Withdrawable Signature: How to Call off a Signature*

Xin Liu, Joonsang Baek, and Willy Susilo

Institute of Cybersecurity and Cryptology
University of Wollongong, Australia

Abstract. Digital signatures are a cornerstone of security and trust in cryptography, providing authenticity, integrity, and non-repudiation. Despite their benefits, traditional digital signature schemes suffer from inherent immutability, offering no provision for a signer to retract a previously issued signature. This paper introduces the concept of a withdrawable signature scheme, which allows for the retraction of a signature without revealing the signer's private key or compromising the security of other signatures the signer created before. This property, defined as "withdrawability", is particularly relevant in decentralized systems, such as e-voting, blockchain-based smart contracts, and escrow services, where signers may wish to revoke or alter their commitment.

The core idea of our construction of a withdrawable signature scheme is to ensure that the parties with a withdrawable signature are not convinced whether the signer signed a specific message. This ability to generate a signature while preventing validity from being verified is a fundamental requirement of our scheme, epitomizing the property of *withdrawability.* After formally defining security notions for withdrawable signatures, we present two constructions of the scheme based on the pairing and the discrete logarithm. We provide security proof that both constructions are unforgeable under insider corruption and satisfy the criteria of withdrawability. We anticipate our new type of signature will significantly enhance flexibility and security in digital transactions and communications.

Keywords: Digital signatures, Withdrawable signature scheme, Withdrawability.

1 Introduction

Digital signatures are instrumental in constructing trust and security, acting as the essential mechanism for authentication, data integrity, and non-repudiation in contemporary digital communications and transactions. In specific applications of digital signature schemes, such as decentralized e-voting systems, there may arise a natural need for the signer to possess the capability to "undo" a digital signature. Undoing a digital signature implies that the signer may desire to *retract* the signature they created, as seen in e-voting systems where a voter might wish to change or withdraw their vote before the final vote tally.

However, in traditional digital signature schemes, undoing a digital signature is impossible, as it persists indefinitely once a signature is created. Furthermore, digital signatures provide authenticity, integrity, and non-repudiation for signed messages. As a result, when a message is signed, the non-repudiation of its content is guaranteed, meaning that once the signature is generated, the signer cannot rescind it. In light of this limitation, one might ask whether it is possible for a signer to efficiently revoke or withdraw a previously issued digital signature without revealing their private key or compromising the security of other signatures created by the signer. We answer this question by presenting a *withdrawable signature* scheme that provides a practical and secure solution for revocating or withdrawing a signature in a desirable situation.

We note that a traditional signature scheme can achieve "withdrawability" by employing a trusted third party (TTP) to establish signature revocation lists. In cases where a signer desires to invalidate a signature, they notify the TTP, which subsequently adds the revoked signature to the revocation list. This enables future verifiers to consult the revocation list via the TTP, allowing them to determine if the signature has been previously revoked before acknowledging its validity. As all participants fully trust the TTP, including the revoked signature in the revocation list ensures its validity and enables the withdrawal of the signature. However, this approach has a centralized nature as it depends on the TTP's involvement, which may not be desirable in decentralized systems. As in decentralized

* This is an enhanced version of our paper, originally presented at the Information Security Conference (ISC) 2023. In this version, we corrected typos and revised the security proofs for withdrawability.

systems, signers may prefer to manage their signatures without relying on centralized authorities. Therefore, constructing a withdrawable signature scheme that does not rely on a TTP turns out to be a non-trivial problem to solve.

Withdrawable signatures can have various applications in different scenarios where the ability to revoke a signature without compromising the signer's private key is demanded. Here are some potential applications:

Smart Contracts [32]. In the context of blockchain-based smart contracts, withdrawable signatures can enable users to sign off on contract conditions while retaining the ability to revoke their commitment. This can be particularly useful in situations where the fulfillment of the contract depends on the actions of multiple parties or external events.
E-Voting Systems [20]. In a decentralized e-voting system, withdrawable signatures enable voters to securely sign their votes while retaining the option to modify or retract their choices before the final votes count. This additional flexibility improves the voting procedure by allowing voters to respond to fresh insights or unfolding events before the voting period concludes.
Escrow Services [17]. Withdrawable signatures could be employed in decentralized escrow services where multiple parties must sign off on a transaction. If one party decides not to proceed with the transaction due to disputes or changes in conditions, they can revoke their signature without affecting the security of other parties' signatures.

In light of the above discussion, we require the following three properties from the withdrawable signature:

1. A withdrawable signature should be verifiable, especially, it should be verified through the signer's valid public key.
2. Only the signer can generate a valid withdrawable signature.
3. A withdrawable signature, once withdrawn, cannot become valid again without the original signer's involvement.

In the forthcoming subsection, we provide a technical outline of the withdrawable signature scheme, focusing on the technical challenges we had to face.

1.1 Technical Overview

The most important feature of our withdrawable signature scheme is *withdrawability*. The idea behind is that a signer, Alice, should not only be able to sign a message m with her private key to obtain the signature σ but also have the option to revoke the signature if she changes her mind. This means the signature σ will no longer be verifiable with Alice's valid public key.

Our approach to introducing withdrawability into standard digital signatures is achieved through controlling the withdrawability feature through the verifiability of the generated signature. As a verifiable signature is not withdrawable by the signer, our approach to constructing the withdrawable signature centers on generating "non-verifiable" signatures when withdrawability is desired. Meanwhile, a withdrawable signature scheme needs to allow for the confirmation of a non-verifiable (withdrawable) signature into a verifiable one, permitting the signer to lift the withdrawability of the withdrawable signature when desired.

In what follows, we describe the challenges to realizing the withdrawable signature at a technical level.

First attempt: A simple withdrawable signature scheme with TTP. As mentioned earlier, one straightforward solution to achieve withdrawability is to have a trusted third party (TTP) maintain a signature revocation list. However, if we want to attain withdrawability without relying on a revocation list, an alternative approach can be explored as follows: In this approach, the signer, Alice, "hides" a signature ω by encrypting it using her public key and the TTP's public key, resulting in a hidden signature σ, which can be regarded as a withdrawable signature. For example, the BLS signature [8] on a message m, computed as $\omega = H(m)^{\mathsf{sk}_s}$ with the signer's secret key $\mathsf{sk}_s \in \mathbb{Z}_p$ and the hash function $H : \{0,1\}^* \to \mathbb{Z}_p$, can be encrypted into $\sigma = (g^{\mathsf{sk}_\mathsf{t} a} \cdot H(m)^{\mathsf{sk}_s}, g^a)$, where $g^{\mathsf{sk}_\mathsf{t}}$ is the TTP's public key, with sk_t as the corresponding secret key, and $a \in \mathbb{Z}_p^*$ is a uniform random value chosen by the signer.

The withdrawable signature σ preserves the verifiability of the signature as the verification works by checking whether the following equality holds: $e(g^{\mathsf{sk}_\mathsf{t} a} \cdot H(m)^{\mathsf{sk}_s}, g) \stackrel{?}{=} e(g^{\mathsf{sk}_\mathsf{t}}, g^a) e(H(m), g^{\mathsf{sk}_\mathsf{s}})$, where $g^{\mathsf{sk}_\mathsf{s}}$ is the signer's public key.

In the above scheme, everyone can ensure that the signer has generated a valid signature for the message m under her public key $\mathsf{pk}_s(= g^{\mathsf{sk}_s})$, but they cannot extract the original signature $\omega(= H(m)^{\mathsf{sk}_s})$ from σ. (No party except for the TTP can obtain ω.) Meanwhile, σ is not verifiable through only pk_s, which ensures that σ is a withdrawable signature.The signer then has the option to withdraw σ merely by taking no action. Later, the signer can request the TTP to "decrypt" the signature σ into the original signature ω using the TTP's secret key sk_t.

Towards a withdrawable signature scheme without TTP. Implementing a withdrawable signature scheme using a TTP presents a significant drawback, as signers, particularly in decentralized and trustless systems, may wish to achieve withdrawability without reliance on the TTP. How can we achieve withdrawability without the help of the TTP? One possible method involves directly removing the TTP and allowing the signer to create σ using a secret random value $r \in \mathbb{Z}_p^*$ chosen by her, which can be regarded as equivalent to the TTP's secret key $\mathsf{sk_t}$. Subsequently, the signer publishes the corresponding "public key", represented as g^r, and selects another random value $a \in \mathbb{Z}_p^*$.

The withdrawable signature σ is then computed as $\sigma = (g^{ra} \cdot H(m)^{\mathsf{sk}_s}, g^a)$, where the verification of σ can be easily performed using the public keys g^{sk_s} and g^r (with the value g^a) with the following verification algorithm: $e(g^{ra} \cdot H(m)^{\mathsf{sk}_s}, g) \stackrel{?}{=} e(g^r, g^a)e(H(m), g^{\mathsf{sk}_s})$.

However, without the TTP, the signature σ immediately becomes a valid signature that can be verified using the signer's public keys $(g^{\mathsf{sk_t}}, g^r)$; thus, the withdrawability is lost.

Because of this issue, we still need to introduce an additional entity that, while not a TTP, will act as a specific verifier chosen by the signer. More specifically, the signer can produce a signature that cannot be authenticated solely by the signer's public key but also requires the verifier's secret key. This ensures the signature appears unverifiable to everyone except for the chosen verifier, as everyone can only be convinced that the signature was created either by the signer or the verifier. If the verifier cannot transform this signature back into a signature that can be verified using the signer's public key only, this scheme will achieve withdrawability. In particular, only the signer has the option to transform this signature into a verifiable one. To optimize the length of the withdrawable signature, we limit the number of specific verifiers to one.

Another technical issue then surfaces: How can a signer transform the withdrawable signature into a signature that can be directly verifiable using the signer's public key (and possibly with additional public parameters)? A straightforward solution might be having the signer re-sign the message with her secret key. However, this newly generated signature will have no connection to the original withdrawable signature.

Our response to the challenges. To overcome the limitations above, we introduce a designated-verifier signature scheme to generate a withdrawable signature for a message m, denoted as σ, rather than directly generating a regular signature. For a signer Alice, she can create a withdrawable signature for a certain verifier, Bob. Later, if Alice wants to withdraw the signature σ, she just takes no action. If Alice wants to transform the withdrawable signature, she executes an algorithm, "Confirm", to lift the limitation on verifying σ and yield a signature $\widetilde{\sigma}$, which we call "confirmed signature", verifiable using both Alice's and Bob's public keys. Note that the confirmed signature $\widetilde{\sigma}$ can then be deterministically traced back to the original σ.

