk},\ \mathsf{pk}')
\end{aligned}$$

$C \leftarrow$ **Encrypt**$(\mathsf{param}, m, \mathsf{ID}, \mathsf{pk})$: To encrypt a message m for a user with identity ID and public key pk, it is defined as follow:

$$\begin{aligned}
\langle\ \mathsf{CL.pk}\ ||\ \sigma\ \rangle &:= \mathsf{pk}\\
m' &:= \langle\ \mathsf{ID}\ ||\ \mathsf{CL.pk}\ \rangle\\
\text{IF}\quad &\textbf{CL.Verify}(\mathsf{param}, m', \sigma, \langle \text{“SIGN”}||\mathsf{ID}\rangle, \mathsf{CL.pk}) = \perp\\
&\quad \text{Output } \perp\\
\text{ELSE}&\\
&\quad \text{Output } C \leftarrow \textbf{CL.Encrypt}(\mathsf{param}, m, \langle \text{“ENC”}||\mathsf{ID}\rangle, \mathsf{CL.pk})
\end{aligned}$$

$m/\perp \leftarrow \mathbf{Decrypt}(C, \mathsf{sk}, \mathsf{psk})$: To decrypt a ciphertext C for a user with partial secret key psk and secret key sk, it is defined as follow:

$$\begin{aligned} \mathsf{CL.sk} &:= \mathsf{sk} \\ \langle\, \mathsf{CL.psk_{sign}} \,||\, \mathsf{CL.psk_{enc}} \,\rangle &:= \mathsf{psk} \\ \text{Output } m/\perp &\leftarrow \mathbf{CL.Decrypt}(C, \mathsf{CL.sk}, \mathsf{CL.psk_{enc}}) \end{aligned}$$

It outputs a plaintext m for a valid ciphertext C, or $\perp$ otherwise.

Using our proposed CL-signature scheme in Section 3 and our proposed CL-encryption scheme in Section 4, we can build up a SGC-encryption scheme in the standard model. Security Analysis is given in Appendix B.

6 Concluding Remark

In this paper, we proposed a new paradigm, called "Self-Generated-Certificate (SGC) Public Key Cryptosystem". It is the enhanced version of Certificateless Public Key Cryptography (CL-PKC). It can defend against the DoD attack. We provided a generic construction of a SGC-encryption scheme. Besides, we also proposed a certificateless signature scheme and a certificateless encryption scheme. Both of them are proven secure in the standard model which are the first in the literature for concrete implementation. The security model of our CL-signature scheme is the strongest when compared with others in the literature.

However, we face the same problem mentioned in [9]. We can only achieve Type I^- security in the standard model although we can obtain Type I security in the generic group model [24], as stated in the proof. It is still an open problem to achieve Type I security in the standard model.

References

1. S. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In *ASIACRYPT 2003*, pages 452–473. Springer-Verlag, 2003. LNCS No. 2894.
2. S. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. Cryptology ePrint Archive, Report 2003/126, 2003. `http://eprint.iacr.org/2003/126/`.
3. J. Baek, R. Safavi-Naini, and W. Susilo. Certificateless public key encryption without pairing. In *ISC 05*, pages 134–148. Springer-Verlag, 2005. LNCS Vol. 3650.
4. K. Bentahar, P. Farshim, and J. Malone-Lee. Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058, 2005. `http://eprint.iacr.org/2005/058/`.
5. D. Boneh and J. Katz. Improved efficiency for cca-secure cryptosystems built using identity-based encryption. In *CT-RSA*, pages 87–103, 2005.
6. H. Chabanne, D. H. Phan, and D. Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 542–558. Springer, 2005.
7. Z. Cheng and R. Comley. Efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/012, 2005. `http://eprint.iacr.org/2005/012/`.
8. S. Chow, C. Boyd, and J. Gonzalez. Security-mediated certificateless cryptography. In *PKC 2006*, volume 3958 of *LNCS*, pages 508–524. Springer-Verlag, 2006.
9. A. Dent and C. Kudla. On proofs of security for certificateless cryptosystems. Cryptology ePrint Archive, Report 2005/348, 2005. `http://eprint.iacr.org/2005/348/`.
10. C. Gentry. Certificate-based encryption and the certificate revocation problem. In *EUROCRYPT 2003*, pages 272–293. Springer-Verlag, 2003. LNCS No. 2656.
11. M. Girault. Self-certified public keys. In *EUROCRYPT 91*, pages 490–497. Springer-Verlag, 1992. LNCS No. 547.
12. M. Gorantla, R. Gangishetti, M. Das, and A. Saxena. An effective certificateless signature scheme based on bilinear pairings. In *WOSIS 2005*, pages 31–39. INSTICC Press, 2005.
13. B. Hu, D. Wong, Z. Zhang, and X. Deng. Key replacement attack against a generic construction of certificateless signature. In *ACISP '06*, pages 235–246. Springer-Verlag, 2006. LNCS No. 4058.
14. X. Huang, W. Susilo, Y. Mu, and F. Zhang. On the security of certificateless signature schemes from Asiacrypt 2003. In *CANS 2005*, pages 13–25. Springer-Verlag, 2005. LNCS No. 3810.
15. X. Huang, W. Susilo, Y. Mu, and F. Zhang. Certificateless designated verifier signature schemes. In *AINA 2006*, pages 15–19. IEEE Computer Society, 2006.

