Certificateless Signatures: Structural Extensions of Security Models and New Provably Secure Schemes

Yu-Chi Chen[1], Raylin Tso[2], Willy Susilo[3], Xinyi Huang[4], and Gwoboa Horng[1]

[1]Department of Computer Science and Engineering,
National Chung Hsing University, Taiwan
[2]Department of Computer Science,
National Chengchi University, Taiwan
[3]Centre for Computer and Information Security Research,
School of Computer Science and Software Engineering,
University of Wollongong, Australia
[4]School of Mathematics and Computer Science,
Fujian Normal University, China

Abstract

Certificateless signatures (CLSs) were introduced to solve the key escrow problem of identity-based signatures. In CLS, the full private key is determined by neither the user nor the trusted third party. However, a certificate of a public key is not required in CLS schemes; therefore, anyone can replace the public key. On the formal security, there are two types of adversaries where the Type I adversary acts as the outsider, and the Type II as the key generation center. Huang et al. took a few security issues into consideration and provided some security models. They showed three kinds of Type I adversaries with different security levels. Moreover, Tso et al. found the existence of another Type I adversary that was not discussed by Huang et al.; however, the adversaries are still too subtle to be presently defined. In this paper, we further consider public key replacement and strong unforgeability in certificateless signatures. All feasible situations are revisited along with abilities of adversaries. Additionally, structural extensions of security models are proposed with respect to the described public key replacement and strong unforgeability. Moreover, we also present some schemes, analyze their security against different adversaries, and describe our research results. Finally, one of the proposed certificateless short signature schemes is proven to achieve the strongest security level.

1 Introduction

Public key cryptography is well-known for its ability to realize secure communications between a sender and a receiver when the sender and the receiver do not have a shared key. One of the security issues is the authenticity of public keys. A straightforward and effective approach to public key authentication is to adopt a public key infrastructure (PKI) based system. The trusted entity, referred to as certification authority (CA), is in charge of the certificates used to bind users and their respective public keys. The CA must manage and maintain these certificates through certificate revocations and verifications, which are expensive and daunting tasks.

The notion of Identity-based (ID-based) cryptography was put forth by Shamir [27] to overcome the aforementioned problem. Certificates of the public keys are eliminated in an ID-based cryptosystem. A user's public key is unique information such as an e-mail address. In particular, a trusted third party, referred to as a private key generator (PKG), generates private keys for all users. The PKG decides a master secret key, *msk*, at random, and then computes the master public key, *mpk*, accordingly. Each user can obtain a private key that is outputted by the PKG using *msk*. However, despite the lack of a certificate, this ID-based cryptosystem incurs the inherent key escrow problem, which means that the PKG knows all of the users' private keys. This problem can be resolved through the use of multiple PKGs, although an additional communication cost is necessary.

Certificateless cryptography was first introduced by Al-Riyami and Paterson [1] to solve the key escrow problem of ID-based cryptosystem. Certificates are also not needed in this system; rather, a semi-trusted third party, called a key generation center (KGC) instead of a PKG, generates *partial private keys*. As a PKG does in the ID-based cryptosysem, the KGC chooses a master secret key, *msk*, at random, and then computes the master public key, *mpk*. Each user can obtain a partial private key that is outputted by the KGC using *msk*. Moreover, users can decide their secret value, and their full private keys are composed of partial private keys and chosen secret values. Consequently, the KGC cannot obtain users' secret keys. A user can therefore be a legal receiver if and only if the user has the full private key. As a result, the key escrow problem can be eliminated due to the use of the partial private key in this certificateless cryptosystem.

1.1 Related work

Since the introduction of certificateless cryptography, certificateless signature (CLS) has drawn attention of the research community in the last few years as an alternative to certificateless encryption [8, 12, 21, 33]. simulating possible attacks, two different types of adversaries are defined in the security models of CLS, which are referred to as Type I and Type II adversaries separately. A Type I adversary acts as an outsider who can replace the public keys but cannot access the master secret key, whereas a Type II adversary acts as the KGC that can access the master secret key but cannot replace the public keys.

The first CLS scheme was proposed by Al-Riyami and Paterson [1] in 2003; however, Huang et al. [20] indicated a security loophole in that signature scheme. Later, Yum and Lee [36] proposed a generic construction of CLS in 2004, but Hu et al. [16, 17] had found that construction is insecure against the Type I adversary and also provide an improvement. In 2007, Huang et al. [18, 19] defined formal security models in which the adversaries can be categorized into Normal, Strong, and Super adversaries (ordered based on their attack powers). Tso et al. [29] also showed the existence of another security model. As a result, since the original core of certificateless cryptography, many different kinds of CLS schemes and security models have been presented [15, 24, 34, 37]. With the proposal of some applications of CLS [4, 5, 10, 26, 32, 35], certificateless cryptography has gathered significant attention in the field of cryptography.

1.2 Contributions

On the security of certificateless signatures, a Type I adversary is more complicated than a Type II adversary because of the public key replacement. Therefore, the security models for the Type I adversary are quite subtle in discussing the security levels and requirements. In this paper, we consider important security issues of certificateless signature. This paper seems fully extensive from Huang et al. and Tso et al.'s paper [18, 19, 29]. The contributions of this paper are summarized as follows.

1. *Revisit the public key replacement.* In the literature, public key replacement was usually unclear and unaccounted in CLS because it is performed by an outsider, which also creates many different security models such as different Type I adversaries. In fact, as some Type I adversaries are similar or the same, we analyze possible activities of the outsider in depth, and provide a definition for replacing public keys.

2. *Present all potential security models.* First, the Strong Type I adversary, defined by Huang et al. [18, 19], is shown to be dispensable according to the real attack power and public key replacement. Moreover, we also take some possible situations into consideration, and then propose a structural extension for security models of Type I adversaries. It includes eight kinds of Type I adversaries. In addition, we also generate a structural extension for showing all potential Type II adversaries.

3. *Analyze and propose the relations between CLS schemes and security models.* We review and survey some CLS schemes, including the six proposed schemes, and then analyze their security against different kinds of Type I adversaries. In particular, one of the proposed schemes reaches the strongest security level.

4. *Demonstrate some research results.* There is an open problem shown by Shim [28] in certificateless short signature schemes. She gave an attack, performed by some Type I adversaries, to point out a weakness of some schemes [7, 9, 11, 18, 19, 30, 31]. However, we overcome this problem by proposing the new secure schemes. Moreover, we point out the relation between strong unforgability and non-repudiation in short CLS schemes. The crossing point of these two properties is based on the Type I adversaries replacing public keys. In details, we will describe the results later with some security comparisons

The rest of this paper is organized as follows. In Section 2, we briefly review the construction of CLS and adversaries' attack powers. In Section 3 and 4, security models are shown to simulate whole Types I and II adversaries. Therefore we present some CLS schemes against different adversaries respectively in Section 5, and analyze their security in Section 6. Moreover, discussions and comparisons of certificateless short signature schemes are demonstrated in Section 7. Eventually, we give conclusions for this paper in Section 8.

2 Overview of certificateless signature

A certificateless signature scheme involves three entities, the KGC, a user/signer, and a verifier. Normally, it consists of the following algorithms: Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Secret-Key, Set-Public-Key, Sign, and Verify:

- Setup: This algorithm, run by the KGC, takes a security parameter as an input, and then returns the master secret key, *msk*, and system parameter, *param*.
- Partial-Private-Key-Extract: This algorithm, run by the KGC, takes *param*, *msk* and a user's identity *ID* as inputs. It generates a partial-private-key D_{ID}, and sends it to the user via a secure channel.
- Set-Secret-Value: This algorithm, run by a user, returns a secret value, r_{ID}.
- Set-Secret-Key: This algorithm, run by a user, takes the user's partial-private-key D_{ID} and the secret value r_{ID} as inputs, then returns the user's full secret key, sk_{ID}.
- Set-Public-Key: This algorithm, run by a user, takes *param* and the user's full secret key as inputs, and returns a public key pk_{ID} for the user.
- Sign: This algorithm, run by a signer/user, takes *param*, a message *m*, and the user's full secret key, sk_{ID}, as inputs. It then generates σ as the signature for the message *m*.
- Verify: This algorithm, run by a verifier, takes *param*, a public key pk_{ID}, a message *m*, a user's *ID*, and a signature σ as inputs. It returns 1 as the verifier accepts the signature σ if σ is the signature of the message *m*, the public key pk_{ID}, and the user with *ID*. It returns 0 if not.

2.1 Adversaries' activities and behaviors

Since Al-Riyami and Paterson first introduced CLS [1], plenty of research works have been presented for CLS; for example, the adversaries and their attack powers. In the literature, we have the following definition regarding the Type I and Type II adversaries in CLS.

Definition 1. *The Type I adversary, $\mathcal{A}_I$, acts as the outsider who can replace public keys but cannot access the master secret key. The Type II adversary, $\mathcal{A}_{II}$, acts as the KGC which can access the master secret key but cannot replace public keys.*

The complete Types I and II adversaries will be presented in details later (in Section 3 and 4). Here, we present an overview of oracles which is used to simulate adversaries' activities and behaviors. Basically, the following three oracles can be accessed by the Type I or II adversary in certificateless cryptography [1, 17, 18, 19, 36].

- **Create-User**: This oracle takes ID as an input. Nothing will be returned by the oracle if ID has been created before. Otherwise, it will perform Partial-Private-Key-Extract, Set-Secret-Value, and Set-Public-Key for ID to get the partial-private-key D_{ID}, the secret value r_{ID}, and the public key pk_{ID}. Finally, it adds $\langle ID, D_{ID}, r_{ID}, pk_{ID}\rangle$ to K-list and returns pk_{ID}.

- **Public-Key-Replace**: This oracle takes (ID, r'_{ID}, pk'_{ID}) or $(ID, \perp, pk'_{ID})$ as an input, where ID has been created. Here, $\perp$ denotes that the adversary does not provide the corresponding secret value r'_{ID} for pk'_{ID}. It will replace the ID's public key with the new public key pk'_{ID} to update K-list if the input is $(ID, \perp, pk'_{ID})$. Otherwise, it will replace the ID's key with the new public key pk'_{ID} and secret value r'_{ID} to update K-list.

- **Secret-Value-Extract**: This oracle takes ID as an input. It will return r_{ID} from K-list.

In fact, there is another oracle, **Partial-Private-Key-Extract**, which can be accessed by the Type I adversaries only, because the Type II adversaries have the master key.

- **Partial-Private-Key-Extract**: This oracle takes ID as an input. It will return D_{ID} from K-list.

In certificateless signature schemes, the **Sign** oracle is important. To simulate and perform the adaptively chosen message and identity attack, an adversary can send a query, a message/identity pair, to the **Sign** oracle, and then it will receive a message/identity/signature triplet. We now describe and analyze the different **Sign** oracles below.

2.2 Sign oracle

In the literature, Huang et al. [18, 19] showed three kinds of different **Sign** oracles: Normal-Sign, Strong-Sign, and Super-Sign. To observe these oracles, the inputs of Normal-Sign and Super-Sign are (ID, m), but that of Strong-Sign is (ID, r_{ID}, m). However, Strong-Sign is unreasonable because the chosen message and identity adversary may not know the secret value during his attack. Although Huang et al. had found a real-life scenario in which the user might reveal his secret value, it does not fully match Strong-Sign up. In other words, the adversary can query a signature of (ID, m) and a secret value of ID instead of Strong-Sign with (ID, r_{ID}, m). As a result, Strong-Sign is directly referred to as the transition between Normal-Sign and Super-Sign, and is ignored in our consideration. In this paper, Normal-Sign is denoted by N-Sign for short and Super-Sign by S-Sign.

Now we will describe the two practical **Sign** oracles: **N-Sign** and **S-Sign**. In particular, there is one significant property to distinguish N-Sign from S-Sign.

- **N-Sign** only returns a signature of (ID, m) if the ID's public key has never been replaced. In this case of N-Sign, we consider a real-life attack that the adversary can eavesdrop to get or be the verifier to receive ID's valid signatures which are generated by ID using his private key. However, $\mathcal{A}_I$ can replace ID's public key with a new one pk'_{ID}, but it is impossible to obtain any signature which is valid on the replaced public key. Hence, N-Sign is defined to return a signature of (ID, m) if the ID's public key has never been replaced.

- **S-Sign** returns a signature of (ID, m), no matter whether the public key has been replaced or not. In this case of S-Sign, we have not found a real and suitable attack, but S-Sign could be regarded as an oracle with the full attack power. However, the scenario of revealing the secret value is under S-Sign, since replacing a public key with a corresponding secret value is equal to getting the secret value. As a result, S-Sign is more powerful than N-Sign undoubtedly.

3 Security models for Type I adversaries

The notion of the security on CLS is known, but the security models are quite subtle to be formal defined in the literature. We consider some attack scenarios to simulate all potential Type I adversaries, and eventually define security models.

3.1 Strong unforgeability and existential unforgeability

In certificateless signatures, for existential unforgeability under the adaptive chosen message and identity attack, the goal of the adversary is to output a forged signature σ^* on (ID^*, m^*), and in the meanwhile, the following conditions hold. (We can look (σ^*, m^*, ID^*) as the forgery of the adversary.)