Generally, there is a withdrawable signature scheme involving two parties, denoted by user_1 and user_2. Without loss of generality, assume that user_1 is the signing user, while user_2 is the certain verifier. Let a set of their public keys be $\gamma = \{\mathsf{pk}_{\mathsf{user}_1}, \mathsf{pk}_{\mathsf{user}_2}\}$. At a high level, we leverage the structure of the underlying regular signature to construct a withdrawable signature σ designated to the verifier user_2. Later with the signer's secret key $\mathsf{sk}_{\mathsf{user}_1}$ and σ, user_1 can generate a verifiable signature for m through the public key set γ. This signature is the confirmed signature $\widetilde{\sigma}$ and can easily be linked with the withdrawable signature σ through the public key set γ.

If we still take the BLS-like signature scheme as an instantiation with $\mathsf{pk}_{\mathsf{user}_1} = g^{\mathsf{sk}_{\mathsf{user}_1}}$ and $\mathsf{pk}_{\mathsf{user}_2} = g^{\mathsf{sk}_{\mathsf{user}_2}}$, considering two hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$. The signer user_1 can generate the withdrawable signature σ of message m for user_2 as follows:

$$y \xleftarrow{\$} \mathbb{Z}_p^*,\ r = H_2(m, g^y,\ H_1(m)^{\mathsf{sk}_{\mathsf{user}_1}}),\ u = H_1(m)^r$$
$$\sigma = (\sigma_1, \sigma_2, \sigma_3) = (e(u^y H_1(m)^{\mathsf{sk}_{\mathsf{user}_1}}, g^{\mathsf{sk}_{\mathsf{user}_2}}), u, g^y).$$

The verification of σ can be performed using the secret key of user_2 and the public key of user_1 as follows: $\sigma_1 \stackrel{?}{=} e(\sigma_2^{\mathsf{sk}_{\mathsf{user}_2}}, \sigma_3)e(H_1(m)^{\mathsf{sk}_{\mathsf{user}_2}}, \mathsf{pk}_{\mathsf{user}_1}) \stackrel{?}{=} e(\sigma_2, \sigma_3^{\mathsf{sk}_{\mathsf{user}_2}})e(H_1(m)^{\mathsf{sk}_{\mathsf{user}_2}}, \mathsf{pk}_{\mathsf{user}_1})$.

Now, assume that user_1 needs to transform σ into a confirmed signature associated with γ. Since user_1 has the secret key $\mathsf{sk}_{\mathsf{user}_1}$, user_1 can easily resign m and reconstruct randomness through $r = H_2(m, \sigma_3, H_1(m)^{\mathsf{sk}_{\mathsf{user}_1}})$ and transform σ into a confirmed signature $\widetilde{\sigma}$ for m of public key set γ with r as follows.

$$\widetilde{\sigma} = (g^{\mathsf{sk}_{\mathsf{user}_2}\mathsf{sk}_{\mathsf{user}_1}r} H_1(m)^{\mathsf{sk}_{\mathsf{user}_1}}, g^r, (\mathsf{pk}_{\mathsf{user}_2})^r).$$

The above signature scheme achieves withdrawability in such a way that even if user_2 reveals it's secret key $\mathsf{sk}_{\mathsf{user}_2}$, other users won't be convinced that σ was generated from user_1. This is due to the potential for user_2 to compute the same σ using $\mathsf{sk}_{\mathsf{user}_2}$, as described below:

$$\begin{aligned}\sigma &= (e(u^y, g^{\mathsf{sk}_{\mathsf{user}_2}})e(H_1(m)^{\mathsf{sk}_{\mathsf{user}_2}}, g^{\mathsf{sk}_{\mathsf{user}_1}}), g^y, u)\\ &= (e(u^y H_1(m)^{\mathsf{sk}_{\mathsf{user}_1}}, g^{\mathsf{sk}_{\mathsf{user}_2}}), g^y, u).\end{aligned}$$

Meanwhile, for a user, i.e., user_3, who only obtains σ and without $\mathsf{sk}_{\mathsf{user}_2}$, the withdrawability is achieved through the DBDH problem, as to decide σ is generated by user_1 or user_2, user_3 needs to decide whether σ contains $e(u^y, g^{\mathsf{sk}_{\mathsf{user}_2}})$ or $e(u^y, g^{\mathsf{sk}_{\mathsf{user}_1}})$ when given u, y and γ. The detailed proof of withdrawability is later given in our paper.

We then demonstrate how to construct a withdrawable signature scheme using the Schnorr [29]-like signature scheme with user_1 and user_2. Assume the public/secret key pair of user_1 and user_2 are still $(\mathsf{pk}_{\mathsf{user}_1} = g^{\mathsf{sk}_{\mathsf{user}_1}}, \mathsf{sk}_{\mathsf{user}_1})$,$\mathsf{pk}_{\mathsf{user}_2} = g^{\mathsf{sk}_{\mathsf{user}_2}}, \mathsf{sk}_{\mathsf{user}_2})$ where $\mathsf{sk}_{\mathsf{user}_1}, \mathsf{sk}_{\mathsf{user}_2} \in \mathbb{Z}_p^*$, respectively. Employing a hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$, the signer user_1 is capable of generating the fundamental Schnorr signature $\omega = (t, z)$ for a given message m in the following manner: $e \xleftarrow{\$} \mathbb{Z}_p^*,\ t = H(m, g^e),\ z = e - \mathsf{sk}_{\mathsf{user}_1} \cdot t$.

With ω, the withdrawable signature σ of message m for user_2 is generated as follows:

$$r = H(m, g^{e\mathsf{sk}_{\mathsf{user}_1}}), \sigma = (\sigma_1, \sigma_2, \sigma_3) = \left(g^e, \mathsf{pk}_{\mathsf{user}_2}^{z-r\cdot t}, g^r\right).$$

The verification algorithm of σ can be performed using the secret key of user_2 and the public key of user_1 as follows: user_2 first reconstructs $t' = H(m, \sigma_1)$, and verifies if $\sigma_2 = (\sigma_1(\mathsf{pk}_{\mathsf{user}_1}\sigma_3)^{-t'})^{\mathsf{sk}_{\mathsf{user}_2}}$ holds or not.

Now, assume that user_1 needs to transform σ into a confirmed signature that is associated with $\gamma = \left\{\mathsf{pk}_{\mathsf{user}_1}, \mathsf{pk}_{\mathsf{user}_2}\right\}$. Given that only user_1 has the secret key $\mathsf{sk}_{\mathsf{user}_1}$, user_1 can still easily reconstruct randomness $r = H(m, \sigma_1^{\mathsf{sk}_{\mathsf{user}_1}})$. However, a key distinction with the withdrawable signature based on BLS is that with the withdrawable signature σ, user_1 now cannot reconstruct ω from σ directly because solving for "e" in the expression g^e (related to the Discrete Logarithm (DL) problem) is computationally difficult. Therefore, an alternative method is required for user_1 to transform σ into a confirmed signature $\widetilde{\sigma}$. Our solution is summarised as follows:

- user_1 begins by re-signing message m and generating a new Schnorr signature $\omega_s = (t_s, z_s)$ with its secret key.
- Using t_s, user_1 can compute $\widetilde{z}_s = z_s - rt_s$. The verification of $\widetilde{z}_s$ is finalized through the public key $\mathsf{pk}_{\mathsf{user}_1}$ and σ_3 from the withdrawable signature.
- To ensure that $\widetilde{\sigma}$ is associated with $\mathsf{pk}_{\mathsf{user}_2}$, user_1 can generate another Schnorr signature ω_t for user_2's public key $\mathsf{pk}_{\mathsf{user}_2}$ with r. The signature ω_2 is verifiable through g^r, which could be obtained through σ_3. This process ensures that the confirmed signature $\widetilde{\sigma}$ is connected with σ.

The resulting confirmed signature $\widetilde{\sigma}$ is shown as follows:

$$e_j \xleftarrow{\$} \mathbb{Z}_p^*, t_j = H(\mathsf{pk}_{\mathsf{user}_2}, g^{e_j}), \widetilde{\sigma} = (t_s, \widetilde{z}_s, e_j, t_j).$$

We show that this construction of withdrawable signature using the Schnorr [29]-like signature also achieves the attribute of withdrawability.

1.2 Our Contributions

Motivated by the absence of the certain type of signature scheme we want for various aforementioned applications, we present the concept called withdrawable signatures. Our contributions in this regard can be summarized as follows:

1. We provide a formal definition of a *withdrawable signature* scheme that reflects all the characteristics we discussed previously.
2. We formulate security notions of withdrawable signature, reflecting the *withdrawability* and *unforgeability*, two essential security properties.
3. We propose two constructions of withdrawable signature schemes based on pairing and DL.

This paper is organized as follows: We first review the related work in section 2. In Section 3, we provide a comprehensive definition of withdrawable signatures, including their syntax and security notion. Section 4 begins with a detailed overview of the preliminaries we used to build our withdrawable signature schemes, then we give the full description of our two proposed constructions. Following that, Section 5 focuses on the security analysis of the above two withdrawable signature constructions.

2 Related Work

In this section, we first review the previous work relevant to our withdrawable signature scheme and highlight differences between our scheme and existing ones.

Designated-Verifier Signature Scheme. The concept of designated-verifier signature (or proof) (DVS) was first introduced by Jakobsson et al. [19], and independently by Chaum [9]. Since then, the field has been studied for several decades with various assumptions leading to different instantiations. [22, 34–37]. The concept of designated-verifier signature (or proof) (DVS) was first introduced by Jakobsson et al. [19], and independently by Chaum [9]. Since then, the field has been studied for several decades with various assumptions leading to different instantiations. Notable contributions in this area include works [22, 34–37].

Revocable Group Signature Scheme. Group signature [2, 6, 9] allows any member within a group to authenticate a message on behalf of the collective. In the context of revocable group signature schemes [3, 23, 27], revocation refers to the capability of the group manager to revoke a member's signing privilege.

Ring Signature Scheme. The concept of ring signature was first proposed by Rivest, Shamir, and Tauman in [28]. In a ring signature scheme, a signer can select a set of public keys, including their own, and create a signature on behalf of that set. [1, 4, 5, 7, 10–12, 15, 16, 39]. For instance, in the public key setting, include RSA-based [28], discrete logarithm-based [16], pairing-based [7], lattice-based [11, 12] approaches. Ring signatures can be generically constructed via zero-knowledge proof on a signer index, particularly through a one-out-of-many proof, as demonstrated in [15, 26]. The logarithmic-size constructions are also suggested in [4, 13, 14].

Revocable Ring Signature Scheme. The notion of revocable ring signatures [24] was first introduced in 2007. This concept added new functionality where a specified group of revocation authorities could remove the signer's anonymity. In [41], Zhang *et al.* presented a revocable and linkable ring signature (RLRS) scheme. This innovative framework empowers a revocation authority to reveal the real signer's identity in a linkable ring signature scheme [25].