16. B. Libert and J. Quisquater. On constructing certificateless cryptosystems from identity based encryption. In *PKC 2006*, pages 474–490. Springer-Verlag, 2006. LNCS No. 3958.
17. A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *CRYPTO 2004*, volume 2442 of *LNCS*, pages 597–612. Springer, 2002.
18. K. Paterson and J. Schuldt. Efficient identity-based signatures secure in the standard model. Cryptology ePrint Archive, Report 2006/080, 2006. `http://eprint.iacr.org/2006/080/`, To Appear in ACISP 2006.
19. H. Petersen and P. Horster. Self-certified keys - concepts and applications. In *3rd Int. Conference on Communications and Multimedia Security*, pages 102–116. Chapnam and Hall, 1997.
20. S. Saeednia. Identity-based and self-certified key-exchange protocols. In *ACISP 1997*, pages 303–313. Springer-Verlag, 1997. LNCS No. 1270.
21. S. Saeednia. A note on girault's self-certified mode. *Information Processing Letters*, 86(6):323–327, 2003.
22. A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 84*, pages 47–53. Springer-Verlag, 1984. LNCS No. 196.
23. Y. Shi and J. Li. Provable efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/287, 2005. `http://eprint.iacr.org/2005/287/`.
24. V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT 97*, volume 1233 of *LNCS*, pages 250–266. Springer, 1997.
25. B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer-Verlag, 2005.
26. D. H. Yum and P. J. Lee. Generic construction of certificateless encryption. In *ICCSA '04*, pages 802–811. Springer-Verlag, 2004. LNCS No. 3040.
27. D. H. Yum and P. J. Lee. Generic construction of certificateless signature. In *ACISP '04*, pages 200–211. Springer-Verlag, 2004. LNCS No. 3108.

A Complexity Assumptions

Definition 7 (Non pairing-based Generalized Bilinear DH (NGBDH) Assumption). *Given a group G of prime order p with generator g and elements $g^a, g^b \in G$ where a, b are selected uniformly at random from $\mathbb{Z}_p^*$, the NGBDH problem in G is to output (g^{abc}, g^c).*

An adversary $\mathcal{B}$ has at least an ϵ advantage if

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

We say that the (ϵ, t)-NGBDH assumption holds in a group G if no algorithm running in time at most t can solve the NGBDH problem in G with advantage at least ϵ.

It can be seen as the non pairing-based version of the Generalized Bilinear Diffie-Hellman (GBDH) Problem [1].

Definition 8 (Many-DH Assumption [17] (Simplified Version)). *Given a group G of prime order p with generator g and elements $g^a, g^b, g^c, g^{ab}, g^{ac}, g^{bc} \in G$ where a, b, c are selected uniformly at random from $\mathbb{Z}_p^*$, the Many-DH problem in G is to output g^{abc}.*

An adversary $\mathcal{B}$ has at least an ϵ advantage if

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

We say that the (ϵ, t)-Many-DH assumption holds in a group G if no algorithm running in time at most t can solve the Many-DH problem in G with advantage at least ϵ.