1. σ^* is a valid signature of (ID^*, m^*), which means σ^* can pass verification.
2. (ID^*, m^*) has never been submitted to require the signature.
3. σ^* has never been returned.

Nevertheless, for strong unforgeability under the adaptive chosen message and identity attack, the goal of the adversary and Conditions 1 and 3 are the same as before, but the different Condition 2 must hold as follows.

2. (ID^*, m^*) can be submitted to require the signature.

Due to the public key replacement, we give the lead-in of strong unforgeability which had not been discussed previously in CLS. Here we briefly show an example with respect to short signature and certificateless short signature. In normal short signature, the signature of m is unique for m thus this signature scheme is strong unforgeability. However, in certificateless short signature, the signature of (ID, m) might not be unique for (ID, m) since the adversary has ability to replace the public key. As the above result, strong unforgeability is a truly important issue in CLS.

In the following, the potential Type I adversaries' behaviors are simulated to define the security models of CLS. As we mentioned in Section 2.2 and 3.1, Sign oracle and unforgeability of σ^* on (ID^*, m^*) are two important issues for simulating the Type I adversary; however, there exists another one, the secret value of ID^*. Therefore we list three optional conditions which would be considered, whereas $\mathcal{A}_I$'s goal is to output a forged signature σ^* on (ID^*, m^*) which has never been returned by Sign oracle.

- The **Sign** oracle is **N-Sign** or **S-Sign**.
- ID^* can be submitted to require the secret value or not.
- (ID^*, m^*) can be submitted to require the signature or not.

Table 1: Eight different kinds of Type I adversaries

	Type of Sign oracle	Submit ID^* to **Secret-Value-Extract**	Submit (ID^*, m^*) t
N-Type I	N-Sign	×	×
SV-Type I	N-Sign	✓	×
SU-Type I	N-Sign	×	✓
SS-Type I	S-Sign	×	×
SV-SU-Type I	N-Sign	✓	✓
SS-SU-Type I	S-Sign	×	✓
SS-SV-Type I	S-Sign	✓	×
S-Type I	S-Sign	✓	✓

Due to these, we have eight (2^3) kinds of Type I adversaries named as N-Type I, SV-Type I, SU-Type I, SS-Type I, SV-SU-Type I, SS-SU-Type I, SS-SV-Type I, and S-Type I adversaries, respectively. For example, SV-Type I means that this Type I adversary can submit ID^* to **Secret-Value-Extract**. The comparisons of eight Type I adversaries are illustrated in Table 1. However, their activities and behaviors can be simulated to the following security games. Here we only present N-Type I, SV-Type I, SS-SU-Type I, and S-Type I adversaries. For other Type I adversaries, we can refer to Appendix A in details.

3.2 Security against the N-Type I adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity N-Type I adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* including the master public key to $\mathcal{A}_I$.

Query: In this phase, $\mathcal{A}_I$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, **Secret-Value-Extract**, and **Partial-Private-Key-Extract** defined in Section 2.1. Moreover, $\mathcal{A}_I$ can submit queries to the **N-Sign** oracle defined in Section 2.2.

Forgery: $\mathcal{A}_I$ outputs a forged triplet (σ^*, m^*, ID^*). $\mathcal{A}_I$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) has never been submitted to **N-Sign**.

2. σ^* has never returned by **N-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s *current* public key.

3. ID^* has never been submitted to **Partial-Private-Key-Extract** or **Secret-Value-Extract**.

3.3 Security against the SV-Type I adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity SV-Type I adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* including the master public key to $\mathcal{A}_I$.

Query: $\mathcal{A}_I$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, **Secret-Value-Extract**, and **Partial-Private-Key-Extract** defined in Section 2.1. Moreover, $\mathcal{A}_I$ can submit queries to the **N-Sign** oracle defined in Section 2.2.

Forgery: $\mathcal{A}_I$ outputs a forged triplet (σ^*, m^*, ID^*). $\mathcal{A}_I$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) has never been submitted to **N-Sign**.

2. σ^* has never returned by **N-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s current public key.

3. ID^* has never been submitted to **Partial-Private-Key-Extract**.

3.4 Security against the SS-SU-Type I adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity SS-SU-Type I adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* including the master public key to $\mathcal{A}_I$.

Query: $\mathcal{A}_I$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, **Secret-Value-Extract**, and **Partial-Private-Key-Extract** defined in Section 2.1. Moreover, $\mathcal{A}_I$ can submit queries to the **S-Sign** oracle defined in Section 2.2.

Forgery: $\mathcal{A}_I$ outputs a forged triplet (σ^*, m^*, ID^*). $\mathcal{A}_I$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) can be submitted to **S-Sign**.

2. σ^* has never returned by **S-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s current public key.

3. ID^* has never been submitted to **Partial-Private-Key-Extract** or **Secret-Value-Extract**.

3.5 Security against the S-Type I adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity S-Type I adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* including the master public key to $\mathcal{A}_I$.

Query: $\mathcal{A}_I$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, **Secret-Value-Extract**, and **Partial-Private-Key-Extract** defined in Section 2.1. Moreover, $\mathcal{A}_I$ can submit queries to the **S-Sign** oracle defined in Section 2.2.

Forgery: $\mathcal{A}_I$ outputs a forged triplet (σ^*, m^*, ID^*). $\mathcal{A}_I$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) can be submitted to **S-Sign**.
2. σ^* has never returned by **S-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s current public key.
3. ID^* has never been submitted to **Partial-Private-Key-Extract**.

Upon showing the different Type I adversaries with their corresponding security games separately, we have the following definition for the security of a CLS scheme.

Definition 2. *A certificateless signature scheme is secure against a Type I adversary if and only if no PPT algorithm has non-negligible probability of winning the corresponding game.*

In a practical sense, we also adopt the form of Table 1 to trace all potential Type II adversaries. More details of Type II adversaries are located in Section 4.

3.6 Remarks on Type I adversaries

To the best of our knowledge, the only paper deals with some attack situations in CLS is the paper of Huang et al. [18, 19]. They deeply considered and defined three kinds of Type I adversaries which are briefly discussed in Section 2.2. Without the strong Type I adversary of Huang et al.[1], the

[1] In Section 2.2, we have analyzed the Strong-Sign oracle is unreasonable, thus the the strong Type I adversary is also informal since it sends requests to the Strong-Sign oracle. Factually, the Strong-Sign oracle can be done and referred to as a combination of the **S-Sign** oracle and the **Public-Key-Replace** oracle. The strong Type I adversary sending (ID, r'_{ID}, m) to the Strong-Sign oracle is equivalent to the SS-Type I adversary (SS-SV-Type I, SS-SU-Type I, or S-Type I adversaries as well) which first sends (ID, r'_{ID}, pk'_{ID}) to **Public-Key-Replace** and then asks for a signature of (ID, m) to **S-Sign**.

normal Type I adversary is the same with the N-Type I adversary, and the super Type I adversary is with the SS-SV-Type adversary. Moreover, Tso et al. [29] also found another Type I adversary which is the same with the SS-Type adversary.

In addition to the Type I adversaries mentioned above, there exists another Type I adversary [34, 36] which is weaker than our N-Type I adversary. This is referred to as the W-Type I adversary. The activities and behaviors of the W-Type I adversary will be simulated and modelled by the following security game.

Security against the W-Type I adversary: The unforgeability of a CLS scheme against the adaptive chosen message and identity W-Type I adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* including the master public key to $\mathcal{A}_I$.

Query: $\mathcal{A}_I$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, **Secret-Value-Extract**, and **Partial-Private-Key-Extract** defined in Section 2.1. Moreover, $\mathcal{A}_I$ can submit queries to the **N-Sign** oracle defined in Section 2.2.

Forgery: $\mathcal{A}_I$ outputs a forged triplet (σ^*, m^*, ID^*). $\mathcal{A}_I$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) has never been submitted to **N-Sign**.

2. σ^* has never returned by **N-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s *original* public key (which has never been replaced).

3. ID^* has never been submitted to **Partial-Private-Key-Extract** or **Secret-Value-Extract**.

Due to the winning conditions of the W-Type I adversary, pk_{ID^*} is the ID^*'s *original* current public key, which violates Definition 1. We conclude that the W-Type I adversary is not feasible for CLS, thus some CLS schemes [1, 11] are weak to be used in real-life because they are only proven to be secure against the W-Type I adversary.

4 Security models for Type II adversaries

We use the same concept to construct the extension of security models for Type II adversaries as well as those for Type I adversaries. Straightly, Table 2 shows all potential Type II adversaries. We present N-Type II and

Table 2: Four different kinds of Type II adversaries

	Type of Sign oracle	Submit (ID^*, m^*) to **Sign**
N-Type II	N-Sign	×
SU-Type II	N-Sign	✓
SS-Type II	S-Sign	×
S-Type II	S-Sign	✓

S-Type II adversaries as follows. For more details, SU-Type II and SS-Type II adversaries are shown in Appendix B.

4.1 Security against the N-Type II adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity N-Type II adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* and the master key *msk* to $\mathcal{A}_{II}$.

Query: In this phase, $\mathcal{A}_{II}$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, and **Secret-Value-Extract** defined in Section 2.1. Moreover, $\mathcal{A}_{I}$ can submit queries to the **N-Sign** oracle defined in Section 2.2.

Forgery: $\mathcal{A}_{II}$ outputs a forged triplet (σ^*, ID^*, m^*). $\mathcal{A}_{II}$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) has never been submitted to **N-Sign**.
2. σ^* has never returned by **N-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s *original* public key.
3. ID^* has never been submitted to **Secret-Value-Extract**.

4.2 Security against the S-Type II adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity S-Type II adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* and the master key *msk* to $\mathcal{A}_{II}$.

Query: In this phase, $\mathcal{A}_{II}$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, and **Secret-Value-Extract** defined in Section 2.1. Moreover, $\mathcal{A}_{I}$ can submit queries to the **S-Sign** oracle defined in Section 2.2.

Forgery: $\mathcal{A}_{II}$ outputs a forged triplet (σ^*, ID^*, m^*). $\mathcal{A}_{II}$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) can be submitted to **S-Sign**.
2. σ^* has never returned by **S-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s original public key.
3. ID^* has never been submitted to **Secret-Value-Extract**.

Definition 3. *A certificateless signature scheme is secure against a Type II adversary if and only if no PPT algorithm has non-negligible probability of winning the corresponding game.*

5 Certificateless short signature schemes

Boneh et al. introduced the concept of short signatures in 2001 [3], which are useful for systems with low bandwidth or low computation power. Inheriting the advantages of both certificateless cryptography and short signatures, certificateless short signatures were introduced, and then have garnered considerable attention in recent years. However, the short CLS schemes in the literature [29] are not secure against the Type I adversaries who are allowed to submit ID^* to **Secret-Value-Extract**; for instance, existing short CLS schemes [7, 9, 11, 18, 19, 29, 30, 31] cannot withstand the SV-Type I, SS-SV-Type I, SV-SU-Type I, or S-Type I adversary. This is referred to as an open problem in short CLS.

In this section, we first describe bilinear pairing. In Section 5.3 through 5.10, we present nine certificateless short signature schemes, including three literature schemes and six new schemes (proposed in this paper). However, these schemes are respectively secure against different Type I adversaries which are ordered as Table 1.

5.1 Bilinear pairing

A bilinear pairing is a mapping $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive cyclic groups of prime order q, and $\mathbb{G}_q$ is a multiplicative cyclic group of the same order q. Additionally, bilinear pairing is with the following properties:

(1) Computable: given $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, there exists a polynomial time algorithm to compute $\hat{e}(P, Q) \in \mathbb{G}_T$.

(2) Bilinear: for any $x, y \in \mathbb{Z}_q^*$, we have $\hat{e}(xP, yQ) = \hat{e}(P, Q)^{xy}$ for any $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.

(3) Non-degenerate: if P is a generator of $\mathbb{G}_1$ and Q is a generator of $\mathbb{G}_2$, then $\hat{e}(P, Q) \neq 1$.

However, there are three kinds of the bilinear pairings based on the relation between $\mathbb{G}_1$ and $\mathbb{G}_2$.

- Type 1: $\mathbb{G}_1 = \mathbb{G}_2$ is a group of prime order q.
- Type 2: $\mathbb{G}_1 \neq \mathbb{G}_2$ are groups of prime order q but with an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.
- Type 3: $\mathbb{G}_1 \neq \mathbb{G}_2$ are groups of prime order q without any isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.

There are several works to propose speed-up algorithms to improve the efficiency regarding computation [3, 23, 25].