Universal Designated Verifier Signature Scheme. Designated-verifier signature schemes have multiple variations, including Universal Designated Verifier Signature (UDVS) schemes. Steinfeld et al. proposed the first UDVS scheme based on the bilinear group [30]. They developed two other UDVS schemes, which expanded the conventional Schnorr/RSA signature schemes [31]. Following the work by Steinfeld et al., several UDVS schemes have been proposed in literature [18,33,38,40]. Additionally, the first lattice-based UDVS was proposed in [21].

Discussion on differences. Our withdrawable signature constructions presented above comprise two primary parts: withdrawable signature generation and transformation of a withdrawable signature into a confirmed one. When viewed through the "withdrawability" requirements of the first part, our withdrawable signature scheme is relevant to existing group and ring signatures, wherein the signer retains anonymity within a two-party setup. What distinguishes our approach is the second transformation stage, which offers a unique feature not found in the aforementioned revocable group and ring signatures. Our scheme empowers signers to retract their signatures independently, without relying on a certain group manager or a set of revocation authorities. Additionally, the right to remove its "anonymity" rests only with the signer.

Readers might also discern similarities between our withdrawable signature scheme and designated-verifier signature (DVS) schemes. In the withdrawable signature generation phase of our scheme, the generated signature can only convince a specific verifier (the designated verifier) that the signer has generated a signature, the same as the core concept of DVS. Note that a DVS holds "non-transferability", which means that a DVS cannot be transferred by either the signer or the verifier to convince a third party. Although this non-transferability aligns with our concept of withdrawability, our scheme diverges by permitting the signer to transform the withdrawable signature into one that's verifiable using both the signer's and verifier's public keys, challenging the foundational property of DVS.

To achieve this additional property at the transformation stage, we consider leveraging the structural properties of existing regular signatures. Provided that our withdrawable signature scheme was derived from a particular signature, which has been generated with the signer's secret key, only the signer can access this underlying regular signature during the transformation stage. Then one might have also noticed that the construction of our withdrawable signature scheme is related to the UDVS scheme. In a UDVS scheme, once the signer produces a signature on a message, any party possessing this message-signature pair can designate a third party as the designated verifier by producing a DVS with this message-signature pair for this verifier. Much like DVS, UDVS is bound by non-transferability as well. Meanwhile, our withdrawable signature scheme takes another different approach than UDVS's as our scheme does not require the signer to reveal the underlying regular signature at the withdrawable signature generation stage.

In our withdrawable signature scheme construction, the underlying regular signature is treated as a secret held by the signer. This secret ensures the signer creates a corresponding withdrawable signature specific to a certain (designated) verifier. Later at the transformation stage, we require the additional input as the public key set of signer and verifier and the signer's secret key to reconstruct the underlying additional regular signature. With these inputs, we can finalize our transformation algorithm.

3 Definitions

In this section, we provide a comprehensive overview of the syntax and security notion of withdrawable signature.

3.1 Notation and Terminology

Throughout this paper, we use λ as the security parameter. By $a \xleftarrow{\$} \mathcal{S}$, we denote an element a is chosen uniformly at random from a set $\mathcal{S}$. Let $\mathcal{S} = \left\{\mathsf{pk}_1, \cdots, \mathsf{pk}_\mu\right\}$ be a set of public keys, where each public key pk_i is generated by the same key generation algorithm $\mathsf{KeyGen}(1^k)$ and $\mu = |\mathcal{S}|$. The corresponding secret key of pk_i is denoted by sk_i. Given two distinct public keys $\mathsf{pk}_s, \mathsf{pk}_j \xleftarrow{\$} \mathcal{S}$ where $j \neq s$, the signer's public key is denoted by pk_s.

3.2 Withdrawable Signature: A Formal Definition

Naturally, our withdrawable signature scheme involves two parties: signers and verifiers. At a high level, the scheme consists of two stages, i.e., generating a withdrawable signature and transforming it into a confirmed signature. These two stages are all completed by the signer.

More precisely, a withdrawable signature scheme $\mathcal{WS}$ consists of five polynomial time algorithms, $(\mathsf{KeyGen}, \mathsf{WSign}, \mathsf{WSVerify}, \mathsf{Confirm}, \mathsf{CVerify})$, each of which is described below:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^k)$: The key generation algorithm takes the security parameters 1^k as input, to return a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\sigma \leftarrow \mathsf{WSign}(m, \mathsf{sk}_s, \gamma)$: The "withdrawable signing" algorithm takes as input a message m, signer's secret key sk_s and $\gamma = \left\{\mathsf{pk}_s, \mathsf{pk}_j\right\}$ where $\mathsf{pk}_s, \mathsf{pk}_j \in \mathcal{S}$, to return a new withdrawable signature σ of m respect to pk_s, which is designated to verifier pk_j.
- $1/0 \leftarrow \mathsf{WSVerify}(m, \mathsf{sk}_j, \mathsf{pk}_s, \sigma)$: The "withdrawable signature verification" algorithm takes as input a withdrawable signature σ of m with respect to pk_s, the designated verifier's secret key sk_j, to return either 1 or 0.

– $\widetilde{\sigma} \leftarrow \mathsf{Confirm}(m, \mathsf{sk}_s, \gamma, \sigma)$: The "confirm" algorithm takes as input a withdrawable signature σ of m with respect to pk_s, signer's secret key sk_s, the public key set γ, to return a confirmed signature $\widetilde{\sigma}$ of m, $\widetilde{\sigma}$ is a verifiable signature with respect to γ.
– $1/0 \leftarrow \mathsf{CVerify}(m, \gamma, \sigma, \widetilde{\sigma})$: The "confirmed signature verification" algorithm takes as input a confirmed signature signature $\widetilde{\sigma}$ of m with respect to γ, and the corresponding withdrawable signature σ, to return either 1 or 0.

3.3 Security Notions of Withdrawable Signature

The security notion of a withdrawable signature scheme $\mathcal{WS}$ covers the properties of correctness, unforgeability under insider corruption, and withdrawability three aspects.

Correctness. As long as the withdrawable signature σ is verifiable through the withdrawable signature verification algorithm $\mathsf{WSVerify}$, it can be concluded that the corresponding confirmed signature $\widetilde{\sigma}$ will also be verifiable through the confirm verification algorithm $\mathsf{CVerify}$.

Unforgeability under insider corruption. Nobody except the signer can transform a verifiable withdrawable signature σ generated from sk_s for pk_j into corresponding confirmed signature $\widetilde{\sigma}$, even the adversary can always obtain the secret key sk_j of the verifier.

Withdrawability. The withdrawability means that, given a verifiable withdrawable signature σ, it must be intractable for any PPT adversary $\mathcal{A}$ to distinguish whether σ was generated by the signer or the verifier unless the $\mathsf{Confirm}$ algorithm of σ has been achieved. The withdrawability ensures that the capability of generating a withdrawable signature is equivalent between the signer and the certain verifier.

Below, we provide formal security definitions. The formal definitions of correctness, unforgeability under insider corruption, and withdrawability.

We call a withdrawable signature scheme $\mathcal{WS}$ *secure* if it is *correct, unforgeable under insider corruption, withdrawable.*

Definition 1 (Correctness). *A withdrawable signature scheme $\mathcal{WS}$ is considered correct for any security parameter k, any public key set γ, and any message $m \in \{0,1\}^*$, if with following algorithms:*

– $(\mathsf{pk}_s, \mathsf{sk}_s), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(1^k)$
– $\gamma \leftarrow \{\mathsf{pk}_s, \mathsf{pk}_j\}$
– $\sigma \leftarrow \mathsf{WSign}(m, \mathsf{sk}_s, \gamma)$
– $\widetilde{\sigma} \leftarrow \mathsf{Confirm}(m, \mathsf{sk}_s, \gamma, \sigma)$

it holds with an overwhelming probability (in k) that the corresponding verification algorithms:

$$\mathsf{WSVerify}(m, \mathsf{sk}_j, \mathsf{pk}_s, \sigma) = 1 \text{ and } \mathsf{CVerify}(m, \gamma, \sigma, \widetilde{\sigma}) = 1.$$

Unforgeability under insider corruption. Unforgeability under insider corruption means that the adversary $\mathcal{A}$ cannot generate a valid confirmed signature from a withdrawable signature for a certain signer without its secret key, even if $\mathcal{A}$ can adaptively corrupt some honest participants as certain verifiers and obtain their secret keys.

Definition 2 (Unforgeability under insider corruption). *Considering an unforgeability under insider corruption experiment $\mathsf{Exp}^{\text{EUF-CMA}}_{\mathcal{WS},\mathcal{A}}(1^k)$ for a PPT adversary $\mathcal{A}$ and security parameter k. The three oracles we use to build the $\mathsf{Exp}^{\text{EUF-CMA}}_{\mathcal{WS},\mathcal{A}}(1^k)$ are shown as follows.*

Oracle $\mathcal{O}^{\mathsf{Corrupt}}_i(\cdot)$	Oracle $\mathcal{O}^{\mathsf{WSign}}_{\mathsf{sk}_s,\gamma}(\cdot)$	Oracle $\mathcal{O}^{\mathsf{Confirm}}_{\mathsf{sk}_s,\sigma,\gamma}(\cdot)$
if $i \neq s$,	**if** $\mathsf{pk}_s \in \gamma \wedge s \notin \mathcal{CO}$,	**if** $\sigma \in \mathcal{W}$
$\mathcal{CO} \leftarrow \mathcal{CO} \cup \mathsf{sk}_i$	$\sigma \leftarrow \mathsf{WSign}(m, \mathsf{sk}_s, \gamma)$	$\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
return sk_i	$\mathcal{W} \leftarrow \mathcal{W} \cup \{\sigma\}$	$\widetilde{\sigma} \leftarrow \mathsf{Confirm}(m, \mathsf{sk}_s, \gamma, \sigma)$
else return $\perp$	**return** σ	**return** $\widetilde{\sigma}$
	else return $\perp$	**else return** $\perp$

With these three oracles, we have the following experiment $\mathsf{Exp}^{\text{EUF-CMA}}_{\mathcal{WS},\mathcal{A}}(1^k)$:

$\mathsf{Exp}^{\text{EUF-CMA}}_{\mathcal{WS},\mathcal{A}}(1^k)$

for $i = 1$ **to** μ **do**

$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^k), s, j \in [1, \mu], j \neq s;$

$\mathcal{CO}, \mathcal{W}, \mathcal{M} \leftarrow \emptyset;$

$(m^*, \widetilde{\sigma}^*) \leftarrow \mathcal{A}^{\mathcal{O}_i^{\mathsf{Corrupt}}(\cdot), \mathcal{O}^{\mathsf{WSign}}_{\mathsf{sk}_s,\gamma}(\cdot), \mathcal{O}^{\mathsf{Confirm}}_{\mathsf{sk}_s,\sigma,\gamma}(\cdot)}(1^k, \gamma^*, \sigma^*)$

if $\gamma^* = \{\mathsf{pk}_s, \mathsf{pk}_j\}, j \in \mathcal{CO} \wedge m^* \notin \mathcal{M}$

$\wedge \mathsf{WSVerify}(m^*, \mathsf{sk}_j, \mathsf{pk}_s, \sigma^*) = 1 \wedge \mathsf{CVerify}(m^*, \gamma^*, \sigma^*, \widetilde{\sigma}^*) = 1$

return 1

else return 0

A withdrawable signature scheme $\mathcal{WS}$ is unforgeable under insider corruption of EUF-CMA security if for all PPT adversary $\mathcal{A}$, there exists a negligible function negl *such that:*

$$\Pr[\mathsf{Exp}^{\text{EUF-CMA}}_{\mathcal{WS},\mathcal{A}}(1^k) = 1] \leq \mathsf{negl}(1^k).$$

Withdrawability. The withdrawability means that a PPT adversary $\mathcal{A}$ is always given a message m and an unconfirmed withdrawable signature σ, it is infeasible to determine who created the signature.