In the original version presented in [17], the number of input tuples can be as much as $\mathcal{O}(\log k)$ for some security parameter k. Here we simplify it for just enough to our scheme.

Definition 9 (Decisional Bilinear DH-1 (DBDH-1) Assumption). [7] *Given a group G of prime order p with generator g and elements $g^a, g^b, g^c, g^z \in G$ where a, b, c, z are selected uniformly at random from $\mathbb{Z}_p^*$. A fair binary coin $\beta \in \{0, 1\}$ is flipped. If $\beta = 1$, it outputs the tuple $(g, A = g^a, B = g^b, C =$*

[7] We use the same notation as used in [6] to denote the classical non pairing-based version. We use DBDH-2 to denote the pairing-based version which is presented in Definition 10. Although it is not used in our schemes, we state here for completion.

$g^c, Z = g^{abc}$). *If* $\beta = 0$, *it outputs the tuple* $(g, A = g^a, B = g^b, C = g^c, Z = g^z)$. *The problem is to guess the value of* β.

An adversary $\mathcal{B}$ *has at least an* ϵ *advantage in solving the DBDH-1 problem if*

$$\left| \Pr[\mathcal{B}(g, g^a, g^b, g^c, g^{abc}) = 1] - \Pr[\mathcal{B}(g, g^a, g^b, g^c, g^z) = 1] \right| \geq 2\epsilon$$

where the probability is oven the randomly chosen a, b, c, z *and the random bits consumed by* $\mathcal{B}$.

We say that the (ϵ, t)*-DBDH-1 assumption holds in a group* G *if no algorithm running in time at most* t *can solve the DBDH-1 problem in* G *with advantage at least* ϵ.

Definition 10 (Decisional Bilinear DH-2 (DBDH-2) Assumption). *Given a group* G *of prime order* p *with generator* g, *a bilinear pairing* $e : G \times G \to G_1$ *and elements* $g^a, g^b, g^c \in G$, $e(g, g)^z \in G_1$ *where* a, b, c, z *are selected uniformly at random from* $\mathbb{Z}_p^*$. *A fair binary coin* $\beta \in \{0, 1\}$ *is flipped. If* $\beta = 1$, *it outputs the tuple* $(g, A = g^a, B = g^b, C = g^c, Z = e(g, g)^{abc})$. *If* $\beta = 0$, *it outputs the tuple* $(g, A = g^a, B = g^b, C = g^c, Z = e(g, g)^z)$. *The problem is to guess the value of* β.

An adversary $\mathcal{B}$ *has at least an* ϵ *advantage in solving the DBDH-2 problem if*

$$\left| \Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g, g)^z) = 1] \right| \geq 2\epsilon$$

where the probability is oven the randomly chosen a, b, c, z *and the random bits consumed by* $\mathcal{B}$.

We say that the (ϵ, t)*-DBDH-2 assumption holds in a group* G *if no algorithm running in time at most* t *can solve the DBDH-2 problem in* G *with advantage at least* ϵ.

B Security Analysis of SGC-Encryption Scheme in Section 5

The semantic security depends on the underlying CL-encryption scheme. Here we analyze the DoD-Free Security.

Theorem 5 (DoD-Free Secure). *The SGC-encryption scheme proposed in this section is secure against DoD adversary (defined in Section 2) with advantage at least* ϵ *and runs in time at most* t, *assuming that the underlying CL-signature scheme is Type I* (ϵ, t)*-existential unforgeable.*

Proof. Assume there is a DoD adversary $\mathcal{A}$ exists. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to break the underlying CL-signature scheme with probability at least ϵ and in time at most t.

Setup. $\mathcal{B}$ is now the CL-signature adversary. It interacts with the master $\mathcal{M}$ (the challenger of the underlying CL-signature scheme) which gives the public parameter and all necessary oracle access for $\mathcal{B}$ (according to the definition of CL-signature, specified in Def. 2). $\mathcal{B}$ is asked to produce a identity-message-siganture pair $(\widetilde{\mathsf{ID}}, \tilde{m}, \tilde{\sigma})$ (without the knowledge of the partial secret key of $\widetilde{\mathsf{ID}}$).

Oracle Simulation. In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to answer all oracle queries for $\mathcal{A}$.