Definition 4. *(Computational Diffie-Hellman (CDH) Problem in $\mathbb{G}_1$) Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups with the bilinear map, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Given (P, aP, bP) for unknown $a, b \in \mathbb{Z}_q^*$, compute abP. If there is a probabilistic polynomial-time algorithm $\mathcal{A}$ with probability at least ε to solve the CDH problem, $\Pr[\mathcal{A}(P, aP, bP) \to abP] \geq \varepsilon$.*

The CDH problem is assumed to be intractable if for any PPT algorithm $\mathcal{A}$, $\Pr[\mathcal{A}(P, aP, bP) \to abP]$ is negligible. However, in security proof of cryptographic schemes, we also define the CDH problem is a hardness assumption. In fact, the bilinear pairing is widely adopted to design cryptographic schemes. Therefore the following CLS schemes are pairing-based.

5.2 Fan et al.'s scheme against the W-Type I adversary

Fan et al.'s scheme is found to be only secure against the W-Type I adversary, which means it cannot withstand the N-Type I adversary. This scheme is composed of the following algorithms.

Setup: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be additive cyclic groups of prime order q, $\mathbb{G}_T$ be a multiplicative cyclic group of the same order, and e be the bilinear pairing where $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Moreover, let $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_1 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{G}_2$ be two cryptographic hash functions. The KGC randomly chooses $s \in \mathbb{Z}_q^*$ as the master secret key $msk = s$, and then picks the generators $P_1 \in \mathbb{G}_1$ and $P_2 \in \mathbb{G}_2$ with $g = \hat{e}(P_1, P_2)$. Finally, it publishes the system parameter, $param = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, H_0, H_1, q, P_1, P_2, P_{pub} = sP_2\}$.

Partial-Private-Key-Extract: Given a user's identity ID, the KGC uses $msk = s$ to compute the ID's partial private key, $D_{ID} = \frac{1}{s+H_0(ID)+H_0(ID||pk_{ID,1})}P_1$. It thus gives D_{ID} to ID via a secure channel.

Set-Secret-Value: The user ID selects $r_{ID} \in \mathbb{Z}_q^*$ at random and sets r_{ID} as his secret value.

Set-Secret-Key: The user ID sets his full secret key, $sk_{ID} = \{D_{ID}, r_{ID}\}$.

Set-Public-Key: Given ID's secret value r_{ID}, the user ID obtains his public key $pk_{ID} = \{pk_{ID,1}, pk_{ID,2}\}$ where $pk_{ID,1} = r_{ID}P_2$ and $pk_{ID,2} = r_{ID}(H_0(ID)P_2 + P_{pub})$.

Sign: Given a message m and ID's secret key sk_{ID}, the signer/user ID generates the signature
$\sigma = \frac{1}{r_{ID}+H_1(m,pk_{ID,1})}D_{ID}$.

Verify: Taking $(m, \sigma, pk_{ID}, ID, param)$ as input, the verifier sets $h = H_1(m, pk_{ID,1})$, and the algorithm returns 1 if the following equation holds, $\hat{e}(\sigma, pk_{ID,2} + H_0(ID||pk_{ID,1})pk_{ID,1} + h(P_{pub} + H_0(ID)P_2 + H_0(ID||pk_{ID,1})P_2)) = g$; otherwise, returns 0.

Remark 1. *Fan et al.'s scheme is insecure against the N-Type I adversary.*

5.3 The proposed scheme 1 against the N-Type I adversary

The proposed scheme 1 against the N-Type I adversary consists of the following algorithms.

Setup: Let $\mathbb{G}_1$ be an additive cyclic group of prime order q, $\mathbb{G}_T$ be a multiplicative cyclic group of the same order, and e be the bilinear pairing where $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Moreover, let $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ be two cryptographic hash functions. The KGC randomly chooses $s \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, and then sets the master secret key $msk = s$ and the master public key $P_{pub} = sP$. Finally, it announces the system parameter, $param = \{\mathbb{G}_1, \mathbb{G}_T, \hat{e}, H_0, H_1, q, P, P_{pub} = sP\}$

Partial-Private-Key-Extract: Given a user's identity ID, the KGC uses $msk = s$ to compute the ID's partial private key, $D_{ID} = sH_0(ID)$. It thus gives D_{ID} to ID via a secure channel.

Set-Secret-Value: The user ID selects $r_{ID} \in \mathbb{Z}_q^*$ at random and sets r_{ID} as his secret value.

Set-Secret-Key: The user ID sets his full secret key, $sk_{ID} = \{D_{ID}, r_{ID}\}$.

Set-Public-Key: Given ID's secret key sk_{ID}, the user ID obtains his public key $pk_{ID} = r_{ID}P$.

Sign: Given a message m and ID's secret key sk_{ID}, the signer/user ID generates the signature
$\sigma = D_{ID} + r_{ID}H_1(m||ID)$.

Verify: Taking $(m, \sigma, pk_{ID}, ID, param)$ as input, the verifier can check the following equation holds or not, $\hat{e}(\sigma, P) =? \hat{e}(H_0(ID), P_{pub})\hat{e}(pk_{ID}, H_1(m||ID))$. If it holds, the algorithm returns 1; otherwise, returns 0.

Remark 2. *Scheme 1 is insecure against the SS-Type I, SU-Type I, or SV-Type I adversary. This scheme is modified from Huang et al.'s scheme (Sect. 5.5), and its formal security proof is almost the same.*

5.4 The proposed scheme 2 against the SV-Type I adversary

The proposed scheme 2 against the SV-Type I adversary consists of the following algorithms.

Setup: Let $\mathbb{G}_1$ be an additive cyclic group of prime order q, $\mathbb{G}_T$ be a multiplicative cyclic group of the same order, and e be the bilinear pairing where $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Moreover, let $H_0 :, H_2\{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ be three cryptographic hash functions. The KGC randomly chooses $s_1, s_2 \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, and then sets the master secret key $msk = \{s_1, s_2\}$ and the master public key $P_{pub1} = s_1P, P_{pub2} = s_2P$. Finally, it announces the system parameter, $param = \{\mathbb{G}_1, \mathbb{G}_T, \hat{e}, H_0, H_1, H_2, q, P, P_{pub1}, P_{pub2}\}$

Partial-Private-Key-Extract: Given a user's identity ID, the KGC randomly chooses $x \in \mathbb{Z}_q^*$ and uses $msk = s$ to compute the ID's partial private key, $D_{ID,1} = x + s_1H_0(ID)$ and $D_{ID,2} = s_2H_1(ID)$. It thus gives $D_{ID} = \{D_{ID,1}, D_{ID,2}\}$ and $pk_{ID,2} = xP$ to ID via a secure channel.

Set-Secret-Value: The user ID selects $r_{ID} \in \mathbb{Z}_q^*$ at random and sets r_{ID} as his secret value.

Set-Secret-Key: The user ID sets his full secret key, $sk_{ID} = \{D_{ID}, r_{ID}\}$.

Set-Public-Key: Given ID's secret key sk_{ID}, the user ID obtains his public key $pk_{ID} = \{pk_{ID,1}, pk_{ID,2}\}$ where $pk_{ID,1} = r_{ID}P$.

Sign: Given a message m and ID's secret key sk_{ID}, the signer/user ID sets $h = H_2(m||ID)$ and generates the signature
$\sigma = \frac{1}{hr_{ID}+D_{ID,1}}D_{ID,2}$.

Verify: Taking $(m, \sigma, pk_{ID}, ID, param)$ as input, the verifier sets $h = H_2(m||ID)$ and can check the following equation holds or not, $\hat{e}(\sigma, h \cdot pk_{ID,1} + pk_{ID,2} + H_0(ID)P_{pub1}) =? \hat{e}(H_1(ID), P_{pub2})$. If it holds, the algorithm returns 1; otherwise, returns 0.

Correctness: If the public key PK_{ID} and the signature σ are generated correctly as this scheme, then the correctness holds since

$$\hat{e}(\sigma, h \cdot pk_{ID,1} + pk_{ID,2} + H_0(ID)P_{pub1})$$

$$\begin{aligned}
&= \hat{e}(\frac{1}{hr_{ID}+D_{ID,1}}D_{ID,2}, h(r_{ID}P)+xP+H_0(ID)(s_1P)) \\
&= \hat{e}(\frac{1}{hr_{ID}+D_{ID,1}}D_{ID,2}, (hr_{ID}+x+H_0(ID)s_1)P) \\
&= \hat{e}(D_{ID,2}, P) \\
&= \hat{e}(s_2H_1(ID), P) \\
&= \hat{e}(H_1(ID), P_{pub2})
\end{aligned}$$

Remark 3. *Scheme 2 is insecure against the SS-Type I or SU-Type I adversary. This scheme is modified from Scheme 4 (Sect. 5.7), and its formal security proof is almost the same.*

5.5 Huang et al.'s scheme against the SU-Type I adversary

Huang et al.'s scheme against the SU-Type I adversary consists of the following algorithms.

Setup: Let $\mathbb{G}_1$ be an additive cyclic group of prime order q, $\mathbb{G}_T$ be a multiplicative cyclic group of the same order, and e be the bilinear pairing where $\hat{e}:\mathbb{G}_1\times\mathbb{G}_1\rightarrow\mathbb{G}_T$. Moreover, let $H_0:\{0,1\}^*\rightarrow\mathbb{G}_1$ and $H_1:\{0,1\}^*\rightarrow\mathbb{G}_1$ be two cryptographic hash functions. The KGC randomly chooses $s\in\mathbb{Z}_q^*$ and $P\in\mathbb{G}_1$, and then sets the master secret key $msk=s$ and the master public key $P_{pub}=sP$. Finally, it announces the system parameter, $param=\{\mathbb{G}_1,\mathbb{G}_T,\hat{e},H_0,H_1,q,P,P_{pub}=sP\}$

Partial-Private-Key-Extract: Given a user's identity ID, the KGC uses $msk=s$ to compute the ID's partial private key, $D_{ID}=sH_0(ID)$. It thus gives D_{ID} to ID via a secure channel.

Set-Secret-Value: The user ID selects $r_{ID}\in\mathbb{Z}_q^*$ at random and sets r_{ID} as his secret value.

Set-Secret-Key: The user ID sets his full secret key, $sk_{ID}=\{D_{ID},r_{ID}\}$.

Set-Public-Key: Given ID's secret key sk_{ID}, the user ID obtains his public key $pk_{ID}=r_{ID}P$.

Sign: Given a message m and ID's secret key sk_{ID}, the signer/user ID generates the signature
$\sigma=D_{ID}+r_{ID}H_1(m||ID||pk_{ID})$.

Verify: Taking $(m,\sigma,pk_{ID},ID,param)$ as input, the verifier can check the following equation holds or not, $\hat{e}(\sigma,P)=?\hat{e}(H_0(ID),P_{pub})\hat{e}(pk_{ID},H_1(m||ID||pk_{ID}))$. If it holds, the algorithm returns 1; otherwise, returns 0.

Remark 4. *Huang et al.'s scheme is insecure against the SS-Type I or SV-Type I adversary. Its formal security proof is done in the paper of [19].*

5.6 The proposed scheme 3 against the SS-Type I adversary

The proposed scheme 3 against the SS-Type I adversary consists of the following algorithms.

Setup: Let $\mathbb{G}_1$ be an additive cyclic group of prime order q, $\mathbb{G}_T$ be a multiplicative cyclic group of the same order, and e be the bilinear pairing where $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Moreover, let $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ be cryptographic hash functions. The KGC randomly chooses $s \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, and then sets the master secret key $msk = s$ and the master public key $P_{pub} = sP$. Finally, it announces the system parameter, $param = \{\mathbb{G}_1, \mathbb{G}_T, \hat{e}, H_0, H_1, q, P, P_{pub} = sP\}$

Partial-Private-Key-Extract: Given a user's identity ID, the KGC randomly chooses $x \in \mathbb{Z}_q^*$ and uses $msk = s$ to compute the ID's partial private key, $D_{ID} = sH_0(ID)$. It thus gives D_{ID} to ID via a secure channel.

Set-Secret-Value: The user ID selects $r_{ID} \in \mathbb{Z}_q^*$ at random and sets r_{ID} as his secret value.

Set-Secret-Key: The user ID sets his full secret key, $sk_{ID} = \{D_{ID}, r_{ID}\}$.

Set-Public-Key: Given ID's secret key sk_{ID}, the user ID obtains his public key $pk_{ID} = \{pk_{ID,1},\ pk_{ID,2},\ pk_{ID,3}\}$ where $pk_{ID,1} = r_{ID}D_{ID}$, $pk_{ID,2} = r_{ID}H_0(ID)$, and $pk_{ID,3}$ is randomly chosen from $\mathbb{G}_1$.

Sign: Given a message m and ID's secret key sk_{ID}, the signer/user generates the signature
$\sigma = D_{ID} + \frac{1}{r_{ID}} H_1(m||ID||pk_{ID,1}||pk_{ID,2}) + pk_{ID,3}$.

Verify: Taking $(m, \sigma, pk_{ID}, ID, param)$ as input, the verifier can check whether the following equations hold or not.

$$\hat{e}(P_{pub}, pk_{ID,2}) =?\hat{e}(P, pk_{ID,1}) \text{ and}$$
$$\hat{e}(\sigma - pk_{ID,3}, pk_{ID,2}) =?\hat{e}(pk_{ID,1} + T, H_0(ID)),$$

where $T = H_1(m||ID||pk_{ID,1}||pk_{ID,2})$. If they hold, the algorithm returns 1; otherwise, returns 0.