Definition 3 (Withdrawability). *Assume two public/secret key pairs are generated as* $(\mathsf{pk}_0, \mathsf{sk}_0)$, $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(1^k)$. *Let* $\gamma = \{\mathsf{pk}_0, \mathsf{pk}_1\}$ *and* $b \xleftarrow{\$} \{0, 1\}$, *considering a withdrawability experiment* $\mathsf{Exp}^{\mathsf{Withdraw}}_{\mathcal{WS},\mathcal{A}}(1^k)$ *for a PPT adversary $\mathcal{A}$ and security parameter k.*

The oracle we use to build our withdrawability experiment $\mathsf{Exp}^{\mathsf{Withdraw}}_{\mathcal{WS}}(1^k)$ *is shown as follows.*

Oracle $\mathcal{O}^{\mathsf{WSign}}_{\mathsf{sk}_s,\gamma}(\cdot)$

if $\gamma = \{\mathsf{pk}_0, \mathsf{pk}_1\}, b \xleftarrow{\$} \{0, 1\}$

$\sigma_b \leftarrow \mathsf{WSign}(m, \mathsf{sk}_b, \gamma)$

$\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$

return σ_b

else return $\perp$

With this signing oracle, we have the following experiment $\mathsf{Exp}^{\mathsf{Withdraw}}_{\mathcal{WS}}(1^k)$*:*

$\mathsf{Exp}^{\mathsf{Withdraw}}_{\mathcal{WS},\mathcal{A}}(1^k)$

for $i = 0$ **to** 1 **do**

$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^k), \gamma = \{\mathsf{pk}_0, \mathsf{pk}_1\}$

$b \xleftarrow{\$} \{0, 1\}$, $\mathcal{M} \leftarrow \emptyset$;

if $\gamma = \{\mathsf{pk}_0, \mathsf{pk}_1\} \wedge m^* \notin \mathcal{M}$

$\sigma_b \leftarrow \mathsf{WSign}(m^*, \mathsf{sk}_b, \gamma)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{WSign}}_{\mathsf{sk}_b,\gamma}(\cdot)}(1^k, m^*, \sigma_b^*)$

if $b = b'$

return 1

else return 0

A withdrawable signature $\mathcal{WS}$ achieves withdrawability if, for any PPT adversary $\mathcal{A}$, as long as the $\mathsf{Confirm}$ *algorithm hasn't been executed, there exists a negligible function* negl *such that:*

$$\Pr[\mathsf{Exp}^{\mathsf{Withdraw}}_{\mathcal{WS},\mathcal{A}}(1^k) = 1] \leq \frac{1}{2} + \mathsf{negl}(1^k).$$

4 Our Withdrawable Signature Schemes

In this section, we present two specific constructions of withdrawable signatures. We start by introducing the necessary preliminaries that form the basis of our constructions.

4.1 Preliminaries

Bilinear Groups. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three (multiplicative) cyclic groups of prime order p. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. A bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

- **Bilinearity:** For all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
- **Non-degeneracy:** $e(g_1, g_2) \neq 1$ (i.e. $e(g_1, g_2)$ generates $\mathbb{G}_T$).
- **Computability:** For all $u \in \mathbb{G}_1,\ v \in \mathbb{G}_2$, there exists an efficient algorithm to compute $e(u, v)$.

If $\mathbb{G}_1 = \mathbb{G}_2$, then e is *symmetric* (Type-1) and *asymmetric* (Type-2 or 3) otherwise. For Type-2 pairings, there is an efficiently computable homomorphism ϕ: $\mathbb{G}_2 \to \mathbb{G}_1$. For Type-3 pairings no such homomorphism is known.
Digital Signatures. A signature scheme DS consists of three PPT algorithms, described as follows:

$$\mathsf{DS} = \begin{cases} (\mathsf{pk}_s, \mathsf{sk}_s) & \leftarrow \mathsf{KeyGen}(1^k) \\ \sigma & \leftarrow \mathsf{Sign}(m, \mathsf{sk}_s) \\ 0/1 & \leftarrow \mathsf{Verify}(m, \mathsf{pk}_s, \sigma) \end{cases}$$

The relevant security model of existential unforgeability against chosen-message attacks (EUF-CMA) for digital signature schemes is given as follows.

Definition 4 (EUF-CMA)**.** *Given a signature scheme* $\mathsf{DS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*, and a ppt adversary* $\mathcal{A}$*, considering the following game* $\mathsf{Exp}_{\mathcal{A}}^{\text{EUF-CMA}}$:

- *Let SP be the system parameters. The challenger $\mathcal{B}$ runs the key generation algorithm to generate a key pair* $(\mathsf{pk}_s, \mathsf{sk}_s)$ *and sends* pk *to the adversary* $\mathcal{A}$*. The challenger keeps* sk_s *to respond to signature queries from the adversary.*
- $\mathcal{A}$ *is given access to an oracle* $\mathcal{O}^{\mathsf{Sign}}_{\mathsf{sk}_s}(\cdot)$ *such that* $\mathcal{O}^{\mathsf{Sign}}_{\mathsf{sk}_s}(\cdot) : \sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}_s)$.
- $\mathcal{A}$ *outputs a message* m^**, and returns a forged signature* σ^* *on* m^*.
- $\mathcal{A}$ *succeeds if* σ^* *is a valid signature of the message* m^* *and the signature of* m^* *has not been queried in the query phase.*

A signature scheme is (t, q_s, ε)*-secure in the* EUF-CMA *security model if there exists no adversary who can win the above game in time* t *with advantage* ε *after it has made* q_s *signature queries.*

Designated-Verifier Signatures [19]. A designated-verifier signature DVS consists of four PPT algorithms, where the signer's public/secret key pair is denoted as $(\mathsf{pk}_s, \mathsf{sk}_s)$, and the designated-verifier's public /secret key pair is denoted as $(\mathsf{pk}_d, \mathsf{sk}_d)$. The definition of designated-verifier signature schemes is described as follows:

$$\mathsf{DVS} = \begin{cases} (\mathsf{pk}, \mathsf{sk}) & \leftarrow \mathsf{KeyGen}(1^k) \\ \sigma & \leftarrow \mathsf{Sign}(m, \mathsf{pk}_d, \mathsf{sk}_s) \\ \sigma & \leftarrow \mathsf{Simul}(m, \mathsf{pk}_s, \mathsf{sk}_d) \\ 0/1 & \leftarrow \mathsf{Verify}(m, \mathsf{pk}_s, \mathsf{sk}_d, \sigma) \end{cases}$$

The relevant non-transferability model of designated-verifier signature is called "non-transferability". Non-transferability implies, that, given a message-DVS signature pair (m, σ), which is accepted by the designated verifier, without access to the secret key of the signer, it is computationally infeasible to determine whether the message was signed by the signer or the signature was simulated by the designated-verifier. The formal definition is shown as follows.

Definition 5 (Non-transferability). *Given a designated-verifier signature scheme polynomial in* λ*, and a ppt adversary* $\mathcal{A}$*, consider the following game* $\mathsf{Exp}^{\mathsf{Sign}}_{\mathsf{NonTrans},\mathsf{DV},\mathcal{A}}$*:*

- *Generates* $(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow \mathsf{KeyGen}(1^k)$ *and* $(\mathsf{pk}_d, \mathsf{sk}_d) \leftarrow \mathsf{KeyGen}(1^k)$.
- $\mathcal{A}$ *is given access to a signing oracle* $\mathcal{O}^{\mathsf{Sign}}_{\mathsf{sk}_s,\mathsf{pk}_d}(\cdot)$ *such that:*$\mathcal{O}^{\mathsf{Sign}}_{\mathsf{sk}_s,\mathsf{pk}_d}(\cdot) : \sigma_0 \leftarrow \mathsf{Sign}(m, \mathsf{sk}_s, \mathsf{pk}_d)$*; and a simulation oracle* $\mathcal{O}^{\mathsf{Simul}}_{\mathsf{sk}_d,\mathsf{pk}_s}(\cdot)$ *such that:* $\mathcal{O}^{\mathsf{Sign}}_{\mathsf{sk}_d,\mathsf{pk}_s}(\cdot) : \sigma_1 \leftarrow \mathsf{Simul}(m, \mathsf{sk}_s, \mathsf{pk}_d)$.
- $\mathcal{A}$ *outputs a message* m^**, furthermore, a random bit* b *is chosen, and* $\mathcal{A}$ *is given the signature* σ^*_b.

– *The adversary outputs a bit b', and succeeds if $b' = b$.*

A DVS *achieves nom-transferability if* $\Pr[\mathsf{Exp}^{\mathsf{Sign}}_{\mathsf{NonTrans,DV},\mathcal{A}}(1^k) = 1] \leq \frac{1}{2} + \mathsf{negl}(1^k)$.

Universal Designated-Verifier Signatures [30, 31]. The universal designated-verifier signature scheme can operate as a standard publicly-verifiable digital signature but possesses additional functionality. This extended functionality allows any holder of a regular signature (not necessarily only the signer) to designate the signature to any desired verifier using the verifier's public key.

4.2 Computational Assumptions

We begin by revisiting the Computational Diffie-Hellman (CDH) assumption and the Discrete Logarithm (DL) assumption, under which the BLS signatures and Schnorr signatures are respectively proven EUF-CMA secure. Furthermore, we will also revisit the Decisional Bilinear Diffie-Hellman (DBDH) assumption and the Decisional Diffie-Hellman (DDH) assumption. These two assumptions are used to prove the withdrawability feature of our proposed withdrawable signature constructions based on BLS and Schnorr signatures, respectively.