(Partial-Secret-Key-Extract-Oracle.) When $\mathcal{B}$ receives a query for an identity ID, $\mathcal{B}$ queries $\mathcal{M}$'s Partial-Secret-Key-Extract-Oracle for identities $\langle$"SIGN"$||$ID$\rangle$ and $\langle$"ENC"$||$ID$\rangle$ to get $\mathsf{CL.psk_{sign}}$ and $\mathsf{CL.psk_{enc}}$ respectively. It outputs $\mathsf{psk} := \langle\ \mathsf{CL.psk_{sign}}\ ||\ \mathsf{CL.psk_{enc}}\ \rangle$ to $\mathcal{A}$.

(Secret-Key-Extract-Oracle.) When $\mathcal{B}$ receives a query for an identity ID with a partial secret key psk, it forwards the request to $\mathcal{M}$'s Secret-Key-Extract-Oracle to get a secret key and compute the output according to the algorithm described.

(Decryption-Oracle.) When $\mathcal{B}$ receives a query for a ciphertext C and an identity ID, $\mathcal{B}$ queries $\mathcal{M}$'s Secret-Key-Extract-Oracle for an identity ID. It returns a secret key CL.sk. It also queries $\mathcal{M}$'s Partial-Secret-Key-Extract-Oracle for an identity $\langle$"ENC"$||$ID$\rangle$. It returns a partial secret key $\mathsf{CL.psk_{enc}}$. $\mathcal{B}$ uses this secret key and partial secret key to decrypt the ciphertext C, using the **Decrypt** algorithm described

in the scheme. Note that even if ID is the challenged one, $\mathcal{B}$ is still allowed to query $\mathcal{M}$'s Partial-Secret-Key-Extract-Oracle for identity $\langle \text{“ENC”} || \mathsf{ID} \rangle$. $\mathcal{B}$ is only not allowed to query for identity ID.

For other oracle queries, $\mathcal{B}$ just simply forwards the corresponding queries to $\mathcal{M}$ and forwards the replies from $\mathcal{M}$ to $\mathcal{A}$. In addition, $\mathcal{B}$ stores all queries and answers in a database.

Output Calculation. After a polynomial number of oracle queries, $\mathcal{A}$ outputs a message m^* and an identity ID^*. $\mathcal{A}$ wins if the following conditions fulfill:

1. The public key pk^* of ID^* is valid. That is, σ^* is a valid signature on message $m' := \langle\ \mathsf{ID}^* \ ||\ \mathsf{CL.pk}^*\ \rangle$ and identity $\langle \text{“SIGN”} || \mathsf{ID}^* \rangle$, where $\langle \mathsf{CL.pk}^*, \sigma^* \rangle := \mathsf{pk}^*$.
2. $\mathbf{Decrypt}(C^*, \mathsf{sk}^*, \mathsf{psk}^*) \neq m^*$ where $C^* = \mathbf{Encrypt}(m^*, \mathsf{ID}^*, \mathbf{pk}^*)$.
3. $\mathcal{A}$ does not know the partial secret key psk^* (it does not query the Partial-Secret-Key-Extract-Oracle for ID^*. That means $\mathcal{A}$ does not know $\mathsf{CL.psk}^*_{\mathsf{sign}}$).

If the public key of ID^* has not been replaced, due to correctness we always have $\mathbf{Decrypt}(C^*, \mathsf{sk}^*, \mathsf{psk}^*) = m^*$. Condition (2) implies $\mathsf{CL.pk}^*$, the public key of ID^*, has been replaced. Together with condition (1) and (3), it implies that σ^* is a successful forgery.

$\mathcal{B}$ knows σ^* from the replaced public key pk^* by looking into the database. $\mathcal{B}$ outputs $\widetilde{\mathsf{ID}} \leftarrow \langle \text{“SIGN”} || \mathsf{ID}^* \rangle$, $\tilde{m} \leftarrow m'$, $\tilde{\sigma} \leftarrow \sigma^*$ to $\mathcal{M}$ as a signature forgery.

Probability and Time Analysis. It is easy to see that the successful probability and time complexity of $\mathcal{B}$ to forge the signature is the same as $\mathcal{A}$ to break the DoD security. That is, with probability ϵ and running time τ. □