Remark 5. *Scheme 3 is insecure against the SU-Type I or SV-Type I adversary. This scheme is modified from Tso et al.'s scheme (Sect. 5.8), and its formal security proof is almost the same.*

5.7 The proposed scheme 4 against the SV-SU-Type I adversary

The proposed scheme 4 against the SV-SU-Type I adversary consists of the following algorithms.

Setup: Let $\mathbb{G}_1$ be an additive cyclic group of prime order q, $\mathbb{G}_T$ be a multiplicative cyclic group of the same order, and e be the bilinear pairing where $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Moreover, let $H_0, H_2\{0,1\}^* \to \mathbb{Z}_q^*$ and $H_1 : \{0,1\}^* \to \mathbb{G}_1$ be three cryptographic hash functions. The KGC randomly chooses $s_1, s_2 \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, and then sets the master secret key $msk = \{s_1, s_2\}$ and the master public key $P_{pub1} = s_1P, P_{pub2} = s_2P$. Finally, it announces the system parameter, $param = \{\mathbb{G}_1, \mathbb{G}_T, \hat{e}, H_0, H_1, H_2, q, P, P_{pub1}, P_{pub2}\}$

Partial-Private-Key-Extract: Given a user's identity ID, the KGC randomly chooses $x \in \mathbb{Z}_q^*$ and uses $msk = s$ to compute the ID's partial private key, $D_{ID,1} = x + s_1H_0(ID)$ and $D_{ID,2} = s_2H_1(ID)$. It thus gives $D_{ID} = \{D_{ID,1}, D_{ID,2}\}$ and $pk_{ID,2} = xP$ to ID via a secure channel.

Set-Secret-Value: The user ID selects $r_{ID} \in \mathbb{Z}_q^*$ at random and sets r_{ID} as his secret value.

Set-Secret-Key: The user ID sets his full secret key, $sk_{ID} = \{D_{ID}, r_{ID}\}$.

Set-Public-Key: Given ID's secret key sk_{ID}, the user ID obtains his public key $pk_{ID} = \{pk_{ID,1}, pk_{ID,2}\}$ where $pk_{ID,1} = r_{ID}P$.

Sign: Given a message m and ID's secret key sk_{ID}, the signer/user ID sets $h = H_2(m||ID||pk_{ID})$ and generates the signature
$\sigma = \frac{1}{hr_{ID}+D_{ID,1}}D_{ID,2}$.

Verify: Taking $(m, \sigma, pk_{ID}, ID, param)$ as input, the verifier sets $h = H_2(m||ID||pk_{ID})$ and can check the following equation holds or not, $\hat{e}(\sigma, h \cdot pk_{ID,1} + pk_{ID,2} + H_0(ID)P_{pub1}) =?\hat{e}(H_1(ID), P_{pub2})$. If it holds, the algorithm returns 1; otherwise, returns 0.

Remark 6. *Scheme 4 is insecure against the SS-Type I adversary.*

5.8 Tso et al.'s scheme against the SS-SU-Type I adversary

Tso et al.'s scheme against the SS-SU-Type I adversary consists of the following algorithms.

Setup: Let $\mathbb{G}_1$ be an additive cyclic group of prime order q, $\mathbb{G}_T$ be a multiplicative cyclic group of the same order, and e be the bilinear pairing where $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Moreover, let $H_0 : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \to \mathbb{G}_1$ be cryptographic hash functions. The KGC randomly chooses $s \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, and then sets the master secret key $msk = s$ and the master public key $P_{pub} = sP$. Finally, it announces the system parameter, $param = \{\mathbb{G}_1, \mathbb{G}_T, \hat{e}, H_0, H_1, q, P, P_{pub} = sP\}$

Partial-Private-Key-Extract: Given a user's identity ID, the KGC randomly chooses $x \in \mathbb{Z}_q^*$ and uses $msk = s$ to compute the ID's partial private key, $D_{ID} = sH_0(ID)$. It thus gives D_{ID} to ID via a secure channel.

Set-Secret-Value: The user ID selects $r_{ID} \in \mathbb{Z}_q^*$ at random and sets r_{ID} as his secret value.

Set-Secret-Key: The user ID sets his full secret key, $sk_{ID} = \{D_{ID}, r_{ID}\}$.

Set-Public-Key: Given ID's secret key sk_{ID}, the user ID obtains his public key $pk_{ID} = \{pk_{ID,1}, pk_{ID,2}\}$ where $pk_{ID,1} = r_{ID}D_{ID}$ and $pk_{ID,2} = r_{ID}H_0(ID)$.

Sign: Given a message m and ID's secret key sk_{ID}, the signer/user generates the signature
$\sigma = D_{ID} + \frac{1}{r_{ID}}H_1(m||ID||pk_{ID})$.

Verify: Taking $(m, \sigma, pk_{ID}, ID, param)$ as input, the verifier can check the following two equations hold or not.

$$\hat{e}(P_{pub}, pk_{ID,2}) =?\hat{e}(P, pk_{ID,1}) \text{ and}$$
$$\hat{e}(\sigma, pk_{ID,2}) =?\hat{e}(pk_{ID,1} + H_1(m||ID||pk_{ID}), H_0(ID)).$$

If they hold, the algorithm returns 1; otherwise, returns 0.

Remark 7. *Tso et al.'s scheme is insecure against the SV-Type I adversary. Its formal security proof is done in the paper of [29].*

5.9 The proposed scheme 5 against the SS-SV-Type I adversary

The proposed scheme 5 against the SS-SV-Type I adversary consists of the following algorithms.

Setup: Let $\mathbb{G}_1$ be an additive cyclic group of prime order q, $\mathbb{G}_T$ be a multiplicative cyclic group of the same order, and e be the bilinear pairing where $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Moreover, let $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_1 : \{0,1\}^* \to \mathbb{G}_1$, and $H_2 : \{0,1\}^* \to \mathbb{G}_1$ be cryptographic hash functions. The KGC randomly chooses $s \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, and then sets the master secret key $msk = s$ and the master public key $P_{pub} = sP$. Finally, it announces the system parameter, $param = \{\mathbb{G}_1, \mathbb{G}_T, \hat{e}, H_0, H_1, q, P, P_{pub} = sP\}$

Partial-Private-Key-Extract: Given a user's identity ID, the KGC randomly chooses $x \in \mathbb{Z}_q^*$ and uses $msk = s$ to compute the ID's partial private key, $D_{ID} = x + sH_0(ID, pk_{ID,2}, P_{pub})$. It thus gives D_{ID} and $pk_{ID,2} = xP$ to ID via a secure channel.

Set-Secret-Value: The user ID selects $r_{ID} \in \mathbb{Z}_q^*$ at random and sets r_{ID} as his secret value.

Set-Secret-Key: The user ID sets his full secret key, $sk_{ID} = \{D_{ID}, r_{ID}\}$.

Set-Public-Key: Given ID's secret key sk_{ID}, the user ID obtains his public key $pk_{ID} = \{pk_{ID,1}, pk_{ID,2}\}$ where $pk_{ID,1} = r_{ID}P$.

Sign: Given a message m and ID's secret key sk_{ID}, the signer/user ID sets $T_1 = H_1(m||ID)$ and $T_2 = H_2(m||ID)$. He then generates the signature $\sigma = r_{ID}T_1 + D_{ID}T_2$.

Verify: Taking $(m, \sigma, pk_{ID}, ID, param)$ as input, the verifier sets $h = H_0(ID, pk_{ID,2}, P_{pub})$, $T_1 = H_1(m||ID)$ and $T_2 = H_2(m||ID)$, and then can check the following equation holds or not, $\hat{e}(\sigma, P) =? \hat{e}(pk_{ID,1}, T_1)\hat{e}(pk_{ID,2} + hP_{pub}, T_2)$. If it holds, the algorithm returns 1; otherwise, returns 0.

Correctness: If the public key PK_{ID} and the signature σ are generated correctly as this scheme, then the correctness holds since

$$\begin{aligned}\hat{e}(\sigma, P) &= \hat{e}(r_{ID}T_1 + D_{ID}T_2, P)\\ &= \hat{e}(r_{ID}T_1, P)\hat{e}(D_{ID}T_2, P)\\ &= \hat{e}(r_{ID}P, T_1)\hat{e}(D_{ID}P, T_2)\\ &= \hat{e}(pk_{ID,1}, T_1)\hat{e}(pk_{ID,2} + hP_{pub}, T_2)\end{aligned}$$

Remark 8. *Scheme 5 is insecure against the SU-Type I adversary. This scheme is modified from Scheme 6 (Sect. 5.10), and its formal security proof is almost the same.*

5.10 The proposed scheme 6 against the S-Type I adversary

The proposed scheme 6 against the S-Type I adversary consists of the following algorithms.

Setup: Let $\mathbb{G}_1$ be an additive cyclic group of prime order q, $\mathbb{G}_T$ be a multiplicative cyclic group of the same order, and e be the bilinear pairing where $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Moreover, let $H_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, and $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$ be cryptographic hash functions. The KGC randomly chooses $s \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, and then sets the master secret key $msk = s$ and the master public key $P_{pub} = sP$. Finally, it announces the system parameter, $param = \{\mathbb{G}_1, \mathbb{G}_T, \hat{e}, H_0, H_1, q, P, P_{pub} = sP\}$

Partial-Private-Key-Extract: Given a user's identity ID, the KGC randomly chooses $x \in \mathbb{Z}_q^*$ and uses $msk = s$ to compute the ID's partial private key, $D_{ID} = x + sH_0(ID||pk_{ID,2}||P_{pub})$. It thus gives D_{ID} and $pk_{ID,2} = xP$ to ID via a secure channel.

Set-Secret-Value: The user ID selects $r_{ID} \in \mathbb{Z}_q^*$ at random and sets r_{ID} as his secret value.

Set-Secret-Key: The user ID sets his full secret key, $sk_{ID} = \{D_{ID}, r_{ID}\}$.

Set-Public-Key: Given ID's secret key sk_{ID}, the user ID obtains his public key $pk_{ID} = \{pk_{ID,1}, pk_{ID,2}\}$ where $pk_{ID,1} = r_{ID}P$.

Sign: Given a message m and ID's secret key sk_{ID}, the signer/user ID sets $T_1 = H_1(m||ID||pk_{ID}||P_{pub})$ and $T_2 = H_2(m||ID||pk_{ID}||P_{pub})$. He then generates the signature $\sigma = r_{ID}T_1 + D_{ID}T_2$.

Verify: Taking $(m, \sigma, pk_{ID}, ID, param)$ as input, the verifier sets $h = H_0(ID||pk_{ID,2}||P_{pub})$, $T_1 = H_1(m||ID||pk_{ID}||P_{pub})$ and $T_2 = H_2(m||ID||pk_{ID}||P_{pub})$, and then can check the following equation holds or not, $\hat{e}(\sigma, P) =?\hat{e}(pk_{ID,1}, T_1)\hat{e}(pk_{ID,2} + hP_{pub}, T_2)$. If it holds, the algorithm returns 1; otherwise, returns 0.

Remark 9. *Scheme 6 is secure against the S-Type I adversary, thus we will give its formal security proof in Section 6.*

6 Security analysis

In Section 3 and 5, we has introduced all possible Type I adversaries and presented the nine CLS schemes. Now we show the security analysis of the proposed scheme 6 in this section. Particularly, we also demonstrate the insecurity of the other schemes in Appendix C; for example, the proposed scheme 2 is secure against the SV-$\mathcal{A}_I$, thus we analyze that it is insecure against the SS-$\mathcal{A}_I$ and SU-$\mathcal{A}_I$.

Theorem 1. *The proposed scheme 6 is provably secure against the adaptively chosen message and identity attacks, performed by the S-Types I and II adversaries, in the random oracle model assuming the CDH problem is intractable.*

This theorem follows from Lemmas 1 and 2 straightly due to two types of adversaries.

Lemma 1. *If there exists an adaptively chosen message and identity S-Type I adversary, $\mathcal{A}_I$, who can ask at most q_C* ***Create-User*** *queries, q_K* ***Partial-Private-Key-Extract*** *queries, and q_S* ***S-Sign*** *queries, and can break the proposed scheme 6 in polynomial time with success probability ε, then there exists an algorithm $\mathcal{C}$ which can depend on $\mathcal{A}_I$'s forgery to solve the CDH problem with probability $Pr[\mathcal{C}(P, aP, bP) \rightarrow abP] \geq (1 - \frac{1}{1-q_C})^{q_K}(1 - \frac{1}{q_S+1})^{q_S}(\frac{1}{q_C(q_S+1)})\varepsilon$.*

Proof. If there exists an S-Type I adversary $\mathcal{A}_I$ who can break the strong unforgeability of the proposed scheme 6 by winning the security game, then we can construct an algorithm $\mathcal{C}$ which can depend on $\mathcal{A}_I$'s forgery to solve the CDH problem as Section 5.1.

Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups with the bilinear map $\hat{e}$. Given P, aP, bP where a, b are unknown, $\mathcal{C}$'s purpose is to compute abP, which is the output of the CDH problem. $\mathcal{C}$ acts as the challenger. $\mathcal{A}_I$ is eligible for accessing the oracles defined in Section 2.1 and 2.2. The three hash functions H_0, H_1, H_2 will be random oracles.

Setup: $\mathcal{C}$ chooses $h^*, v^* \in \mathbb{Z}_q^*$ at random. $\mathcal{C}$ then sets $P_{pub} = \frac{1}{h^*}(bP - v^*P)$ and sends $param = \{\mathbb{G}_1, \mathbb{G}_T, \hat{e}, P, P_{pub}\}$ to $\mathcal{A}_I$.

Query: $\mathcal{A}_I$ can adaptively access the following oracles in a polynomial number of times.

1. **Create-User**: $\mathcal{C}$ maintains K-list which is initially empty. $\mathcal{A}_I$ can submit ID to this oracle. For returning $\mathcal{A}_I$'s request, $\mathcal{C}$ first chooses a number $t \in \{1, ..., q_C\}$ at random.

   (1) If $i \neq t$, $\mathcal{C}$ randomly chooses $v_i, v_i', r_{ID_i} \in \mathbb{Z}_q^*$ and sets $H_0(ID_i||pk_{ID_i,2}||P_{pub}) = v_i'$, $D_{ID_i,1} = v_i$, $D_{ID_i,2} = pk_{ID_i,2} = v_iP - v_i'(P_{pub})$, $pk_{ID_i,1} = r_{ID_i}P$, and the secret value r_{ID_i}.

   (2) If $i = t$, $\mathcal{C}$ randomly chooses $r_{ID_t}, v_t, v_t' \in \mathbb{Z}_q^*$ and sets $H_0(ID_t||pk_{ID_t,2}||P_{pub}) = v_t'$, $D_{ID_t,1} = \perp$, $D_{ID_t,2} = pk_{ID_t,2} = v_tP$, $pk_{ID_t,1} = r_{ID_t}P$, and the secret value r_{ID_t}.[2] (Hereafter, $\perp$ means that nothing is set or returned.)

   In both cases, $\mathcal{C}$ will add the outputted tuple $(ID_i, H_0(ID_i, P_{pub}), D_{ID_i}, r_{ID_i}, pk_{ID_i,1}, pk_{ID_i,2})$ on K-List. If $\mathcal{A}_I$ submits ID_i to ask for the public key or $H_0(ID_i, P_{pub})$, $\mathcal{C}$ returns $pk_{ID_i,1}, pk_{ID_i,2}$ or $H_0(ID_i||pk_{ID_i,2}||P_{pub})$ according to K-list.

2. **Partial-Private-Key-Extract**: $\mathcal{A}_I$ can submit ID_i to this oracle. $\mathcal{C}$ outputs $\perp$ if ID_i has not been created. Else, if ID_i has been created and $i \neq t$, $\mathcal{C}$ returns D_{ID_i} from K-list; otherwise, $\mathcal{C}$ returns failure and terminates.

3. **Public-Key-Replace**: $\mathcal{A}_I$ can submit $(pk'_{ID_i,1}, pk'_{ID_i,2})$ to this oracle for replacing the public key. If ID_i has been created, $\mathcal{C}$ replaces the original $(pk_{ID_i,1}, pk_{ID_i,2})$ with the new $(pk'_{ID_i,1}, pk'_{ID_i,2})$; otherwise, it outputs $\perp$.

4. **Secret-Value-Extract**: $\mathcal{A}_I$ can submit ID_i to this oracle. $\mathcal{C}$ outputs $\perp$ if ID_i has not been created. Else, $\mathcal{C}$ returns r_{ID_i} from K-list.

5. H_1 **queries**: $\mathcal{C}$ maintains H_1-list which is initially empty. $\mathcal{A}_I$ can submit $M_i = (m_j, ID_k, pk_{ID_k}, P_{pub})$ to the random oracle H_1. $\mathcal{C}$ outputs $\perp$ if ID_i has not been created. Otherwise, $\mathcal{C}$ performs as follows for the request M_i. $\mathcal{C}$ randomly chooses $y_i \in \mathbb{Z}_q^*$ and sends $Y_i = y_iP$ as $H_1(M_i)$ to $\mathcal{A}_I$. Finally, $\mathcal{C}$ adds the outputted tuple (M_i, y_i, Y_i) on H_1-list.

[2] ID_i can receive many partial private key because of replacing $pk_{ID_i,2}$. Therefore, for ID_t, it is possible that $D_{ID_t,2} = pk_{ID_t,2} = v^*P$ and $H_0(ID_t||pk_{ID_t,2}||P_{pub}) = h^*$ accordingly.

6. H_2 **queries**: $\mathcal{C}$ maintains H_2-list which is initially empty. $\mathcal{A}_I$ can submit $M_i = (m_j, ID_k, pk_{ID_k}, P_{pub})$ to the random oracle H_2. $\mathcal{C}$ outputs $\perp$ if ID_i has not been created. Otherwise, $\mathcal{C}$ performs as follows for the request M_i.

   - If $ID_k \neq ID_t$, $\mathcal{C}$ randomly chooses $\alpha_i \in \mathbb{Z}_q^*$ and sends $R_i = \alpha_i P$ as $H_2(M_i)$ to $\mathcal{A}_I$. $\mathcal{C}$ therefore adds the outputted tuple $(M_i, \alpha_i, R_i, c_i = \perp)$ on H_2-list.
   - Otherwise, $ID_k = ID_t$, $\mathcal{C}$ randomly chooses $\alpha_i \in \mathbb{Z}_q^*$ and flips a biased-coin, $c_i \in \{0,1\}$, with $\Pr[c_i = 1] = \beta$ and $\Pr[c_i = 0] = 1 - \beta$. (The value, $\beta < 1$, will be considered later.) In the case of $c_i = 1$, $\mathcal{C}$ sends $R_i = \alpha_i(aP)$ as $H_2(M_i)$ to $\mathcal{A}_I$. In the case of $c_i = 0$, $\mathcal{C}$ sends $R_i = \alpha_i P$ as $H_2(M_i)$ to $\mathcal{A}_I$. Finally, $\mathcal{C}$ adds the outputted tuple $(M_i, \alpha_i, R_i, c_i)$ on H_2-list.

7. **S-Sign**: $\mathcal{A}_I$ can submit $\gamma_i = (ID_k, m_j)$ as a signature query. $\mathcal{C}$ outputs $\perp$ if ID_k has not been created. Otherwise, $\mathcal{C}$ performs as follows for (ID_k, m_j) according to K, H_1, H_2-lists.

   - If $ID_k \neq ID_t$, $\mathcal{C}$ generates the signature by $\sigma_i = y_i \cdot pk_{ID_k,1} + D_{ID_k} R_i$.
   - If $ID_k = ID_t$ and $c_i = 0$, $\mathcal{C}$ generates the signature $\sigma_i = y_i \cdot pk_{ID_t,1} + \alpha_i(pk_{ID_t,2} + v_t'(P_{pub}))$.
   - If $ID_k = ID_t$ and $c_i = 1$, $\mathcal{C}$ returns failure and terminates. In this case of $c_i = 1$, $\mathcal{C}$ must make sure that $D_{ID_t,2} = pk_{ID_t,2} = v^* P$ and $H_0(ID_t || pk_{ID_t,2} || P_{pub}) = h^*$.

Forgery: After all queries, $\mathcal{A}_I$ outputs a forgery (m^*, ID^*, σ^*). By assumption, $\mathcal{A}_I$ wins this game because σ^* is valid. If $ID^* \neq ID_t$, $\mathcal{C}$ outputs failure and terminates this game. Otherwise, in the case of $ID^* = ID_t$, $\mathcal{C}$ performs as follows.

(1) $\mathcal{C}$ checks H_2-list. If $c^* = 0$, $\mathcal{C}$ outputs failure and terminates.

(2) Otherwise, in the case of $c^* = 1$, $\mathcal{C}$ depends on $\mathcal{A}_I$'s forgery to solve the CDH problem. Since σ^* is valid, we suppose the following equation holds,

$$\begin{aligned}\hat{e}(\sigma^*, P) \;\; = \;\; & \hat{e}(pk_{ID^*,1}, H_1(m^* || ID^* || pk_{ID^*} || P_{pub})) \cdot \\ & \hat{e}(pk_{ID^*,2}, H_2(m^* || ID^* || pk_{ID^*} || P_{pub})) \cdot \\ & \hat{e}(hP_{pub}, H_2(m^* || ID^* || pk_{ID^*} || P_{pub})) \\ & \text{where } h = H_0(ID^* || pk_{ID^*,2} || P_{pub}).\end{aligned}$$

Based on K, H_1, H_2-lists, the forged signature must be $\sigma^* = y^* \cdot pk_{ID_t,1} + \alpha^*(abP)$ where y^* is obtained from H_1-list, α^* from H_2-list, and $pk_{ID_t,1}$ from K-list. Eventually, $\mathcal{C}$ utilizes σ^* to solve the CDH problem and output $abP = \frac{1}{\alpha^*}(\sigma^* - y^* \cdot pk_{ID_t,1})$.

The algorithm $\mathcal{C}$ is done through the above simulation, which remains to compute the probability that $\mathcal{C}$ solves the CDH problem. Hence, we show the three events if $\mathcal{C}$ succeeds.

- $\mathcal{E}_1$: $\mathcal{C}$ does not abort in the *Query* phase.
- $\mathcal{E}_2$: The forged signature σ^* is valid on (m^*, ID^*, pk_{ID^*}).
- $\mathcal{E}_3$: $\mathcal{C}$ does not abort in the *Forgery* phase.

The probability of $\mathcal{C}$ is $\Pr[\mathcal{C}(P, aP, bP) \rightarrow abP] = \Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] = \Pr[\mathcal{E}_1]\Pr[\mathcal{E}_2]\Pr[\mathcal{E}_3]$ because $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ are independent.

Claim 1. *$\mathcal{C}$ does not abort in the* Query *phase with $Pr[\mathcal{E}_1] \geq (1 - \frac{1}{q_C})^{q_K}(1-\beta)^{q_S}$.*

$\mathcal{C}$ does not output failure in **Partial-Private-Key-Extract** with probability $(1 - \frac{1}{q_C})^{q_K}$, and does not output failure in **S-Sign** with probability $(1 - (\frac{1}{q_C})\beta)^{q_S} \geq (1-\beta)^{q_S}$. Hence, $\Pr[\mathcal{E}_1] \geq (1 - \frac{1}{q_C})^{q_K}(1-\beta)^{q_S}$.

In addition, $\Pr[\mathcal{E}_2] = \varepsilon$ and $\Pr[\mathcal{E}_3] = \beta/q_C$. The probability of $\mathcal{C}$ is $\Pr[\mathcal{C}(P, aP, bP) \rightarrow abP] \geq (1 - \frac{1}{q_C})^{q_K}(1-\beta)^{q_S}(\frac{\beta}{q_C})\varepsilon$. However, $\beta(1-\beta)^{q_S}$ could be maximized at $\beta = \frac{1}{1+q_S}$, so $\Pr[\mathcal{C}(P, aP, bP) \rightarrow abP] \geq (1 - \frac{1}{q_C})^{q_K}(1 - \frac{1}{1+q_S})^{q_S}(\frac{1}{q_C(1+q_S)})\varepsilon$. On the other hand, for the performance, τ is denoted by the running time of $\mathcal{A}_I$, and τ' of $\mathcal{C}$. $\mathcal{A}_I$ can ask at the most q_{H_1} H_1 queries and q_{H_2} H_2 queries where $q_{H_1} = q_{H_2} = q_S + 1$. We conclude $\tau' \leq \tau + 2q_C\tau_{sm} + q_{H_1}\tau_{sm} + q_{H_2}\tau_{sm} + q_S\tau_{sm} = \tau + (2q_C + 3q_S + 2)\tau_{sm}$. The proof of this lemma is complete. □

Lemma 2. *If there exists an adaptively chosen message and identity S-Type II adversary, $\mathcal{A}_{II}$, who can ask at most q_C **Create-User** queries, q_V **Secret-Value-Extract** queries, and q_S **S-Sign** queries, and can break the proposed scheme 6 in polynomial time with success probability ε, then there exists an algorithm $\mathcal{C}$ which can depend on $\mathcal{A}_{II}$'s forgery to solve the CDH problem with probability $Pr[\mathcal{C}(P, aP, bP) \rightarrow abP] \geq (1 - \frac{1}{1-q_C})^{q_V}(1 - \frac{1}{q_S+1})^{q_S}(\frac{1}{q_C(q_S+1)})\varepsilon$.*

The proof of Lemma 2 is similar to that of Lemma 1. We can refer to Appendix D for more details.