Definition 6 (CDH Assumption). *Let $\mathbb{G}$ be a generic group of prime order p, and g is a generator of $\mathbb{G}$. Given (g, g^a, g^b) for $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, no adversary $\mathcal{A}$ can output $g^x \in \mathbb{Z}_p^*$ where $g^x = g^{ab}$.*

Definition 7 (DL Assumption). *Let $\mathbb{G}$ be a generic group of prime order p, and g is a generator of $\mathbb{G}$. Given (g, g^a) for $a \xleftarrow{\$} \mathbb{Z}_p^*$, no adversary $\mathcal{A}$ can output $a' \in \mathbb{Z}_p^*$ where $g^{a'} = g^a$.*

Definition 8 (DBDH Assumption). *Let $\mathbb{G}$ be a generic group of prime order p, and g is a generator of $\mathbb{G}$, $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Given $(g, g^a, g^b, g^c) \in \mathbb{G}$ and $Z \in \mathbb{G}_T$, no adversary $\mathcal{A}$ can decide whether $Z = e(g, g)^{abc}$ or not.*

Definition 9 (DDH Assumption). *Let $\mathbb{G}$ be a generic group of prime order p, and g is a generator of $\mathbb{G}$. Given $(g, g^a, g^b, Z) \in \mathbb{G}$, no adversary $\mathcal{A}$ can decide whether $Z = g^{ab}$ or not.*

4.3 A Construction Based on BLS

Suppose $\mathbb{G}$ is a generic group of prime order p, and g is a generator, with two hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$. $\mathbb{PG} : \mathbb{G} \times \mathbb{G} = \mathbb{G}_T$ is a Type-1 bilinear pairing as defined in Section 4.1.

Let BLS.DS denotes the BLS signature scheme [8], which contains three algorithms: BLS.DS = (KeyGen, BLS.Sign, BLS.Verify). Comprehensive details of these three algorithms are outlined as follows. The output of the signing algorithm is denoted as $\omega \leftarrow \mathsf{BLS.Sign}(m, \mathsf{sk}_s)$ where ω is derived, such that $\omega = H_1(m)^{\mathsf{sk}_s}$.

$\underline{\mathsf{Setup}(\cdot)}$ define $H_1 : \{0,1\}^* \to \mathbb{G}$ **return** H_1	$\underline{\mathsf{KeyGen}(1^k)}$ $\mathsf{sk}_s \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{pk}_s = g^{\mathsf{sk}_s}$ **return** $(\mathsf{pk}, \mathsf{sk})$
$\underline{\mathsf{BLS.Sign}(m, \mathsf{sk}_s)}$ $h = H_1(m)$ $\omega \leftarrow h^{\mathsf{sk}_s}$ **return** ω	$\underline{\mathsf{BLS.Verify}(m, \mathsf{pk}_s, \omega)}$ parse ω, $t' = H(m)$ **if** $e(\sigma, g) \neq e(h', \mathsf{pk}_s)$ **return** 0 **else return** 1

Fig.1. The Detail of BLS Signature Scheme

Following this, we have a construction of a withdrawable signature scheme based on the original BLS signature scheme.

$\underline{\mathsf{Setup}(\cdot)}$

define $H_1 : \{0,1\}^* \to \mathbb{G}$

define $H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$

return H_1, H_2

$\underline{\mathsf{KeyGen}(1^k)}$

$\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{pk}_i = g^{\mathsf{sk}_i}$

return $(\mathsf{pk}_i, \mathsf{sk}_i)$

$\underline{\mathsf{WSign}(m, \mathsf{sk}_s, \gamma)}$

$(\mathsf{pk}_s, \mathsf{sk}_s), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(1^k)$

parse $\gamma = \{\mathsf{pk}_s, \mathsf{pk}_j\}$

$\boxed{\omega = H_1(m)^{\mathsf{sk}_s}}$

$y \xleftarrow{\$} \mathbb{Z}_p^*$, $r = H_2(m, g^y, \omega)$

$\sigma_1 \leftarrow e(H_1(m)^{ry}\omega, \mathsf{pk}_j)$

$\sigma_2 \leftarrow H_1(m)^r$, $\sigma_3 \leftarrow g^y$

$\sigma = (\sigma_1, \sigma_2, \sigma_3)$

return σ

$\underline{\mathsf{WSVerify}(m, \mathsf{sk}_j, \mathsf{pk}_s, \sigma)}$

parse $\sigma = (\sigma_1, \sigma_2, \sigma_3)$

if $\sigma_1 = e(\sigma_2, \sigma_3^{\mathsf{sk}_j})e(H_1(m)^{\mathsf{sk}_j}, \mathsf{pk}_s)$

$= e(\sigma_2^{\mathsf{sk}_j}, \sigma_3)e(H_1(m)^{\mathsf{sk}_j}, \mathsf{pk}_s)$

return 1

else return 0

$\underline{\mathsf{Confirm}(m, \mathsf{sk}_s, \gamma, \sigma)}$

parse $\sigma = (\sigma_1, \sigma_2, \sigma_3)$

$\boxed{\omega = H_1(m)^{\mathsf{sk}_s}}$

$r' = H_2(m, \sigma_3, \omega)$

$\delta_1 \leftarrow \mathsf{pk}_j^{\mathsf{sk}_s r'}\omega$

$\delta_2 \leftarrow \mathsf{pk}_j^{r'}$, $\delta_3 \leftarrow g^{r'}$

$\widetilde{\sigma} = (\delta_1, \delta_2, \delta_3)$

return $\widetilde{\sigma}$

$\underline{\mathsf{CVerify}(m, \gamma, \sigma, \widetilde{\sigma})}$

parse $\widetilde{\sigma} = (\delta_1, \delta_2, \delta_3)$

parse $\sigma = (\sigma_1, \sigma_2, \sigma_3)$

if $e(\sigma_2, g) = e(H_1(m), \delta_3)$,

$e(\delta_1, g) = e(\mathsf{pk}_s, \delta_2)e(H_1(m), \mathsf{pk}_s)$,

$e(\delta_2, g) = e(\delta_3, \mathsf{pk}_j)$

return 1

else return 0

Fig.2. A Construction Based on BLS

4.4 A Construction Based on Schnorr

Recall that $\mathbb{G}$ is a generic group of prime order p, and g is a generator, with hash function $H : \{0,1\}^* \to \mathbb{Z}_p$.

Let $\mathsf{Sch.DS}$ denote the Schnorr signature scheme [29], which contains three algorithms: $\mathsf{Sch.DS} = (\mathsf{KeyGen}, \mathsf{Sch.Sign}, \mathsf{Sch.Verify})$. Details of these three algorithms are outlined in [29]. The output of the signing algorithm is also denoted as $\omega \leftarrow \mathsf{Sch.Sign}(m, \mathsf{sk}_s)$ where $\omega = (t, z)$ is derived as follows:

A randomness e is randomly selected from $\mathbb{Z}_p$, then u is calculated as $u = g^e$. The value t is computed using the hash function $t = H(m, u)$. Finally, z is calculated as $z = (e - \mathsf{sk}_s t) \mod p$.

$\underline{\mathsf{Setup}(\cdot)}$

define $H : \{0,1\}^* \to \mathbb{Z}_p^*$

return H

$\underline{\mathsf{KeyGen}(1^k)}$

$\mathsf{sk}_s \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{pk}_s = g^{\mathsf{sk}_s}$

return $(\mathsf{pk}_s, \mathsf{sk}_s)$

$\underline{\mathsf{Sch.Sign}(m, \mathsf{sk}_s)}$

$e \xleftarrow{\$} \mathbb{Z}_p^*$, $t = H(m, g^e)$

$z = e - \mathsf{sk}_s t$

$\sigma \leftarrow (t, z)$

return σ

$\underline{\mathsf{Sch.Verify}(m, \mathsf{pk}_s, \omega)}$

parse $\omega = (t, z)$

if $H(m, g^z \mathsf{pk}_s^t) \neq t$

return 0

else return 1

Fig.3. The Detail of Schnorr Signature Scheme

Following this, we have a construction of a withdrawable signature based on the Schnorr signature:

$\underline{\mathsf{Setup}(\cdot)}$ define $H : \{0,1\}^* \to \mathbb{Z}_p$ **return** H	$\underline{\mathsf{KeyGen}(1^k)}$ $\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{pk}_i = g^{\mathsf{sk}_i}$ **return** $(\mathsf{pk}_i, \mathsf{sk}_i)$
$\underline{\mathsf{WSign}(m, \mathsf{sk}_s, \gamma)}$ $(\mathsf{pk}_s, \mathsf{sk}_s), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(1^k)$ parse $\gamma = \{\mathsf{pk}_s, \mathsf{pk}_j\}$ $e \xleftarrow{\$} \mathbb{Z}_p^*$, $t = H(m, g^e)$ $z = e - \mathsf{sk}_s t$ $\boxed{\omega = (t, z)}$ $r = H(m, g^{\mathsf{sk}_s e})$ $\sigma_1 \leftarrow g^e$, $\sigma_2 \leftarrow \mathsf{pk}_j^{z-rt}$, $\sigma_3 \leftarrow g^r$ $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ **return** σ	$\underline{\mathsf{WSVerify}(m, \mathsf{sk}_j, \mathsf{pk}_s, \sigma)}$ parse $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ $t' = H(m, \sigma_1)$ **if** $\sigma_2 = (\sigma_1(\mathsf{pk}_s\sigma_3)^{-t'})^{\mathsf{sk}_j}$ **return** 1 **else return** 0
$\underline{\mathsf{Confirm}(m, \mathsf{sk}_s, \gamma, \sigma)}$ parse $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ $e_s \xleftarrow{\$} \mathbb{Z}_p^*$, $r' = H(m, \sigma_1^{\mathsf{sk}_s})$ $t_s = H(m, g^{e_s})$ $z_s = e_s - \mathsf{sk}_s t_s$ $\boxed{\omega_s = (t_s, z_s)}$ $e_j \xleftarrow{\$} \mathbb{Z}_p^*, t_j = H(\mathsf{pk}_j, e_j)$ $z_j = e_j - r' t_j$ $\delta_1 \leftarrow t_s$, $\delta_2 \leftarrow z_s - r' t_s$ $\delta_3 \leftarrow t_j$, $\delta_4 \leftarrow z_j$ $\widetilde{\sigma} = (\delta_1, \delta_2, \delta_3, \delta_4)$ **return** $\widetilde{\sigma}$	$\underline{\mathsf{CVerify}(m, \gamma, \sigma, \widetilde{\sigma})}$ parse $\widetilde{\sigma} = (\delta_1, \delta_2, \delta_3, \delta_4)$ parse $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ **if** $\delta_1 = H(m, g^{\delta_2}\mathsf{pk}_s^{\delta_1}\sigma_3^{\delta_1})$, $\delta_4 = H(\mathsf{pk}_j, g^{\delta_4}\sigma_3^{\delta_3})$ **return** 1 **else return** 0

Fig.4. A Construction Based on Schnorr

5 Security Analysis

In this section, we provide the security analysis of our two constructed withdrawable signature schemes.

5.1 Security of Our Withdrawable Signature Scheme Based on BLS

Theorem 10. *If the underlying BLS signature scheme* BLS.DS *is unforgeable against chosen-message attacks as defined in Definition 4, our withdrawable signature scheme based on BLS presented in Section 4.3 is unforgeable under insider corruption (Definition 2) in the random oracle model with reduction loss* $L = q_{H_1}$ *where* q_{H_1} *denotes the number of hash queries to the random oracle* H_1.

Proof. Let $\mathcal{B}$ be an adversary that is breaking EUF-CMA of underlying BLS signature scheme BLS.DS. We assume that $\mathcal{B}$ runs another adversary $\mathcal{A}$ which can break unforgeability under insider corruption in the random oracle model of our withdrawable signature scheme based on BLS. **Setup.** $\mathcal{B}$ has access to the simulator $\mathcal{C}$. Suppose $\mathcal{C}$ executes the EUF-CMA game of BLS.DS, denoted as denoted as $\mathsf{Exp}_{\mathcal{A}}^{\text{EUF-CMA}}$ which includes a signing oracle denoted as $\mathcal{O}_{\mathsf{sk}_s}^{\mathsf{BLS.Sign}}(\cdot)$, where $\mathcal{O}_{\mathsf{sk}_s}^{\mathsf{BLS.Sign}}(\cdot) : \omega \leftarrow \mathsf{Sch.Sign}(m, \mathsf{sk}_s)$. $\mathcal{C}$ first generates $(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow \mathsf{KeyGen}(1^k)$, $\mathcal{B}$ then gains pk_s from $\mathcal{C}$.

$\mathcal{B}$ then generates other public keys in $\mathcal{S}$ as $\mathcal{S} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_{s-1}, \mathsf{pk}_{s+1}, \cdots, \mathsf{pk}_\mu\}$.

$\mathcal{B}$ now can set the public key set of the signer and a specific (designated) verifier as $\gamma = \{\mathsf{pk}_s, \mathsf{pk}_j\}$ where $j \neq s$ and provide γ to $\mathcal{A}$. **Oracle Simulation.** $\mathcal{B}$ answers the oracle queries as follows.

Corruption Query. The adversary $\mathcal{A}$ makes secret key queries of $\mathsf{pk}_i, i \in [1,\mu]$ in this phase. If $\mathcal{A}$ queries for the secret key of pk_s, abort. Otherwise, $\mathcal{B}$ returns the corresponding sk_i to $\mathcal{A}$, and adds sk_i to the corrupted secret key list $\mathcal{CO}$.

H-Query. The adversary $\mathcal{A}$ makes hash queries in this phase. $\mathcal{C}$ simulates H_1 as a random oracle, $\mathcal{B}$ then answers the hash queries of H_1 through $\mathcal{C}$.

Signature Query. Adversary $\mathcal{A}$ outputs a message m_i and queries for withdrawable signature with signer pk_s and the specific (designated) verifier pk_j. If the signer isn't pk_s, $\mathcal{B}$ abort. Otherwise, $\mathcal{B}$ sets m_i as the input of $\mathcal{C}$. $\mathcal{B}$ then asks for the signing output of $\mathcal{C}$ as $\omega_i \leftarrow \mathsf{BLS.Sign}(m_i, \mathsf{sk}_s)$. With $\omega_i = H_1(m_i)^{\mathsf{sk}_s}$ from $\mathcal{C}$, $\mathcal{B}$ could respond to the signature query of $\mathcal{A}$ with the specific verifier pk_j as follows:

- $\mathcal{O}^{\mathsf{WSign}}_{\mathsf{sk}_s,\gamma}(\cdot)$: Given the output ω_i of $\mathcal{C}$, $\mathcal{B}$ can compute the withdrawable signature $\sigma_i \leftarrow \mathcal{O}^{\mathsf{WSign}}_{\mathsf{sk}_s,\gamma}(\cdot)$ for $\mathcal{A}$ as:
  1. $r_i, y_i \xleftarrow{\$} \mathbb{Z}_p^*$, $\sigma_i = (e(H_1(m_i)^{y_i r_i}\omega_i, \mathsf{pk}_j), H_1(m_i)^{r_i}, g^{y_i})$
- $\mathcal{O}^{\mathsf{Confirm}}_{\mathsf{sk}_s,\sigma,\gamma}(\cdot)$: With ω_i and σ_i, $\mathcal{B}$ can compute the corresponding confirmed signature $\widetilde{\sigma}_i \leftarrow \mathcal{O}^{\mathsf{Confirm}}_{\mathsf{sk}_s,\sigma,\gamma}(\cdot)$ for $\mathcal{A}$ with underlying signature $\omega_i = H_1(m_i)^{\mathsf{sk}_s}$ and r_i as:
  1. Compute $\delta_{1,i} = \mathsf{pk}_s^{\mathsf{sk}_j r_i}\omega_i$.
  2. Compute $\delta_{2,i} = \mathsf{pk}_j^{r_i}$, $\delta_{3,i} = g^{r_i}$
  3. $\widetilde{\sigma}_i = (\delta_{1,i}, \delta_{2,i}, \delta_{3,i})$

Meanwhile, $\mathcal{B}$ sets $\mathcal{M} \leftarrow \mathcal{M} \cup m_i$ and $\mathcal{W} \leftarrow \mathcal{W} \cup \sigma_i$.

Forgery. On the forgery phase, the adversary $\mathcal{B}$ generates a withdrawable signature σ^* for signer pk_s that designated to verifier pk_j where $\gamma^* = \{\mathsf{pk}_s, \mathsf{pk}_j\}$ on some m^* that has not been queried before, and sends to $\mathcal{A}$. The withdrawable signature σ^* is generated by $\mathcal{B}$ as follows:

$$\sigma^* = (e(H_1(m^*)^{r^* y^*}, \mathsf{pk}_j)e(H_1(m^*)^{\mathsf{sk}_j}, \mathsf{pk}_s), H_1(m^*)^{r^*}, g^{y^*})$$

Then σ^* could be transformed by $\mathcal{A}$ into $\widetilde{\sigma}^*$ under γ^* correctly. After $\mathcal{A}$ transforms σ^* into $\widetilde{\sigma}^*$, if $\widetilde{\sigma}^*$ could not be verified through $\mathsf{CVerify}(m^*, \gamma^*, \sigma^*, \widetilde{\sigma}^*)$ with signer pk_s and certain verifier pk_j, abort.

Otherwise, if $\widetilde{\sigma}^* = (\delta_1^*, \delta_2^*, \delta_3^*)$ is valid, $\mathcal{B}$ then could obtain a forged signature ω^* for pk_s on m^*. Since $\mathcal{B}$ is capable of directly computing $\mathsf{pk}_s^{\mathsf{sk}_j r^*}$, the forged signature ω^* can be determined as: $\omega^* = \delta_1^* / \mathsf{pk}_s^{\mathsf{sk}_j r^*}$.

Therefore, we can use $\mathcal{A}$ to break the unforgeability in the EUF-CMA model of our underlying signature scheme $\mathsf{BLS.DS}$, which contradicts the property of our underlying signature scheme.

Probability of successful simulation. All queried signatures ω_i are simulatable, and the forged signature is reducible because the message m^* cannot be chosen for a signature query. Therefore, the probability of successful simulation is $\frac{1}{q_{H_1}}$ for q_{H_1} queries. □

Theorem 11. *If the DBDH problem is hard, our withdrawable signature scheme based on BLS presented in Section 4.3 is withdrawable (Definition 3) in the random oracle model.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can (t, ε)-break the withdrawability of our withdrawable signature scheme based on the BLS scheme, we construct a simulator $\mathcal{B}$ to solve the DBDH problem. Given problem instance $g, g^a, g^c, g^d \in \mathbb{G}$, $Z \in \mathbb{G}_T$ as input, where $Z = e(g,g)^{acd}$ or $Z = e(g,g)^t$ for random $t \in \mathbb{Z}_p^*$. $\mathcal{B}$ randomly chooses $b \xleftarrow{\$} \{0,1\}$ and $\mu \xleftarrow{\$} \mathbb{Z}_p^*$, and sets the challenge signer/verifier public key set as $\gamma = \{\mathsf{pk}_b, \mathsf{pk}_{1-b}\} = \{g^\mu, g^a\}$, where the associated secret key set is $\delta = \{\mathsf{sk}_b, \mathsf{sk}_{1-b}\} = \{\mu, a\}$. The public key set is available from the two problem instances. The signer is denoted as pk_b and the specific verifier is denoted as pk_{1-b}.

Oracle Simulation. $\mathcal{B}$ answers the oracle queries of $\mathcal{A}$ as follows.

H-Query. The adversary $\mathcal{A}$ makes hash queries in this phase, where $\mathcal{B}$ simulates H_1 as a random oracle.

Signature Query. $\mathcal{A}$ outputs a message m_i and queries for withdrawable signature with corresponding signer pk_b and the specific verifier pk_{1-b}, $\mathcal{B}$ responses the signature query of $\mathcal{A}$ as follows:

– $\mathcal{O}^{\mathsf{WSign}}_{\mathsf{sk}_b,\gamma}(\cdot)$: $r_i, y_i \xleftarrow{\$} \mathbb{Z}_p^*, \sigma_{b,i} = (e(H_1(m_i)^{r_i y_i} H_1(m_i)^{\mathsf{sk}_b}, \mathsf{pk}_{1-b}), H_1(m_i)^{r_i}, g^{y_i})$.

Meanwhile, $\mathcal{B}$ sets $\mathcal{M} \leftarrow \mathcal{M} \cup m_i$.