7 Discussions

7.1 Shim's attack against short CLS schemes

In 2009, Shim [28] reflected on the possibility that short CLS schemes might be insecure against her presented attack. Performing this attack, a Type I adversary $\mathcal{A}_I$ first sets a new secret value r'_{ID} of ID and computes the new corresponding public key pk'_{ID}. Secondly $\mathcal{A}_I$ replaces the old public key pk_{ID} with the new one pk'_{ID}, and then submits (m, ID) to **Sign** oracle. Upon receiving the signature σ of (m, ID), $\mathcal{A}_I$ can indirectly compute the partial private key D_{ID}. $\mathcal{A}_I$ can thus forge any signature existentially by using D_{ID}. As a result, Shim considered the short CLS schemes might suffer from such attacks since those schemes are deterministic short signature schemes without using random factors.

According to the security models mentioned in Section 3, the SS-Type I, SS-SV-Type I, SS-SU-Type I, and S-Type I adversaries can access **S-Sign**, and they can therefore perform Shim's attack. However, there are some short CLS schemes such as Tso et al.'s scheme [29] and our proposed schemes 3, 5, and 6, which have been proven to be secure against these kinds of Type I adversaries. As we know, some short CLS schemes [7, 9, 11, 18, 19, 30, 31] are insecure against such attacks undoubtedly. However, this attack does not succeed to break all of short CLS schemes; for example, it can be withstood by the proposed scheme 6.

7.2 The relation between strong unforgability and non-repudiation in short CLS schemes

Girault defined three trust levels for a trusted third party (TTP) [14]. However, the higher the trust level of the TTP is, the higher the security level of the cryptographic scheme becomes. Explicitly, based on the definition of Girault, Hu et al. [17] stated clearly the three trust levels of the KGC in the context of certificateless signature schemes:

- Level 1. The KGC knows the full private key of any user and is able to act as any user to forge signatures which cannot be repudiated by that user (the victim).
- Level 2. The KGC does not know the full private key of any user. But the KGC is able to generate a false private key for any user to forge signatures which cannot be repudiated by that user (the victim).

- Level 3. The KGC does not know the full private key of any user. But the KGC is able to generate a false private key of any user to forge signatures but that user (the victim) can repudiate these forged signatures.

From a legal viewpoint, using a digital signature scheme with trust level 1 or 2, a signer can always repudiate the signatures by blaming the KGC. A CLS scheme is said to provide *non-repudiation* if the KGC is of trust level 3. In general, a CLS scheme meets trust level 3, which implies that only a user has one unique public/private key pair, and thereby is unable to generate another key pair himself. To prove a CLS scheme with trust level 3, we usually use an analysis in which a user cannot output another key pair by replacing the public key. We now conclude that a short CLS scheme is strongly unforgeable against the SU-Type I adversary if it is at trust level 3. Since no random factors are involved and the adversary cannot replace the public key, the short CLS scheme with trust level 3 is strongly unforgeable against SU-$\mathcal{A}_I$ without doubt. As a result, this kind of short CLS schemes avoids the only ability of the SU-$\mathcal{A}_I$, i.e. replacing a public key.

7.3 Comparisons

Finally, we compare our schemes with the other short certificateless signature schemes [7, 9, 11, 18, 19, 29, 30, 31]. The comparisons are given in Table 3 with respect to their efficiency and security (we do not consider the precomputations herein).[3] The computation cost of bilinear pairing is denoted by $\mathcal{P}$, and that of scalar multiplication of $\mathbb{G}_1$ is denoted by $\mathcal{S}$. In addition, we use ✓ to represent that the scheme is secure against this kind of $\mathcal{A}_I$.

As shown in Table 3, although the signature generations of the proposed schemes 5 and 6 are not as efficient as those of the schemes [9, 11, 18, 19, 29, 30, 31], the proposed scheme 6 can achieve the higher security level. However, Fan et al.'s scheme [11] is insecure against the N-$\mathcal{A}_I$. In particular, Choi et al. claimed that their scheme can withstand the SS-SV-$\mathcal{A}_I$ [7]; however, this scheme has been cryptanalyzed to be secure against only the N-$\mathcal{A}_I$ [6]. As a result, the proposed scheme 6 is proven to be secure against the S-Type I adversary, with the formal security proof provided in Section 6. In fact, the open problem of short CLS (described in Section 5) has been solved since in the proposed schemes 2, 4 ,5, and 6, presented in this paper.

[3] In Table 3, we do not compare the communication costs since the schemes are all short signatures.

Table 3: Efficiency and security comparisons with our proposed schemes and others

Scheme	*Sign*	*Verify*	N-$\mathcal{A}_I$	SV-$\mathcal{A}_I$	SU-$\mathcal{A}_I$	SS-$\mathcal{A}_I$	SV-SU-$\mathcal{A}_I$	SS-SU-$\mathcal{A}_I$	SS-
CPL [7]	$3\mathcal{S}$	$3\mathcal{P}$	✓		✓				
DW [9]	$\mathcal{S}$	$\mathcal{P}$	✓		✓				
FHH [11]	$\mathcal{S}$	$\mathcal{P}$							
HMSWW [18, 19]	$\mathcal{S}$	$2\mathcal{P}$	✓		✓				
THS [29]	$\mathcal{S}$	$4\mathcal{P}$	✓		✓	✓		✓	
TYH [30, 31]	$\mathcal{S}$	$4\mathcal{P}$	✓		✓				
Scheme 1	$\mathcal{S}$	$2\mathcal{P}$	✓						
Scheme 2	$\mathcal{S}$	$2\mathcal{P}$	✓	✓					
Scheme 3	$\mathcal{S}$	$5\mathcal{P}$	✓			✓			
Scheme 4	$\mathcal{S}$	$2\mathcal{P}$	✓	✓	✓		✓		
Scheme 5	$2\mathcal{S}$	$3\mathcal{P}$	✓	✓		✓			
Scheme 6	$2\mathcal{S}$	$3\mathcal{P}$	✓	✓	✓	✓	✓	✓	

8 Conclusions

Cryptographic schemes are dependable for realizing secure applications. Security models are given to simulate behaviors and attack powers of different adversaries. By the formal security proof, schemes are claimed to be secure against the adversary under the security model. Hence, such security models are very important since they are not used only to prove the security in theory, but also to preconsider potential attacks in practice.

In this paper, we revisited certificateless signatures for public key replacement and strong unforgability, which have not been considered in depth in the literature. The simulations of potential adversaries resulted in eight different kinds of Type I adversaries. We reviewed and surveyed some schemes and proposed six schemes. Moreover, we proved their security or insecurity against one kind of Type I adversaries. The proposed scheme 6 is the only certificateless short signature scheme that reaches the strongest security level, which is provably secure against both S-Types I and II adversaries. Finally, some research results were presented including the relation between strong unforgability and non-repudiation in certificateless short signatures.

References

[1] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *Proc. ASIACRYPT 2003*, pages 452–473, Taipei, Taiwan, 2003. Lecture Notes in Computer Science 2894, Springer.

[2] D. Boneh and X. Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008.

[3] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Proc. ASIACRYPT 2001*, pages 514–532. Lecture Notes in Computer Science 2248, Springer, 2001.

[4] S. Chang, D. S. Wong, Y. Mu, and Z. Zhang. Certificateless threshold ring signature. *Information Sciences*, 179(20):3685–3696, 2009.

[5] Y. C. Chen, C. L. Liu, G. Horng, and K. C. Chen. A provably secure certificateless proxy signature scheme. *International Journal of Innovative Computing, Information and Control*, 7(9):5557–5569, 2011.

[6] Y. C. Chen, R. Tso, and G. Horng. Cryptanalysis of a provably secure certificateless short signature scheme. In *The 2012 International Computer Symposium*, Hualien, Taiwan, 2012.

[7] K. Y. Choi, J. H. Park, and D. H. Lee. A new provably secure certificateless short signature scheme. *Computers and Mathematics with Applications*, 61(7):1760–1768, 2011.

[8] A. W. Dent. A survey of certificateless encryption schemes and security models. *International Journal of Information Security*, 7(5):349–377, 2008.

[9] H. Du and Q. Wen. Efficient and provably-secure certificateless short signature scheme from bilinear pairings. *Computer Standards and Interfaces*, 31(2):390–394, 2009.

[10] S. Duan. Certificateless undeniable signature scheme. *Information Sciences*, 178(3):742–755, 2008.

[11] C. I. Fan, R. H. Hsu, and P. H. Ho. Truly non-repudiation certificateless short signature scheme from bilinear pairings. *Journal of Information Science and Engineering*, 27(3):969–982, 2011.

[12] D. Fiore, R. Gennaro, and N. P. Smart. Relations between the security models for certificateless encryption and ID-based key agreement. *International Journal of Information Security*, 11(1):1–22, 2012.

[13] C. Gentry. Certificate-based encryption and the certificate revocation problem. In *Proc. EUROCRYPT 2003*, pages 272–293, Warsaw, Porland. Lecture Notes in Computer Science 2656, Springer.

[14] M. Girault. Self-certified public keys. In *Proc. EUROCRYPT 1991*, pages 490–497. Lecture Notes in Computer Science 549, Springer.

[15] M. C. Gorantla and A. Saxena. An efficient certificateless signature scheme. In *Proc. CSI 2005*, pages 110–116, Las Vegas, NV, USA. Lecture Notes in Computer Science 3802, Springer.

[16] B. C. Hu, D. S. Wong, Z. Zhang, and X Deng. Key replacement attack against a generic construction of certificateless signature. In *Proc. ACISP 2006*, pages 235–246, Melbourne, Australia, 2006. Lecture Notes in Computer Science 4058, Springer.

[17] B. C. Hu, D. S. Wong, Z. Zhang, and X Deng. Certificateless signature: a new security model and an improved generic construction. *Designs, Codes and Cryptography*, 42(2):109–126, 2007.

[18] X. Huang, Y. Mu, W. Susilo, D. S. Wong, and W Wu. Certificateless signature revisited. In *Proc. ACISP 2007*, pages 308–322, Townsville, QLD, Australia, 2007. Lecture Notes in Computer Science 4586, Springer.

[19] X. Huang, Y. Mu, W. Susilo, D. S. Wong, and W Wu. Certificateless signatures: New schemes and security models. *Computer Journal*, 55(4):457–474, 2012.

[20] X. Huang, W. Susilo, Y. Mu, and F. Zhang. On the security of certificateless signature schemes from Asiacrypt 2003. In *Proc. CANS 2005*, pages 13–25, Xiamen, China, 2005. Lecture Notes in Computer Science 3810, Springer.

[21] Y. H. Hwang, J. K. Liu, and S. S. M. Chow. Certificateless public key encryption secure against malicious KGC attacks in the standard model. *Journal of Universal Computer Science*, 14(3):463–480, 2008.

[22] D. Jao and K. Yoshida. Boneh-boyen signatures and the strong diffiehellman problem. In *Proc. Pairing-Based Cryptography - Pairing 2009*, pages 1–16. Lecture Notes in Computer Science 5671, Springer, 2009.

[23] D. P. Le and C. L. Liu. Refinements of Miller's algorithm over weierstrass curves revisited. *Computer Journal*, 54(10):1582–1591, 2011.

[24] X. Li, K. Chen, and L. Sun. Certificateless signature and proxy signature schemes from bilinear pairings. *Lithuanian Mathematical Journal*, 45(1):76–83, 2005.

[25] C. L. Liu, G. Horng, and T. Y. Chen. Further refinement of pairing computation based on Miller's algorithm. *Applied Mathematics and Computation*, 189(1):395–409, 2007.

[26] S. H. Seo, K. Y. Choi, J. Y. Hwang, and S. Kim. Efficient certificateless proxy signature scheme with provable security. *Information Sciences*, 188(1):322–337, 2010.

[27] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. CRYPTO 1984*, pages 19–22, Santa Barbara, CA, USA, 1985. Lecture Notes in Computer Science 196, Springer.

[28] K. A. Shim. Breaking the short certificateless signature scheme. *Information Sciences*, 179(3):303–306, 2009.

[29] R. Tso, X. Huang, and W. Susilo. Strongly secure certificateless short signatures. *Journal of Systems and Software*, 85(6):1409–1417, 2012.

[30] R. Tso, X. Yi, and X. Huang. Efficient and short certificateless signature. In *Proc. CANS 2008*, pages 64–79, Hong-Kong, China. Lecture Notes in Computer Science 5339, Springer.

[31] R. Tso, X. Yi, and X. Huang. Efficient and short certificateless signatures secure against realistic adversaries. *Journal of Supercomputing*, 55(2):173–191, 2011.

[32] L. Wang, Z. Cao, X. Li, and H. Qian. Simulatability and security of certificateless threshold signatures. *Information Sciences*, 177(6):1382–1394, 2007.

[33] G. Yang and C. H. Tan. Certificateless public key encryption: A new generic construction and two pairing-free schemes. *Theoretical Computer Science*, 412(8–10):662–674, 2011.

[34] W. S. Yap, S. H. Heng, and B. M. Goi. An efficient certificateless signature scheme. In *Proc. EUC 2006*, pages 322–331, Seoul, Korea. Lecture Notes in Computer Science 4097, Springer.

[35] H. Yuan, F. Zhang, X. Huang, Y. Mu, W. Susilo, and L. Zhang. Certificateless threshold signature scheme from bilinear maps. *Information Sciences*, 180(23):4714–4728, 2010.

[36] D. H. Yum and P. J. Lee. Generic construction of certificateless signature. In *Proc. ACISP 2004*, pages 200–211, Sydney, Australia, 2004. Lecture Notes in Computer Science 3108, Springer.

[37] Z. Zhang, D. S. Wong, J. Xu, and D. Feng. Certificateless public-key signature: Security model and efficient construction. In *Proc. ACNS 2006*, pages 293–308, Melbourne, Australia. Lecture Notes in Computer Science 4058, Springer.

A Other Type I adversaries

A.1 Security against the SU-Type I adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity SU-Type I adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* including the master public key to $\mathcal{A}_I$.

Query: $\mathcal{A}_I$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, **Secret-Value-Extract**, and **Partial-Private-Key-Extract**. Moreover, $\mathcal{A}_I$ can submit queries to the **N-Sign** oracle.

Forgery: $\mathcal{A}_I$ outputs a forged triplet (σ^*, m^*, ID^*). $\mathcal{A}_I$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) can be submitted to **N-Sign**.

2. σ^* has never returned by **N-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s current public key.

3. ID^* has never been submitted to **Partial-Private-Key-Extract** or **Secret-Value-Extract**.

A.2 Security against the SS-Type I adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity SS-Type I adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* including the master public key to $\mathcal{A}_I$.

Query: $\mathcal{A}_I$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, **Secret-Value-Extract**, and **Partial-Private-Key-Extract**. Moreover, $\mathcal{A}_I$ can submit queries to the **S-Sign** oracle.

Forgery: $\mathcal{A}_I$ outputs a forged triplet (σ^*, m^*, ID^*). $\mathcal{A}_I$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) has never been submitted to **S-Sign**.

2. σ^* has never returned by **S-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s current public key.

3. ID^* has never been submitted to **Partial-Private-Key-Extract** or **Secret-Value-Extract**.

A.3 Security against the SV-SU-Type I adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity SV-SU-Type I adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* including the master public key to $\mathcal{A}_I$.

Query: $\mathcal{A}_I$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, **Secret-Value-Extract**, and **Partial-Private-Key-Extract**. Moreover, $\mathcal{A}_I$ can submit queries to the **N-Sign** oracle.

Forgery: $\mathcal{A}_I$ outputs a forged triplet (σ^*, m^*, ID^*). $\mathcal{A}_I$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) can be submitted to **N-Sign**.

2. σ^* has never returned by **N-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s current public key.

3. ID^* has never been submitted to **Partial-Private-Key-Extract**.

A.4 Security against the SS-SV-Type I adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity SS-SV-Type I adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* including the master public key to $\mathcal{A}_I$.

Query: $\mathcal{A}_I$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, **Secret-Value-Extract**, and **Partial-Private-Key-Extract**. Moreover, $\mathcal{A}_I$ can also submit queries to the **S-Sign** oracle.

Forgery: $\mathcal{A}_I$ outputs a forged triplet (σ^*, m^*, ID^*). $\mathcal{A}_I$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) has never been submitted to **S-Sign**.
2. σ^* has never returned by **S-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s current public key.
3. ID^* has never been submitted to **Partial-Private-Key-Extract**.

B Other Type II adversaries

B.1 Security against the SU-Type II adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity SU-Type II adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* and the master key *msk* to $\mathcal{A}_{II}$.

Query: In this phase, $\mathcal{A}_{II}$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, and **Secret-Value-Extract**. Moreover, $\mathcal{A}_I$ can submit queries to the **N-Sign** oracle.

Forgery: $\mathcal{A}_{II}$ outputs a forged triplet (σ^*, ID^*, m^*). $\mathcal{A}_{II}$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) can be submitted to **N-Sign**.
2. σ^* has never returned by **N-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s original public key.
3. ID^* has never been submitted to **Secret-Value-Extract**.

B.2 Security against the SS-Type II adversary

The unforgeability of a CLS scheme against the adaptive chosen message and identity N-Type II adversary is defined by the following game:

Setup: The challenger runs the algorithm Setup, and then returns the system parameters *param* and the master key *msk* to $\mathcal{A}_{II}$.

Query: In this phase, $\mathcal{A}_{II}$ can adaptively send queries to **Create-User**, **Public-Key-Replace**, and **Secret-Value-Extract**. Moreover, $\mathcal{A}_I$ can submit queries to the **S-Sign** oracle.

Forgery: $\mathcal{A}_{II}$ outputs a forged triplet (σ^*, ID^*, m^*). $\mathcal{A}_{II}$ is said to win the game if the following conditions hold.

1. (ID^*, m^*) has never been submitted to **S-Sign**.

2. σ^* has never returned by **S-Sign** and $1 \leftarrow \mathsf{Verify}(param, ID^*, pk_{ID^*}, m^*, \sigma^*)$ where pk_{ID^*} is the ID^*'s original public key.

3. ID^* has never been submitted to **Secret-Value-Extract**.

C Insecurity of some schemes

C.1 Insecurity of Fan et al.'s scheme

This scheme is insecure against the N-Type I adversary's attack. $\mathcal{A}_I$ first picks m^* and sets $h^* = H_1(m^*, pk_{ID^*,1})$. Secondly, he chooses $t \in \mathbb{Z}_q^*$ at random, and then replaces $pk_{ID^*,2}$ with a new one $pk'_{ID^*,2} = tP - H_0(ID^*||pk_{ID^*,1})pk_{ID^*,1} - h(P_{pub} + H_0(ID^*)P_2 + H_0(ID^*||pk_{ID^*,1})P_2)$. $\mathcal{A}_I$ finally can output a forged signature $\sigma^* = t^{-1}P_1$. As a result, (ID^*, m^*) has never been submitted to Sign oracle.

C.2 Insecurity of the proposed scheme 1

This scheme is only secure against the N-Type I adversary, thus it cannot withstand the SS-Type I, SU-Type I, or SV-Type I adversaries' attacks.

- SS-$\mathcal{A}_I$ randomly chooses $t \in \mathbb{Z}_q^*$ and computes $pk'_{ID^*} = tP$, and then submits (ID^*, pk'_{ID^*}) to **Public-Key-Replace**. He sends (ID^*, m) to **S-Sign**, and then receives σ of (ID^*, m) where $\sigma = tH_1(m||ID^*) + D_{ID^*}$. Eventually, he can obtain the partial-private-key $D_{ID^*} = \sigma - tH_1(m||ID^*)$. If $\mathcal{A}_I$ has D_{ID^*}, he can generate any forged signature of (ID^*, m^*).

- SU-$\mathcal{A}_I$ sends (ID^*, m^*) to **N-Sign** and obtains σ of (ID^*, m^*). He randomly chooses $t \in \mathbb{Z}_q^*$ and computes $pk'_{ID^*} = pk_{ID^*} + tP$, and then submits (ID^*, pk'_{ID^*}) to **Public-Key-Replace**. Eventually, he outputs a forged signature σ^* where $\sigma^* = \sigma + tH_1(m^*||ID^*)$. σ^* is valid and has never been returned by **N-Sign**.

- SV-$\mathcal{A}_I$ first submits ID^* to **Secret-Value-Extract**, and then receives r_{ID^*}. He sends (ID^*, m) to **N-Sign**, and then receives σ of (ID^*, m) where $\sigma = r_{ID^*}H_1(m||ID^*) + D_{ID^*}$. Eventually, he can obtain the partial-private-key $D_{ID^*} = \sigma - r_{ID^*}H_1(m||ID^*)$. If $\mathcal{A}_I$ has D_{ID^*}, he can generate any forged signature of (ID^*, m^*).

C.3 Insecurity of the proposed scheme 2

This scheme is only secure against the SV-Type I adversary, thus it cannot withstand the SS-Type I or SU-Type I adversaries's attacks. Now we present them respectively in details.

- SS-$\mathcal{A}_I$ first randomly chooses $r'_{ID^*}, t \in \mathbb{Z}_q^*$ and sends (ID^*, pk'_{ID^*}) as the new public key to **Public-Key-Replace** where $pk'_{ID^*,1} = r'_{ID^*}P$ and $pk'_{ID^*,2} = tP - H_0(ID^*)P_{pub1}$. $\mathcal{A}_I$ submits (m, ID^*) to **S-Sign**, and then obtains the signature σ. Therefore $\mathcal{A}_I$ computes $D_{ID^*,2} = (H_2(m||ID^*)r'_{ID^*} + t)\sigma$ since σ is a valid one. Finally, $\mathcal{A}_I$ can generate a forged signature σ^* on (ID^*, m^*) by computing $\sigma^* = \frac{1}{H_2(m^*||ID^*)r'_{ID^*}+t}D_{ID^*,2}$.

- SU-$\mathcal{A}_I$ first sets $h = H_2(m^*||ID^*)$ and submits (ID^*, m^*) to **N-Sign**, and obtains the signature σ. $\mathcal{A}_I$ randomly chooses $t \in \mathbb{Z}_q^*$ and sends (ID^*, pk'_{ID^*}) as the new public key to **Public-Key-Replace** where $pk'_{ID^*,1} = \frac{1}{t}pk_{ID^*,1}$ and $pk'_{ID^*,2} = \frac{1}{t}pk_{ID^*,2} - H_0(ID^*)P_{pub1} + \frac{1}{t}H_2(ID^*)P_{pub1}$. Finally, $\mathcal{A}_I$ can generate a forged signature $\sigma^* = t\sigma$ of (ID^*, m^*) where σ^* has never been returned by **N-Sign**.

C.4 Insecurity of Huang et al.'s scheme

This scheme is only secure against the SU-Type I adversary, thus it cannot withstand the SS-Type I or SV-Type I adversaries' attacks.

- SS-$\mathcal{A}_I$ randomly chooses $t \in \mathbb{Z}_q^*$ and computes $pk'_{ID^*} = tP$, and then submits (ID^*, pk'_{ID^*}) to **Public-Key-Replace**. He sends (ID^*, m) to **S-Sign**, and then receives σ of (ID^*, m) where $\sigma = tH_1(m||ID^*||pk'_{ID^*}) + D_{ID^*}$. Eventually, he can obtain the partial-private-key $D_{ID^*} = \sigma - tH_1(m||ID^*||pk'_{ID^*})$. If $\mathcal{A}_I$ has D_{ID^*}, he can generate any forged signature of (ID^*, m^*).

- SV-$\mathcal{A}_I$ first submits ID^* to **Secret-Value-Extract**, and then receives r_{ID^*}. He sends (ID^*, m) to **N-Sign**, and then receives σ of (ID^*, m) where $\sigma = r_{ID^*}H_1(m||ID^*||pk'_{ID^*}) + D_{ID^*}$. Eventually, he can obtain the partial-private-key $D_{ID^*} = \sigma - r_{ID^*}H_1(m||ID^*||pk'_{ID^*})$. If $\mathcal{A}_I$ has D_{ID^*}, he can generate any forged signature of (ID^*, m^*).

C.5 Insecurity of the proposed scheme 3

This scheme is only secure against the SS-Type I adversary, thus it cannot withstand the SU-Type I or SV-Type I adversaries' attacks.

- SU-$\mathcal{A}_I$ first submits (ID^*, m^*) to **N-Sign**, and receives the signature σ. $\mathcal{A}_I$ randomly chooses $pk'_{ID^*,3} \in \mathbb{G}_1$ and sends $(ID^*, pk'_{ID^*,1}, pk'_{ID^*,2}, pk'_{ID^*,3})$ as the new public key to **Public-Key-Replace** where $pk'_{ID^*,1} = pk_{ID^*,1}$, $pk'_{ID^*,2} = pk_{ID^*,2}$, and $pk'_{ID^*,3}$. Finally, $\mathcal{A}_I$ can generate a forged signature $\sigma^* = \sigma + pk'_{ID^*,3}$ of (ID^*, m^*) where σ^* has never been returned by **N-Sign**.

- SV-$\mathcal{A}_I$ first submits ID^* to **Secret-Value-Extract**, and then receives r_{ID}. He can obtain the partial-private-key $D_{ID^*} = r_{ID^*}^{-1}(pk_{ID^*,1})$. If $\mathcal{A}_I$ has D_{ID^*}, he can generate any forged signature of (ID^*, m^*).

C.6 Insecurity of the proposed scheme 4

This scheme is only secure against the SV-SU-Type I adversary, thus it cannot withstand the SS-Type I adversary's attacks.

- SS-$\mathcal{A}_I$ first randomly chooses $r'_{ID^*}, t \in \mathbb{Z}_q^*$ and sends (ID^*, pk'_{ID^*}) as the new public key to **Public-Key-Replace** where $pk'_{ID^*,1} = r'_{ID^*}P$ and $pk'_{ID^*,2} = tP - H_0(ID^*)P_{pub1}$. $\mathcal{A}_I$ submits (m, ID^*) to **S-Sign**, and then obtains the signature σ. Therefore $\mathcal{A}_I$ computes $D_{ID^*,2} = (H_2(m||ID^*||pk_{ID})r'_{ID^*} + t)\sigma$ since σ is a valid one. Finally, $\mathcal{A}_I$ can generate a forged signature σ^* on (ID^*, m^*) by computing $\sigma^* = \frac{1}{H_2(m^*||ID^*||pk_{ID})r'_{ID^*} + t} D_{ID^*,2}$.