Challenge. On the challenge phrase, $\mathcal{A}$ gives $\mathcal{B}$ a message $m^* \notin \mathcal{M}$, where $m^* \notin \mathcal{M}$. Upon receiving m^*, $\mathcal{B}$ chooses $k \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $H_1(m^*) = g^k$, $H_1(m^*)^{r^*} = (g^c)^k$ and $g^{y^*} = g^d$. $\mathcal{B}$ then computes the challenge withdrawable signature σ_b^* for $b \xleftarrow{\$} \{0,1\}$ as follows:

$$\sigma_b^* = (Z^k e(H_1(m^*)^{\mu}, g^a), g^{ck}, g^d)$$

Guess. $\mathcal{A}$ outputs a guess b' of b. The simulator outputs true if $b' = b$. Otherwise, false.
Probability of breaking the withdrawability property. We note that the above σ_b^* perfectly simulates the real withdrawable signature when $Z = e(g,g)^{acd}$ since

$$\begin{aligned}\sigma_b^* &= (Z^k e(H_1(m^*)^{\mu}, g^a), g^{ck}, g^d)\\ &= (e(g^{kcd}, g^a) e(H_1(m^*)^a, g^{\mu}), H_1(m^*)^{r^*}, g^{y^*})\\ &= e(H_1(m^*)^{r^* y^*}, \mathsf{pk}_{1-b}) e(H_1(m^*)^{\mathsf{sk}_{1-b}}, \mathsf{pk}_b), H_1(m^*)^{r^*}, g^{y^*})\\ &= e(H_1(m^*)^{r^* y^*} H_1(m^*)^{\mathsf{sk}_b}, \mathsf{pk}_{1-b}), H_1(m^*)^{r^*}, g^{y^*}).\end{aligned}$$

Thus, using $\mathcal{A}$'s ability to decide $b \in \{0,1\}$ (i.e., $\mathcal{A}$ outputs $b' = b$), $\mathcal{B}$ can solve the DBDH problem. However, as the DBDH problem is hard, $\mathcal{A}$'s success probability is bounded by $1/2 + \varepsilon/2$.
Probability of successful simulation. Since there is no abort in our simulation, the probability of successful simulation is 1. □

5.2 Security of the Withdrawable Signature Scheme Based on Schnorr

Theorem 12. *If the underlying Schnorr signature scheme* $\mathsf{Sch.DS}$ *is unforgeable against chosen-message attacks, our withdrawable signature scheme based on Schnorr presented in Section 4.4 is unforgeable under insider corruption (Definition 2) in the random oracle model with reduction loss* $L = 2q_H - 1$ *where* q_H *denotes the number of hash queries to the random oracle* H.

The proof of Theorem 12 follows the same proof structure shown in Proof 5.1, which also contains three algorithms, $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$. The completed proof of Theorem 12 is given as follows.

Proof. Let $\mathcal{B}$ be an adversary that is breaking EUF-CMA of underlying Schnorr signature scheme $\mathsf{Sch.DS}$. We assume that $\mathcal{B}$ runs another adversary $\mathcal{A}$ which can break the unforgeability under insider corruption in the random oracle model of our withdrawable signature scheme based on Schnorr.
Setup. $\mathcal{B}$ has access to the simulator $\mathcal{C}$. Suppose $\mathcal{C}$ executes the EUF-CMA game of $\mathsf{Sch.DS}$, denoted as $\mathsf{Exp}_{\mathcal{A}}^{\text{EUF-CMA}}$ which includes a signing oracle $\mathcal{O}^{\mathsf{Sch.Sign}}_{\mathsf{sk}_s}(\cdot)$, where $\mathcal{O}^{\mathsf{Sch.Sign}}_{\mathsf{sk}_s}(\cdot) : \omega \leftarrow \mathsf{Sch.Sign}(m, \mathsf{sk}_s)$. $\mathcal{C}$ first generates $(\mathsf{pk}_s, \mathsf{sk}_s) \leftarrow \mathsf{KeyGen}(1^k)$, $\mathcal{B}$ then gains pk_s from $\mathcal{C}$.

$\mathcal{B}$ then generates other public keys in $\mathcal{S}$ as $\mathcal{S} = \{\mathsf{pk}_1, \cdots, \mathsf{pk}_{s-1}, \mathsf{pk}_{s+1}, \cdots, \mathsf{pk}_{\mu}\}$ and gains pk_s from $\mathcal{C}$.

$\mathcal{B}$ now can set the public key set of the signer and a specific (designated) verifier as $\gamma = \{\mathsf{pk}_s, \mathsf{pk}_j\}$ where $j \neq s$ and provide γ to $\mathcal{A}$.
Oracle Simulation. $\mathcal{B}$ answers the oracle queries of $\mathcal{A}$ as follows.
Corruption Query. The adversary $\mathcal{A}$ makes secret key queries of public key $\mathsf{pk}_i, i \in [1, \mu]$ in this phase. If $\mathcal{A}$ queries for the secret key of pk_s, abort. Otherwise, $\mathcal{B}$ returns the corresponding sk_i to $\mathcal{A}$, and add sk_i to the corrupted secret key list $\mathcal{CO}$.
H-Query. $\mathcal{C}$ simulates H as a random oracle, $\mathcal{B}$ then answers the hash queries of H through $\mathcal{C}$.
Signature Query. $\mathcal{A}$ outputs a message m_i and queries for withdrawable signature with corresponding signer pk_s and specific verifier pk_j. If the signer of withdrawable dignature isn't pk_s, abort. Otherwise, $\mathcal{B}$ sets m_i as the input of $\mathcal{C}$. $\mathcal{B}$ then asks the signing output of $\mathcal{C}$ as $\omega_i = \mathsf{Sch.Sign}(m_i, \mathsf{sk}_s)$. With ω_i, $\mathcal{B}$ could response the signature query for the specific verifier pk_j chosen by $\mathcal{A}$ as follows:

– $\mathcal{O}^{\mathsf{WSign}}_{\mathsf{sk}_s,\gamma}(\cdot)$: With the output of $\mathcal{C}$, $\mathcal{B}$ can compute the withdrawable signature $\sigma_i \leftarrow \mathcal{O}^{\mathsf{WSign}}_{\mathsf{sk}_s,\gamma}(\cdot)$ for $\mathcal{A}$ with $\omega_i = (t_i, z_i) = (H(m_i, u_i), z_i)$ as:
  1. Randomly choose $r_i \xleftarrow{\$} \mathbb{Z}_p^*$

2. Compute $\sigma_{1,i} = g^{z_i}\mathsf{pk}_s^{t_i}$, $\sigma_{2,i} = \mathsf{pk}_j^{z_i - r_i t_i}$, $\sigma_{3,i} = g^{r_i}$
3. $\sigma_i = (\sigma_{1,i}, \sigma_{2,i}, \sigma_{3,i})$

– $\mathcal{O}^{\mathsf{Confirm}}_{\mathsf{sk}_s,\sigma,\gamma}(\cdot)$: $\mathcal{B}$ then queries for the Schnorr signature of m_i again to $\mathcal{C}$ and returns a corresponding $\omega_{s,i} = (t_{s,i}, z_{s,i})$ instead. With ω_i, $\omega_{s,i}$ and σ_i, $\mathcal{B}$ can compute the confirmed signature $\widetilde{\sigma}_i \leftarrow \mathcal{O}^{\mathsf{Confirm}}_{\mathsf{sk}_s,\sigma,\gamma}(\cdot)$ for $\mathcal{A}$ as follows:
1. Compute $\delta_{1,i} = g^{z_{s,i}}\mathsf{pk}_s^{t_{s,i}}$, $\delta_{2,i} = z_{s,i} - r_i t_{s,i}$
2. Randomly choose $e_{j,i}, t_{j,i} \xleftarrow{\$} \mathbb{Z}_p^*$, $\delta_{3,i} = t_{j,i}$
3. Compute $\delta_{4,i} = e_{j,i} - r_i t_{j,i}$
4. $\widetilde{\sigma}_i = (\delta_{1,i}, \delta_{2,i}, \delta_{3,i}, \delta_{4,i})$

Meanwhile, $\mathcal{B}$ sets the queried message set as $\mathcal{M} \leftarrow \mathcal{M} \cup m_i$ and queried withdrawable signature set as $\mathcal{W} \leftarrow \mathcal{W} \cup \sigma_i$.

Forgery. On the forgery phase, $\mathcal{B}$ returns a withdrawable signature σ^* for $\gamma^* = \{\mathsf{pk}_s, \mathsf{pk}_j\}$ on some m^* that has not been queried before. Then σ^* could be transformed into $\widetilde{\sigma}^*$ under γ^* for signer pk_s correctly. After $\mathcal{A}$ transforms σ^* into $\widetilde{\sigma}^*$, if $\widetilde{\sigma}^*$ could not be verified through $\mathsf{CVerify}(m^*, \gamma^*, \sigma^*, \widetilde{\sigma}^*)$, abort.

Otherwise, if $\widetilde{\sigma}^* = (\delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*)$ is valid, $\mathcal{B}$ then could obtain a forged signature ω^* for pk_s on m^*. Since $\mathcal{B}$ is capable of directly computing $r^* t_s^*$, the forged signature ω^* can be determined as $\omega^* = \delta_2^* + r^* t_s^*$.

Therefore, we can use $\mathcal{A}$ to break the unforgeability in the EUF-CMA model of our underlying signature scheme $\mathsf{Sch.DS}$, which contradicts the property of our underlying signature scheme.

Probability of successful simulation. All queried signatures ω_i are simulatable, and the forged signature is reducible because the message m^* cannot be chosen for a signature query as it will be used for the signature forgery. Therefore, the probability of successful simulation is $\frac{1}{2q_H - 1}$. □

Theorem 13. *If the DDH problem is hard, our withdrawable signature scheme based on Schnorr presented in Section 4.4 is withdrawable (Definition 3) in the random oracle model.*

The complete detailed proof of Theorem 13 is given as follows.

Proof. Suppose there exists an adversary $\mathcal{A}$ who can (t, ε)-break the withdrawability of our withdrawable signature scheme based on Schnorr, we construct a simulator $\mathcal{B}$ to solve the DDH problem. Given as input DDH problem instance $g, g^a, g^c, Z \in \mathbb{G}$, where $Z = g^{ac}$ or $Z = g^x$ for random $x \in \mathbb{Z}_p^*$. $\mathcal{B}$ sets the challenge public key set as $\gamma = \{\mathsf{pk}_b, \mathsf{pk}_{1-b}\} = \{g^\mu, g^a\}$ and associated secret key set $\delta = \{\mathsf{sk}_b, \mathsf{sk}_{1-b}\} = \mu, a$. The public key set is available from the problem instance. The signer is denoted as pk_b where $b \xleftarrow{\$} \{0,1\}$, and the specific verifier is denoted as pk_{1-b}.

Oracle Simulation. $\mathcal{B}$ answers the oracle queries as follows.

H-Query. The adversary $\mathcal{A}$ makes hash queries in this phase where $\mathcal{B}$ simulates H as a random oracle.

Signature Query. $\mathcal{A}$ outputs a message m_i and queries the withdrawable signature for corresponding signer pk_b and specific verifier pk_{1-b}, $\mathcal{B}$ responses the signature queries of $\mathcal{A}$ as follows:

– $\mathcal{O}^{\mathsf{WSign}}_{\mathsf{sk}_b,\gamma}(\cdot)$: $e_i, r_i \xleftarrow{\$} \mathbb{Z}_p^*$, $t_i = H(m_i, g^{e_i})$, $\sigma_{b,i} = (g^{e_i}, \mathsf{pk}_{1-b}^{e_i - \mathsf{sk}_b t_i - r_i t_i}, g^{r_i})$

Meanwhile, $\mathcal{B}$ sets $\mathcal{M} \leftarrow \mathcal{M} \cup m_i$.

Challenge. On the challenge phase, $\mathcal{A}$ gives $\mathcal{B}$ a message m^*, where $m^* \notin \mathcal{M}$. Upon receiving m^*, $\mathcal{B}$ chooses $d, k \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $r^* = d$, $t^* = k = H(m, g^{e^*})$, $e^* = r^* t^* + c = dk + c$, to compute the challenge withdrawable signature of m^* as σ_b^* for $b \xleftarrow{\$} \{0,1\}$ with Z as follows:

$$\sigma_b^* = (g^{dk+c}, Zg^{-a\mu k}, g^d).$$

Guess. $\mathcal{A}$ outputs a guess b' of b. The simulator outputs true if $b' = b$. Otherwise, false.

Probability of breaking the withdrawability property. We note that the above σ_b^* perfectly simulates the real withdrawable signature when $Z = g^{ac}$ since

$$\begin{aligned}
\sigma_b^* &= (g^{dk+c}, Zg^{-a\mu k}, g^d) \\
&= (g^{dk+c}, g^{ac}g^{-a\mu k}, g^d) \\
&= (g^{dk+c}, (g^a)^{c-\mu k}, g^d) \\
&= (g^{dk+c}, (g^a)^{dk+c-dk-\mu k}, g^d) \\
&= (g^{e^*}, \mathsf{pk}_{1-b}^{e^* - r^* t^* - \mathsf{sk}_b t^*}, g^{r^*}).
\end{aligned}$$

Meanwhile, g^c can be computed from $g^c = g^{e^*}(g^{r^*})^{-t^*}$ with g^{e^*}, g^{r^*} and $t^* = H(m^*, g^{e^*})$. Thus, using $\mathcal{A}$'s ability to decide $b \in \{0,1\}$ (i.e., $\mathcal{A}$ outputs $b' = b$), $\mathcal{B}$ can solve the DDH hard problem. However, as the DDH problem is hard, $\mathcal{A}$'s success probability is bounded by $1/2 + \varepsilon/2$.
Probability of successful simulation. Since there is no abort in our simulation. Therefore, the probability of successful simulation is 1. □

6 Conclusion

In this paper, we discussed the challenges associated with traditional signature schemes and the need for a mechanism to revoke or replace signatures. We introduced a unique withdrawability feature for signature schemes, allowing signers to have the ability to call off their signatures as withdrawable signatures, and later, the signature could be transformed into a confirmed signature that could be verified through their public keys.

Furthermore, we proposed cryptographic primitives and two constructions of the withdrawable signature based on the BLS/Schnorr signature. We formally proved that the two proposed constructions are unforgeable under insider corruption and satisfy withdrawability.

There are several directions for future work: one is improving the efficiency of our withdrawable signature scheme. Exploring further to discover practical applications and use cases of withdrawable signature schemes can also be an interesting avenue for future work.

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: Zheng, Y. (ed.) Advances in Cryptology — ASIACRYPT 2002. pp. 415–432. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
2. Abhilash, M., Amberker, B.: Efficient group signature scheme using lattices. International Journal of Information Technology **14**(4), 1845–1854 (2022)
3. Attrapadung, N., Emura, K., Hanaoka, G., Sakai, Y.: Revocable group signature with constant-size revocation list. The Computer Journal **58**(10), 2698–2715 (2015)
4. Backes, M., Döttling, N., Hanzlik, L., Kluczniak, K., Schneider, J.: Ring signatures: logarithmic-size, no setup—from standard assumptions. In: Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38. pp. 281–311. Springer (2019)
5. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3. pp. 60–79. Springer (2006)
6. Beullens, W., Dobson, S., Katsumata, S., Lai, Y.F., Pintore, F.: Group signatures and more from isogenies and lattices: generic, simple, and efficient. Designs, Codes and Cryptography pp. 1–60 (2023)
7. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22. pp. 416–432. Springer (2003)
8. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7. pp. 514–532. Springer (2001)
9. Chaum, D., Van Heyst, E.: Group signatures. In: Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10. pp. 257–265. Springer (1991)
10. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23. pp. 609–626. Springer (2004)
11. Esgin, M.F., Steinfeld, R., Liu, J.K., Liu, D.: Lattice-based zero-knowledge proofs: new techniques for shorter and faster constructions and applications. In: Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I. pp. 115–146. Springer (2019)
12. Esgin, M.F., Steinfeld, R., Sakzad, A., Liu, J.K., Liu, D.: Short lattice-based one-out-of-many proofs and applications to ring signatures. In: Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17. pp. 67–88. Springer (2019)

13. Esgin, M.F., Steinfeld, R., Zhao, R.K.: Matrict+: More efficient post-quantum private blockchain payments. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 1281–1298. IEEE (2022)
14. Gritti, C., Susilo, W., Plantard, T.: Logarithmic size ring signatures without random oracles. IET Information Security **10**(1), 1–7 (2016)
15. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II. pp. 253–280. Springer (2015)
16. Herranz, J., Sáez, G.: Forking lemmas for ring signature schemes. In: Progress in Cryptology-INDOCRYPT 2003: 4th International Conference on Cryptology in India, New Delhi, India, December 8-10, 2003. Proceedings 4. pp. 266–279. Springer (2003)
17. Horne, B., Pinkas, B., Sander, T.: Escrow services and incentives in peer-to-peer networks. In: Proceedings of the 3rd ACM Conference on Electronic Commerce. pp. 85–94 (2001)
18. Huang, X., Susilo, W., Mu, Y., Wu, W.: Secure universal designated verifier signature without random oracles. International Journal of Information Security **7**, 171–183 (2008)
19. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 143–154. Springer (1996)
20. Kurbatov, O., Kravchenko, P., Poluyanenko, N., Shapoval, O., Kuznetsova, T.: Using ring signatures for an anonymous e-voting system. In: 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT). pp. 187–190. IEEE (2019)
21. Li, B., Liu, Y., Yang, S.: Lattice-based universal designated verifier signatures. In: 2018 IEEE 15th International Conference on e-Business Engineering (ICEBE). pp. 329–334. IEEE (2018)
22. Li, Y., Susilo, W., Mu, Y., Pei, D.: Designated verifier signature: definition, framework and new constructions. In: International Conference on Ubiquitous Intelligence and Computing. pp. 1191–1200. Springer (2007)
23. Libert, B., Vergnaud, D.: Group signatures with verifier-local revocation and backward unlinkability in the standard model. In: CANS. vol. 9, pp. 498–517. Springer (2009)
24. Liu, D.Y., Liu, J.K., Mu, Y., Susilo, W., Wong, D.S.: Revocable ring signature. Journal of Computer Science and Technology **22**, 785–794 (2007)
25. Liu, J.K., Wong, D.S.: Linkable ring signatures: Security models and new schemes. In: Computational Science and Its Applications–ICCSA 2005: International Conference, Singapore, May 9-12, 2005, Proceedings, Part II 5. pp. 614–623. Springer (2005)
26. Lyubashevsky, V., Nguyen, N.K.: Bloom: Bimodal lattice one-out-of-many proofs and applications. In: Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part IV. pp. 95–125. Springer (2023)
27. Nakanishi, T., Fujii, H., Hira, Y., Funabiki, N.: Revocable group signature schemes with constant costs for signing and verifying. IEICE transactions on fundamentals of electronics, communications and computer sciences **93**(1), 50–62 (2010)
28. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7. pp. 552–565. Springer (2001)
29. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Advances in Cryptology—CRYPTO'89 Proceedings 9. pp. 239–252. Springer (1990)
30. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Advances in Cryptology-ASIACRYPT 2003: 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30–December 4, 2003. Proceedings 9. pp. 523–542. Springer (2003)
31. Steinfeld, R., Wang, H., Pieprzyk, J.: Efficient extension of standard schnorr/rsa signatures into universal designated-verifier signatures. In: Public Key Cryptography–PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004. Proceedings 7. pp. 86–100. Springer (2004)
32. Szabo, N.: The idea of smart contracts. Nick Szabo's papers and concise tutorials **6**(1), 199 (1997)
33. Thanalakshmi, P., Anbazhagan, N., Joshi, G.P., Yang, E.: A quantum resistant universal designated verifier signature proof. AIMS Mathematics **8**(8), 18234–18250 (2023)
34. Thorncharoensri, P., Susilo, W., Baek, J.: Aggregatable certificateless designated verifier signature. IEEE Access **8**, 95019–95031 (2020)
35. Tian, H., Chen, X., Li, J.: A short non-delegatable strong designated verifier signature. In: Information Security and Privacy: 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings 17. pp. 261–279. Springer (2012)
36. Xin, X., Ding, L., Li, C., Sang, Y., Yang, Q., Li, F.: Quantum public-key designated verifier signature. Quantum Information Processing **21**(1), 33 (2022)

37. Yamashita, K., Hara, K., Watanabe, Y., Yanai, N., Shikata, J.: Designated verifier signature with claimability. In: Proceedings of the 10th ACM Asia Public-Key Cryptography Workshop. pp. 21–32 (2023)
38. Yang, M., Shen, X.q., Wang, Y.m.: Certificateless universal designated verifier signature schemes. The Journal of China Universities of Posts and Telecommunications **14**(3), 85–94 (2007)
39. Yuen, T.H., Esgin, M.F., Liu, J.K., Au, M.H., Ding, Z.: Dualring: generic construction of ring signatures with efficient instantiations. In: Annual International Cryptology Conference. pp. 251–281. Springer (2021)
40. Zhang, R., Furukawa, J., Imai, H.: Short signature and universal designated verifier signature without random oracles. In: Applied Cryptography and Network Security: Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005. Proceedings 3. pp. 483–498. Springer (2005)
41. Zhang, X., Liu, J.K., Steinfeld, R., Kuchta, V., Yu, J.: Revocable and linkable ring signature. In: Information Security and Cryptology: 15th International Conference, Inscrypt 2019, Nanjing, China, December 6–8, 2019, Revised Selected Papers 15. pp. 3–27. Springer (2020)