C.7 Insecurity of Tso et al.'s scheme

This scheme is only secure against the SS-SU-Type I adversary, thus it cannot withstand the SV-Type I adversary' attacks.

- SV-$\mathcal{A}_I$ first submits ID^* to **Secret-Value-Extract**, and then receives r_{ID}. He can obtain the partial-private-key $D_{ID^*} = r_{ID^*}^{-1}(pk_{ID^*,1})$. If $\mathcal{A}_I$ has D_{ID^*}, he can generate any forged signature of (ID^*, m^*).

C.8 Insecurity of the proposed scheme 5

This scheme is only secure against the SS-SV-Type I adversary, thus it cannot withstand the SU-Type I adversary's attacks.

- SU-$\mathcal{A}_I$ first submits (ID^*, m^*) to **N-Sign**, and receives the signature σ. $\mathcal{A}_I$ randomly chooses $t \in \mathbb{Z}_q^*$ and sends (ID^*, pk'_{ID^*}) as the new public key to **Public-Key-Replace** where $pk'_{ID^*} = pk_{ID^*} + tP$. Finally, $\mathcal{A}_I$ can generate a forged signature $\sigma^* = \sigma + tH_1(m^*||ID^*)$ of (ID^*, m^*) where σ^* has never been returned by **N-Sign**.

D Proof of Lemma 2

If there exists an S-Type II adversary $\mathcal{A}_{II}$ who can break the strong unforgeability of the proposed scheme 6 by winning the security game, then we can construct an algorithm $\mathcal{C}$ which can depend on $\mathcal{A}_{II}$'s forgery to solve the CDH problem.

Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups with the bilinear map $\hat{e}$. Given P, aP, bP where a, b are unknown, $\mathcal{C}$'s purpose is to compute abP, which is the output of the CDH problem. $\mathcal{C}$ acts as the challenger. $\mathcal{A}_{II}$ is eligible for accessing the oracles. The three hash functions H_0, H_1, H_2 will be random oracles.

Setup: $\mathcal{C}$ chooses $s \in \mathbb{Z}_q^*$ at random. $\mathcal{C}$ then sets $P_{pub} = sP$ and sends $param = \{\mathbb{G}_1, \mathbb{G}_T, \hat{e}, P, P_{pub}\}$ and the master secret key, $msk = s$, to $\mathcal{A}_{II}$.

Query: $\mathcal{A}_{II}$ can adaptively access the following oracles in a polynomial number of times.

1. **Create-User**: $\mathcal{C}$ maintains K-list which is initially empty. $\mathcal{A}_{II}$ can submit ID_i to this oracle. For $\mathcal{A}_{II}$'s request, $\mathcal{C}$ first chooses a number $t \in \{1, ..., q_C\}$ at random.

   (1) If $i \neq t$, $\mathcal{C}$ randomly chooses $v_i, v_i', r_{ID_i} \in \mathbb{Z}_q^*$ and sets $H_0(ID_i || pk_{ID_i,2} || P_{pub}) = v_i'$, $D_{ID_i,1} = v_i + v_i's$, $D_{ID_i,2} = pk_{ID_i,2} = v_iP$, $pk_{ID_i,1} = r_{ID_i}P$, and the secret value r_{ID_i}.

   (2) If $i = t$, $\mathcal{C}$ randomly chooses $v_t, v_t' \in \mathbb{Z}_q^*$ and sets $H_0(ID_i || pk_{ID_i,2} || P_{pub}) = v_i'$, $D_{ID_i,1} = v_i + v_i's$, $D_{ID_i,2} = pk_{ID_i,2} = v_iP$, $pk_{ID_i,1} = bP$, and the secret value $r_{ID_i} = \perp$.

   In both cases, $\mathcal{C}$ will add the outputted tuple $(ID_i, H_0(ID_i, P_{pub}), D_{ID_i}, r_{ID_i}, pk_{ID_i,1}, pk_{ID_i,2})$ on K-List. If $\mathcal{A}_{II}$ submits ID_i to ask for the public key or $H_0(ID_i, P_{pub})$, $\mathcal{C}$ returns $pk_{ID_i,1}, pk_{ID_i,2}$ or $H_0(ID_i || pk_{ID_i,2} || P_{pub})$ according to K-list.

2. **Public-Key-Replace**: $\mathcal{A}_{II}$ can submit $(pk'_{ID_i,1}, pk'_{ID_i,2})$ to this oracle for replacing the public key. If ID_i has been created, $\mathcal{C}$ replaces the original $(pk_{ID_i,1}, pk_{ID_i,2})$ with the new $(pk'_{ID_i,1}, pk'_{ID_i,2})$; otherwise, it outputs $\perp$.

3. **Secret-Value-Extract**: $\mathcal{A}_{II}$ can submit ID_i to this oracle. $\mathcal{C}$ outputs $\perp$ if ID_i has not been created. Else, if $ID_i = ID_t$, $\mathcal{C}$ returns failure and terminates; otherwise, $\mathcal{C}$ returns r_{ID_i} from K-list.

4. H_1 **queries**: $\mathcal{C}$ maintains H_1-list which is initially empty. $\mathcal{A}_{II}$ can submit $M_i = (m_j, ID_k, pk_{ID_k}, P_{pub})$ to the random oracle H_1. $\mathcal{C}$ outputs $\perp$

if ID_i has not been created. Otherwise, $\mathcal{C}$ performs as follows for the request M_i.

- If $ID_k \neq ID_t$, $\mathcal{C}$ randomly chooses $\alpha_i \in \mathbb{Z}_q^*$ and sends $Y_i = y_i P$ as $H_1(M_i)$ to $\mathcal{A}_{II}$. $\mathcal{C}$ therefore adds the outputted tuple $(M_i, y_i, Y_i, c_i = \perp)$ on H_1-list.
- Otherwise, $ID_k = ID_t$, $\mathcal{C}$ randomly chooses $y_i \in \mathbb{Z}_q^*$ and flips a biased-coin, $c_i \in \{0,1\}$, with $\Pr[c_i = 1] = \beta$ and $\Pr[c_i = 0] = 1 - \beta$. (The value, $\beta < 1$, will be considered later.) In the case of $c_i = 1$, $\mathcal{C}$ sends $Y_i = y_i(aP)$ as $H_1(M_i)$ to $\mathcal{A}_{II}$. In the case of $c_i = 0$, $\mathcal{C}$ sends $Y_i = y_i P$ as $H_1(M_i)$ to $\mathcal{A}_{II}$. Finally, $\mathcal{C}$ adds the outputted tuple (M_i, y_i, Y_i, c_i) on H_1-list.

5. H_2 **queries**: $\mathcal{C}$ maintains H_2-list which is initially empty. $\mathcal{A}_{II}$ can submit $M_i = (m_j, ID_k, pk_{ID_k}, P_{pub})$ to the random oracle H_2. $\mathcal{C}$ outputs $\perp$ if ID_i has not been created. Otherwise, $\mathcal{C}$ performs as follows for the request M_i. $\mathcal{C}$ randomly chooses $\alpha_i \in \mathbb{Z}_q^*$ and sends $R_i = \alpha_i P$ as $H_2(M_i)$ to $\mathcal{A}_{II}$. Finally, $\mathcal{C}$ adds the outputted tuple (M_i, α_i, R_i) on H_2-list.

6. **S-Sign**: $\mathcal{A}_{II}$ can submit $\gamma_i = (ID_k, m_j)$ as a signature query. $\mathcal{C}$ outputs $\perp$ if ID_i has not been created. Otherwise, $\mathcal{C}$ performs as follows for (ID_k, m_j) according to K, H_1, H_2-lists.

   - If $ID_k \neq ID_t$, $\mathcal{C}$ generates the signature $\sigma_i = y_i \cdot pk_{ID_k,1} + D_{ID_k} R_i$.
   - If $ID_k = ID_t$ and $c_i = 0$, $\mathcal{C}$ generates the signature $\sigma_i = y_i \cdot pk_{ID_k,1} + D_{ID_k} R_i$.
   - If $ID_k = ID_t$ and $c_i = 1$, $\mathcal{C}$ returns failure and terminates.

Forgery: After all queries, $\mathcal{A}_{II}$ outputs a forgery (m^*, ID^*, σ^*). By assumption, $\mathcal{A}_{II}$ wins this game because σ^* is valid where pk_{ID^*} is the original public key. If $ID^* \neq ID_t$, $\mathcal{C}$ outputs failure and terminates this game. Otherwise, in the case of $ID^* = ID_t$, $\mathcal{C}$ performs as follows.

(1) $\mathcal{C}$ checks H_2-list. If $c^* = 0$, $\mathcal{C}$ outputs failure and terminates.

(2) Otherwise, in the case of $c^* = 1$, $\mathcal{C}$ depends on $\mathcal{A}_{II}$'s forgery to solve the CDH problem. Since σ^* is valid and $pk_{ID^*,1} = aP$ is the original public key, we suppose the following equation holds,

$$\begin{aligned}
\hat{e}(\sigma^*, P) &= \hat{e}(pk_{ID^*,1}, H_1(m^* || ID^* || pk_{ID^*} || P_{pub})) \cdot \\
&\quad \hat{e}(pk_{ID^*,2}, H_2(m^* || ID^* || pk_{ID^*} || P_{pub})) \cdot \\
&\quad \hat{e}(hP_{pub}, H_2(m^* || ID^* || pk_{ID^*} || P_{pub})) \\
&\quad \text{where } h = H_0(ID^* || pk_{ID^*,2} || P_{pub}).
\end{aligned}$$

Based on K, H_1, H_2-lists, the forged signature can be transformed into $\sigma^* = y^*(abP) + \alpha^* v_t P + \alpha^* v_t' P_{pub}$ where y^* is obtained from H_1-list, α^* from H_2-list, and v_t, v_t' from K-list. Eventually, $\mathcal{C}$ utilizes σ^* to solve the CDH problem and output $abP = \frac{1}{y^*}(\sigma^* - \alpha^* v_t P - \alpha^* v_t' P_{pub})$.

The algorithm $\mathcal{C}$ is done through the above simulation, which remains to compute the probability that $\mathcal{C}$ solves the CDH problem. Hence, we show the three events if $\mathcal{C}$ succeeds.

- $\mathcal{E}_1$: $\mathcal{C}$ does not abort in the *Query* phase.
- $\mathcal{E}_2$: The forged signature σ^* is valid on (m^*, ID^*, pk_{ID^*}).
- $\mathcal{E}_3$: $\mathcal{C}$ does not abort in the *Forgery* phase.

The probability of $\mathcal{C}$ is $\mathsf{Pr}[\mathcal{C}(P, aP, bP) \to abP] = \mathsf{Pr}[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] = \mathsf{Pr}[\mathcal{E}_1]\mathsf{Pr}[\mathcal{E}_2]\mathsf{Pr}[\mathcal{E}_3]$ because $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ are independent.

Claim 2. *$\mathcal{C}$ does not abort in the* Query *phase with $Pr[\mathcal{E}_1] \geq (1 - \frac{1}{q_C})^{q_V}(1-\beta)^{q_S}$.*

$\mathcal{C}$ does not output failure in **Secret-Value-Extract** with probability $(1 - \frac{1}{q_C})^{q_V}$, and does not output failure in **S-Sign** with probability $(1 - (\frac{1}{q_C})\beta)^{q_S} \geq (1-\beta)^{q_S}$. Hence, $\mathsf{Pr}[\mathcal{E}_1] \geq (1 - \frac{1}{q_C})^{q_V}(1-\beta)^{q_S}$.

In addition, $\mathsf{Pr}[\mathcal{E}_2] = \varepsilon$ and $\mathsf{Pr}[\mathcal{E}_3] = \beta/q_C$. The probability of $\mathcal{C}$ is $\mathsf{Pr}[\mathcal{C}(P, aP, bP) \to abP] \geq (1 - \frac{1}{q_C})^{q_V}(1-\beta)^{q_S}(\frac{\beta}{q_C})\varepsilon$. However, $\beta(1-\beta)^{q_S}$ could be maximized at $\beta = \frac{1}{1+q_S}$, so $\mathsf{Pr}[\mathcal{C}(P, aP, bP) \to abP] \geq (1 - \frac{1}{q_C})^{q_V}(1 - \frac{1}{1+q_S})^{q_S}(\frac{1}{q_C(1+q_S)})\varepsilon$. On the other hand, for the performance, τ is denoted by the running time of $\mathcal{A}_{II}$, and τ' of $\mathcal{C}$. $\mathcal{A}_{II}$ can ask at the most q_{H_1} H_1 queries and q_{H_2} H_2 queries where $q_{H_1} = q_{H_2} = q_S + 1$. We conclude $\tau' \leq \tau + 2q_C\tau_{sm} + q_{H_1}\tau_{sm} + q_{H_2}\tau_{sm} + q_S\tau_{sm} = \tau + (2q_C + 3q_S + 2)\tau_{sm}$. The proof of this lemma is complete.